

```
File Edit View Navigate Code Refactor Build Run Tools Git Window Help spring6 [...\spring6\spring6] - LogAspect.java [spring6-aop]
g6-aop | src | main | java | com | atguigu | spring6 | aop | xmlaop | LogAspect
Project | Pr... | x | xmlaop\CalculatorImpl.java | x TestAop.java | x bean.xml | x beanaop.xml | x LogAspect.java | x ex... | x
Commit | spring6 | C:\Users\QMacroQA\Desktop | spring6 | atguigu-spring | spring6-junit | spring6-aop | src | main | java | com | atguigu | spring6-aop | annoaop | example | xmlaop | Calculator | CalculatorImpl | LogAspect | TestAop | Main | resources | bean.xml | beanaop.xml | test | target | .gitignore | pom.xml | spring6-i18n | spring6-ioc-xml | spring6-reflect | spring6-resources | spring6-validator | spring-first | spring-ioc-annotation | spring-jdbc-tx |
Pull Requests |
Bookmarks |
Structure |
LogAspect.java
1 package com.atguigu.spring6.aop.xmlaop;
2
3 import ...
4
5 //切面类
6 @Component // IOC容器
7 public class LogAspect {
8
9     // 前置通知
10    no usages | eva
11    @Before
12    public void beforeMethod(JoinPoint joinPoint){
13        String methodName = joinPoint.getSignature().getName();
14        Object[] args = joinPoint.getArgs();
15        System.out.println("Logger-->前置通知, 方法名称: "+methodName+", 参数: "+ Arrays.toString(args));
16    }
17
18    // 后置通知
19    no usages | eva
20    @After
21    public void afterMethod(JoinPoint joinPoint){
22        String methodName = joinPoint.getSignature().getName();
23        System.out.println("Logger-->后置通知, 方法名称: "+methodName);
24    }
25
26    // 返回通知
27    no usages | eva
28    @AfterReturning
29    public void afterReturning(JoinPoint joinPoint, Object result){
30        String methodName = joinPoint.getSignature().getName();
31        System.out.println("Logger-->返回通知, 方法名称: "+methodName+", 返回名称: "+ result);
32    }
33
34    // 异常通知
35    no usages | eva
36    @AfterThrowing
37    public void afterThrowing(JoinPoint joinPoint, Throwable throwable){
38        String methodName = joinPoint.getSignature().getName();
39        System.out.println("Logger-->异常通知, 方法名称: "+methodName);
40    }
41
42    @Around
43    public void aroundMethod(JoinPoint joinPoint, ProceedingJoinPoint proceedingJoinPoint) throws Throwable {
44        String methodName = joinPoint.getSignature().getName();
45        System.out.println("Logger-->环绕通知, 方法名称: "+methodName);
46        proceedingJoinPoint.proceed();
47    }
48
49 }
```

```
File Edit View Navigate Code Refactor Build Run Tools Git Window Help spring [C:\Users\QMacroQA\Desktop\code\spring6\spring6] - beanaop.xml [spring6-aop]
spring | spring6-aop | src | main | resources | beanaop.xml
Project | spring | C:\Users\QMacroQA\Desktop | spring6 | atguigu-spring | spring6-junit | spring6-aop | src | main | resources | beanaop.xml | bean.xml | test | target | .gitignore | pom.xml | spring6-i18n | spring6-ioc-xml | spring6-reflect | spring6-resources | spring6-validator | spring-first | spring-ioc-annotation | spring-jdbc-tx |
Pull Requests |
Bookmarks |
Structure |
beanaop.xml
1 <!-- 开启组件扫描 -->
2 <context:component-scan base-package="com.atguigu.spring6.xmlaop"/>
3
4 <!-- 配置AOP五种通知类型 -->
5 <aop:config>
6     <!-- 配置切面类 logAspect需要注意注解 -->
7     <aop:aspect ref="LogAspect">
8         <aop:pointcut id="pointcut" expression="execution(* com.atguigu.spring6.xmlaop.CalculatorImpl.*(..))"/>
9         <!-- 配置五种通知类型 -->
10        <!-- 配置切入点 -->
11        <!-- 前置通知 -->
12        <aop:before method="beforeMethod" pointcut-ref="pointcut"/>
13        <!-- 后置通知 -->
14        <aop:after method="afterMethod" pointcut-ref="pointcut"/>
15        <!-- 返回通知 -->
16        <aop:after-returning method="afterReturning" returning="result" pointcut-ref="pointcut"/>
17        <!-- 异常通知 -->
18        <aop:after-throwing method="afterThrowing" pointcut-ref="pointcut" throwing="ex"/>
19        <!-- 环绕通知 -->
20        <aop:around method="around" pointcut-ref="pointcut"/>
21    </aop:aspect>
22 </aop:config>
23 </beans>
```