注解方式的AOP开发步骤

1.添加Spring开发必要的jar包
commons-logging-1.1.3.jar
spring-beans-4.0.0.RELEASE.jar
spring-context-4.0.0.RELEASE.jar
spring-core-4.0.0.RELEASE.jar
spring-expression-4.0.0.RELEASE.jar

2.在applicationContext.xml添加context命名空间

3.配置自动扫描包的注解：
<context:component-scan base-package = "com.spring"/>

1.添加aop相关jar包
com.springsource.org.aopalliance-1.0.0.jar
com.springsource.org.aspectj.weaver-1.6.8.RELEASE.jar
spring-aop-4.0.0.RELEASE.jar
spring-aspects-4.0.0.RELEASE.jar

2.在applicationContext.xml添加AOP命名空间

3.配置是切面里面的注解起作用(@Befeore)：
<aop:aspectj-autoproxy />

4.在切面类上添加注解:
@Aspect和@Component

5.在切面类里面的方法中添加通知注解:
@Before("execution(* com.*.*.*(..))")

1.把横切关注点代码抽象到切面类中

2.通知的类型：
@Before，@After
@AfterReturning,@AfterThrowing
@Around

基于配置文件的方式AOP开发步骤

1.添加Spring开发必要的jar包
commons-logging-1.1.3.jar
spring-beans-4.0.0.RELEASE.jar
spring-context-4.0.0.RELEASE.jar
spring-core-4.0.0.RELEASE.jar
spring-expression-4.0.0.RELEASE.jar

2.配置 bean
<bean id="arithmeticCalculator"

class="com.spring.aop.service.impl.ArithmeticCalculatorImpl">
</bean>

1.添加aop相关jar包
com.springsource.org.aopalliance-1.0.0.jar
com.springsource.org.aspectj.weaver-1.6.8.RELEASE.jar
spring-aop-4.0.0.RELEASE.jar
spring-aspects-4.0.0.RELEASE.jar

2.在applicationContext.xml添加AOP命名空间

3.配置切面的bean
<bean id="loggingAspect"
class="com.spring.aop.aspect.LoggingAspect">
</bean>
4.配置AOP
<aop:config>
<!-- A: 配置切点表达式 -->
<aop:pointcut expression="execution(* com.spring.aop.service.impl.*.*(..))"
id="ponitcut"/>
<!-- B: 配置切面及通知 -->
<aop:aspect ref="loggingAspect">
<aop:before mehtod="beforeMethod" pointcut-ref="ponitcut" />
</aop:aspect>
</aop:config>

接口ArithmeticCalculator
+ add(int,int):int
+ sub(int,int):int
+ mul(int,int):int
+ div(int,int):int

LoggingAspect
+ attribute1:type = defaultValue
+ attribute2:type
- attribute3:type
+ beforeMethod(JoinPoint):void
+ afterMethod(JoinPoint):void

aop:advisor 与 aop:aspect的区别

1. aop:advisor 配置只能配置一个切点

2. aop:advisor 配置的切面，切面逻辑必须实现advice接口

3. aop:aspect 配置，可以根据不同的情况配置不同的切点

4. aop:aspect 配置的切面，切面逻辑可以自定义