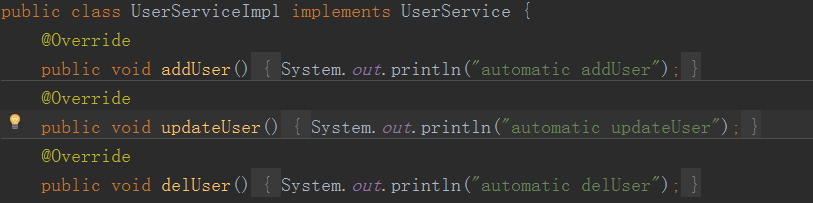
**Spring框架**

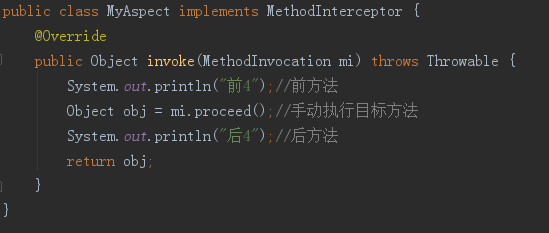
1. **AOP编程（重点，全自动）**
2. 从spring容器中获得的就是目标对象，如果配置了aop，spring将自动生成代 理对象
3. 要确定目标类，需要aspectj切入点表达式（重点）
4. 需要导入aspectj.weaver.jar包
5. Aop编程共需要8个jar包：



