# 绿色农产品质量分析报告

## 1：关于spring中的AOP

AOP是Spring框架面向切面的编程思想，AOP采用一种称为“横切”的技术，将涉及多业务流程的通用功能抽取并单独封装，形成独立的切面，在合适的时机将这些切面横向切入到业务流程指定的位置中。

Spring框架的AOP机制可以让开发者把业务流程中的通用功能抽取出来，单独编写功能代码。在业务流程执行过程中，Spring框架会根据业务流程要求，自动把独立编写的功能代码切入到流程的合适位置。

## 2：****基于注解的AOP配置方式****

1. package com.enjoy.cap10.aop;
3. import org.aspectj.lang.ProceedingJoinPoint;
4. import org.aspectj.lang.annotation.After;
5. import org.aspectj.lang.annotation.AfterReturning;
6. import org.aspectj.lang.annotation.AfterThrowing;
7. import org.aspectj.lang.annotation.Before;
8. import org.aspectj.lang.annotation.Pointcut;
9. import org.aspectj.lang.annotation.Around;
10. import org.aspectj.lang.annotation.Aspect;
12. *//日志切面类*
13. @Aspect
14. public class LogAspects {
15. @Pointcut("execution(public int com.enjoy.cap10.aop.Calculator.\*(..))")
16. public void pointCut(){};
18. *//@before代表在目标方法执行前切入, 并指定在哪个方法前切入*
19. @Before("pointCut()")
20. public void logStart(){
21. System.out.println("除法运行....参数列表是:{}");
22. }
23. @After("pointCut()")
24. public void logEnd(){
25. System.out.println("除法结束......");
26. }
27. @AfterReturning("pointCut()")
28. public void logReturn(){
29. System.out.println("除法正常返回......运行结果是:{}");
30. }
31. @AfterThrowing("pointCut()")
32. public void logException(){
33. System.out.println("运行异常......异常信息是:{}");
34. }
35. @Around("pointCut()")
36. public Object Around(ProceedingJoinPoint proceedingJoinPoint) throws Throwable{
37. System.out.println("@Arount:执行目标方法之前...");
38. Object obj = proceedingJoinPoint.proceed();*//相当于开始调div地*
39. System.out.println("@Arount:执行目标方法之后...");
40. return obj;
41. }
42. }

**目标方法：**

1. package com.enjoy.cap10.aop;
3. public class Calculator {
4. *//业务逻辑方法*
5. public int div(int i, int j){
6. System.out.println("--------");
7. return i/j;
8. }
9. }

**配置类：**

1. public class Cap10MainConfig {
2. @Bean
3. public Calculator calculator(){
4. return new Calculator();
5. }
7. @Bean
8. public LogAspects logAspects(){
9. return new LogAspects();
10. }
11. }

**测试类：**

1. public class Cap10Test {
2. @Test
3. public void test01(){
4. AnnotationConfigApplicationContext app = new AnnotationConfigApplicationContext(Cap10MainConfig.class);
5. Calculator c = app.getBean(Calculator.class);
6. int result = c.div(4, 3);
7. System.out.println(result);
8. app.close();
10. }}

## 3：Spring中AOP主要有以下用途：

1、Authentication 权限

2、Caching 缓存

3、Context passing 内容传递

4、Error handling 错误处理

5、Lazy loading　懒加载

6、Debugging　调试

7、logging, tracing, profiling and monitoring　记录跟踪　优化　校准

8、Performance optimization　性能优化

9、Persistence　　持久化

10、Resource pooling　资源池

11、Synchronization　同步

12、Transactions 事务

## 4：关于该项目中使用spring的AOP的好处

在使用Spring AOP 进行项目开发时，操作容易，简单。并且还有利于巩固该课程知识，在进行软件测试的时候，测试方便，精确，该项目中也运用到了相关知识点，为开发项目打下了一定的基础。