步骤:

- a、导入必备的6个ioc的和4个aop的jar包
- b、在类的根路径下创建一个bean.xml的配置文件
- c、导入ioc/aop和context三个约束

<beans xmlns="http://www.springframework.org/schema/beans"</pre> xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop" xmlns:context="http://www.springframework.org/schema/context" xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd"> <!-- 配置spring要扫描的包 --> <context:component-scan base-package="com.itheima"></context:component-scan> <!-- 开启spring对注解AOP的支持 --> <aop:aspectj-autoproxy/> </beans> Kbeans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" http://www.springframework.org/schema/aop/ xmlns:context="http://www.springframework.org/schema xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd http://www.springframework.org/schema/context www.springframework.org/schema/context/spring-context.xsd/ <!-- 配置spring要扫描的包 -->

d、把资源都用注解的方式让spring来管理

<context:component-scan base-package="com. itheima"></context:component-scan>

e、使用注解来配置切面

<!-- 开启spring对注解AOP的支持 -->

<aop:aspectj-autoproxy/>

</beans>

注解: @Aspect

作用: 指定当前类是一个切面类

f、使用@Before注解来配置前置通知

g、使用注解AOP,必须在xml配置文件中开启

<aop:aspectj-autoproxy/>

AOP常用注解

@Aspect: 指定当前类是一个切面

@Before: 指定当前方法前置通知

属性:

value: 指定切入点表达式|切入点表达式引用

@AfterReturning: 指定当前方法后置通知

属性:

value: 指定切入点表达式|切入点表达式引用

@AfterThrowing: 指定当前方法异常通知

属性:

value: 指定切入点表达式|切入点表达式引用

@After: 指定当前方法是最终通知

属性:

value: 指定切入点表达式|切入点表达式引用

@Around: 指定当前方法是环绕通知

属性:

value: 指定切入点表达式|切入点表达式引用

```
@Component("logger") IOC注解
@Aspect AOP注解
public class Logger {

/**
    * 计划让其成为前置通知:
    * 永远在业务核心方法之前执行
    */

// @Before(value="execution(* com. itheima. service. impl.*.*(..))")
    public void beforePrintLog() {
        System. out. println("Logger类中的beforePrintLog方法开始记录日志了。。。。。");
    }
```