1. Aop编程具体步骤：
2. 编写目标类（接口和实现类），接口省略

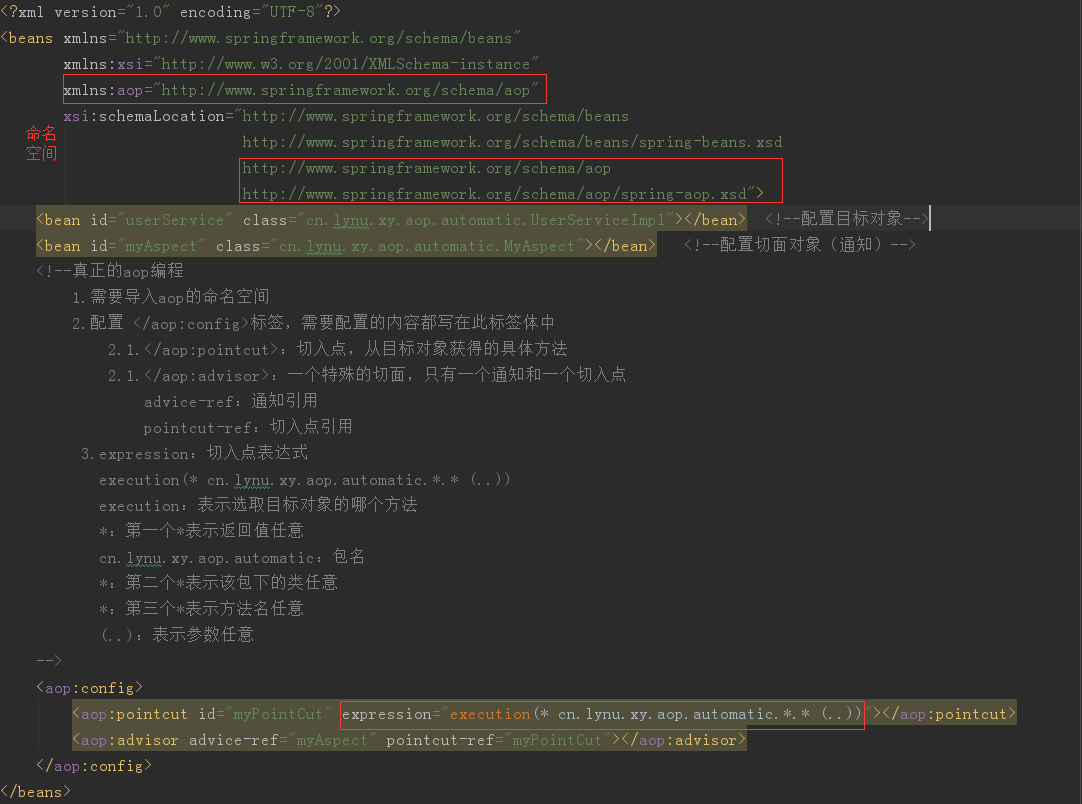


1. 编写切面类

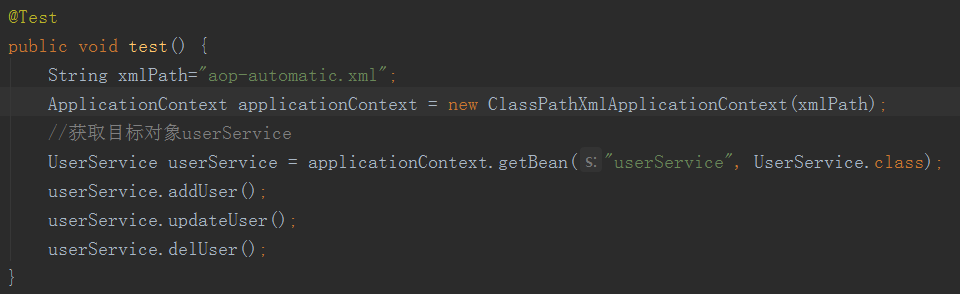


注释：切面类需要实现不同的接口（通知类型）  
 切面类中确定通知类型，本次采用环绕通知：MethodInterceptor  
 接口就是规范，从而确定方法名

1. 编写xml配置文件

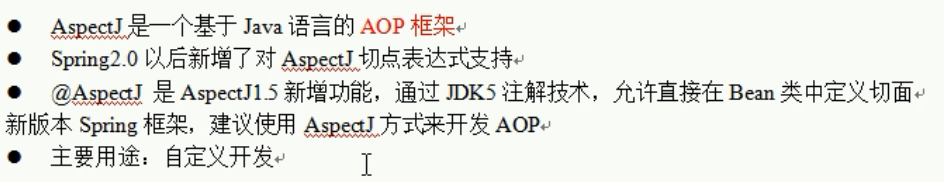


1. 测试



**注：这里获取的是目标对象，但是spring底层拿到的其实是代理对象**

1. **ASpectJ**
2. 介绍



1. **切入点表达式（重点掌握）**
2. **execution（）：用于描述方法**
3. **语法格式：execution（修饰符 返回值 包.类.方法名（参数） throws异常）**

