**重点内容**

Spring、AspectJ和AOP之间的关系？

对于AOP这种编程思想，很多框架都进行了实现。Spring就是其中之一，可以完成面向切面编程。然而，AspectJ也实现了AOP的功能，且其实现方式更为简捷，使用更为方便，而且还支持注解式开发。所以，Spring又将AspectJ的对于AOP的实也引入到了自己的框架中。

1. AspectJ简介

AspectJ是一个面向切面的框架，它扩展了Java语言。AspectJ定义了AOP语法，所以它有一个专门的编译器用来生成遵守Java字节编码规范的Class文件。

1. AspectJ的通知类型

AspectJ中常用的通知有五种类型：

1. 前置通知
2. 后置通知
3. 环绕通知
4. 异常通知
5. 最终通知

其中最终通知是指，无论程序执行是否正常，该通知都会执行。类似于try。。。catch中的finally代码块。

1. AspectJ的切入点表达式

AspectJ除了提供了六种通知外，还定义了专门的表达式用于指定切入点。表达式的原型是：

execution([modifiers-pattern]) 访问权限类型

ret-type-pattern 返回值类型

[declaring-type=pattern] 全限定性类名

name-pattern(param-pattern) 方法名（参数名）

[throws-pattern] 抛出异常类型

切入点表达式要匹配的对象就是目标方法的方法名。所以，execution表达式中明显就是方法的签名。注意，表达式中加[]的部分表示可以省略，各部分间用空格分开。在其中可以使用以下符号：



1. AspectJ的开发环境
2. AspectJ基于注解的AOP实现
3. 实现步骤
4. 1
5. 1
6. 1
7. 1
8. 1
9. 1
10. 1
11. @Before前置通知-方法有JoinPoint参数
12. @AfterReturning后置通知-注解有returning属性
13. @Around环绕通知-增强方法有ProceedingJoinPoint参数
14. @AfterThrowing异常通知-注解中有throwing属性
15. @After最终通知
16. @Pointcut定义切入点
17. AspectJ基于XML的AOP实现
18. 实现步骤
19. 1
20. 1
21. 1
22. 1
23. 1
24. <aop:before/>前置通知
25. <aop:after-returning/>后置通知
26. <aop:around/>环绕通知
27. <aop:after-Throwing/>异常通知
28. <aop:after>最终通知