**修饰符：一般省略**

**返回值：不能省略**

**void（无返回值）**

**string（返回字符串）**

**\*（任意）**

**包：可以省略**

**cn.lynu.xy 表示固定的包**

**cn.lynu.xy.\*.service 表示xy包下的任意子包下的service包**

**cn.lynu.xy.. 表示xy包及下面的所有子包**

**cn.lynu.xy.\*.service.. 表示xy包下的任意子包下的service包及service包下的所 有子包**

**类：可以省略**

**UserServiceImpl 表示指定类**

**\*Impl 表示以Impl结尾的类**

**User\* 表示以User开头的类**

**\* 表示任意**

**方法名：不能省略**

**addUser 表示固定方法**

**add\* 表示以add开头的方法**

**\*Do 表示以Do结尾的方法**

**（参数）：**

**（） 表示无参**

**（int） 表示一个整型**

**（int，int） 表示两个整型**

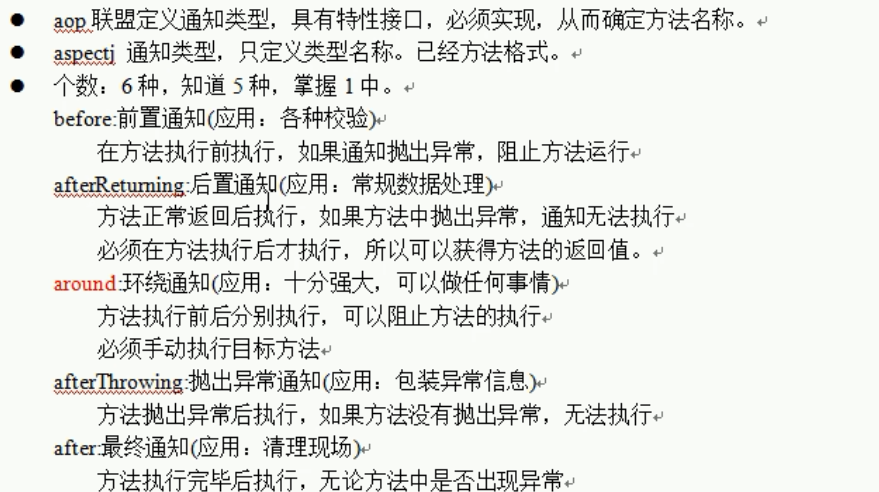
**（..） 表示参数任意**

**throws：省略一般不写**

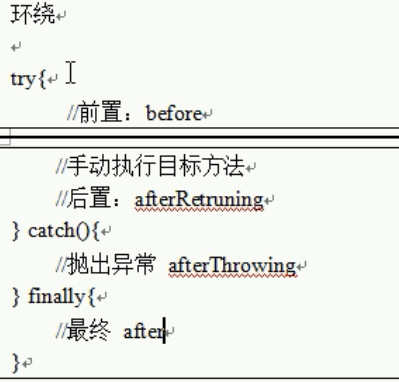
**3）例子（掌握）**

**\* execution（\* cn.lynu.xy.\*.service..\*.\*（..）） 表示执行xy包下的任意包的service包及 service包下的所有子包中的任意类，其中返回值和参数都任意**

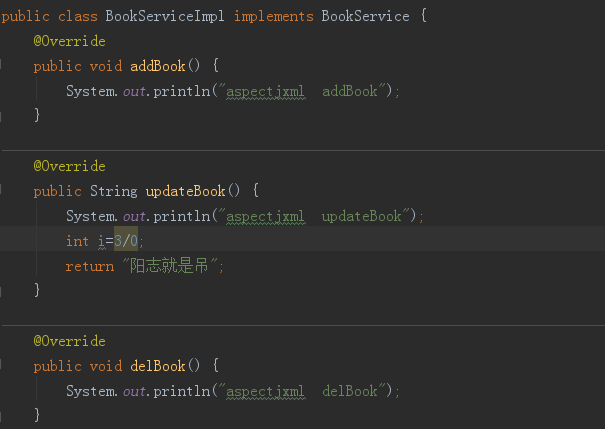
1. AspectJ的通知类型



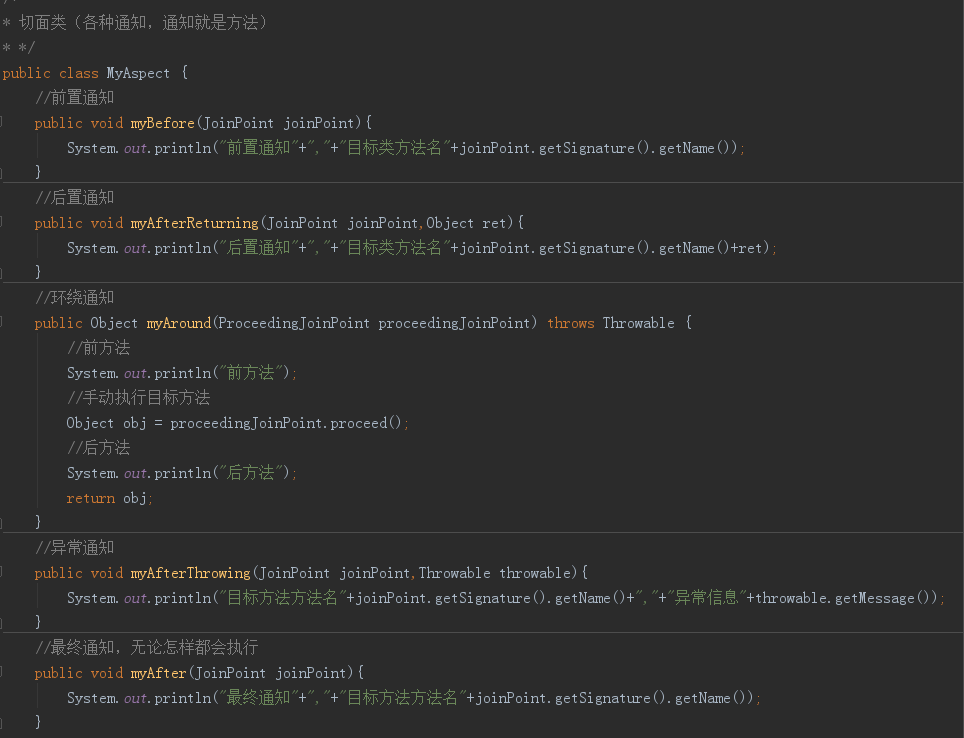
举例:



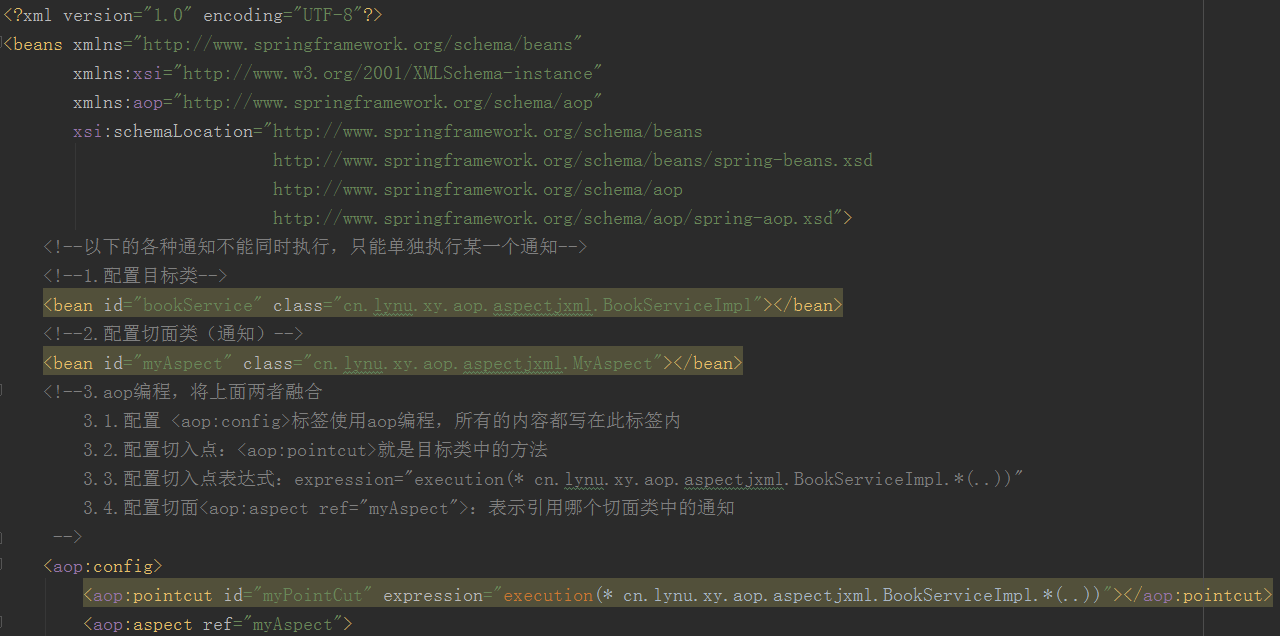
1. **ASpectJ基于xml开发**
2. 编写目标类（接口和实现类，接口省略）

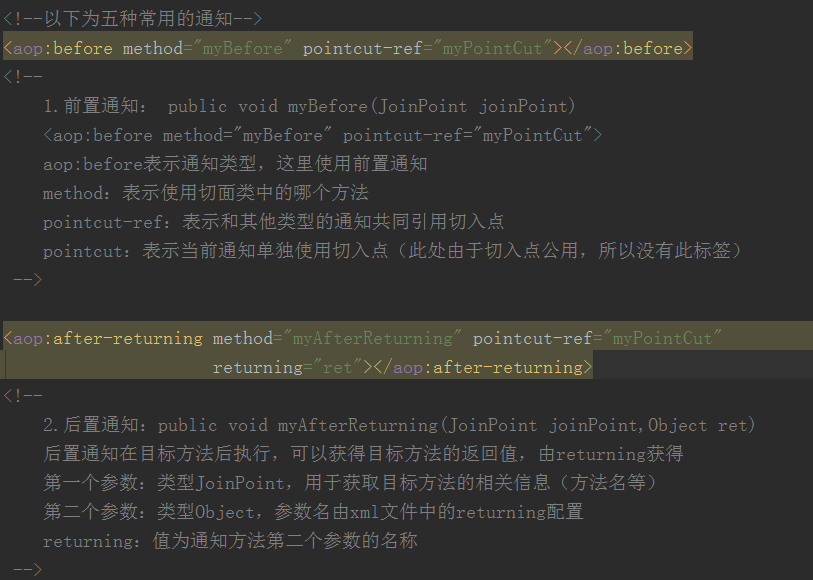


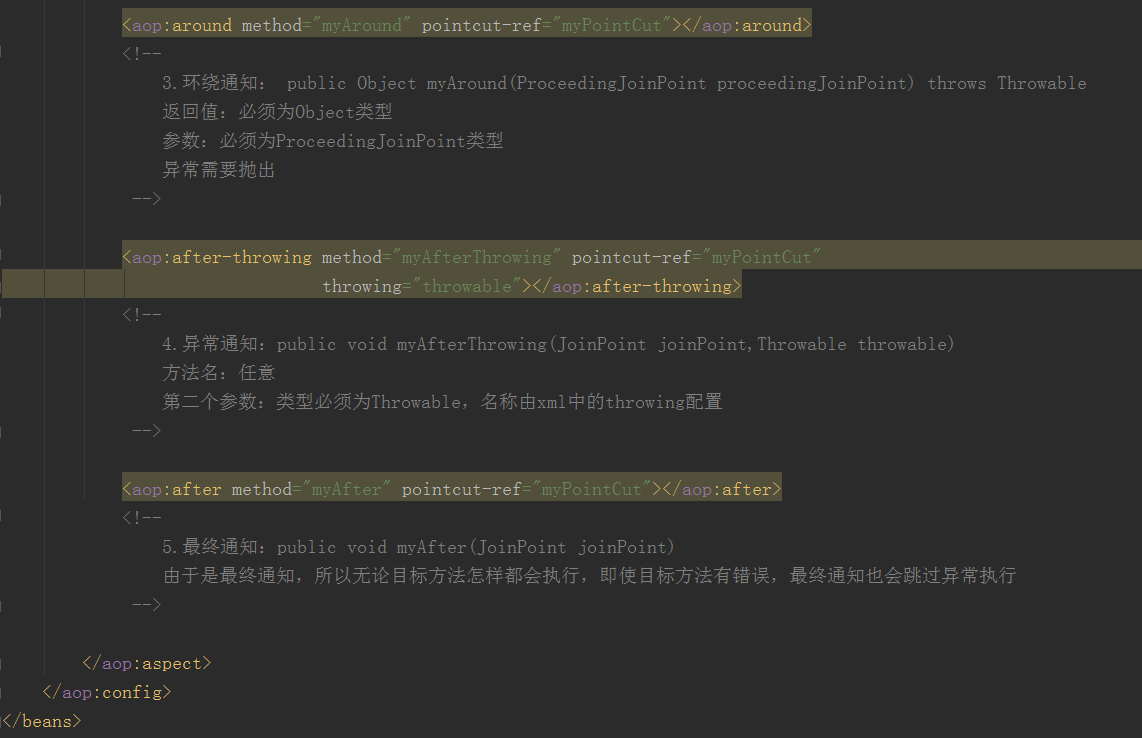
1. 编写切面类



1. 编写配置文件（需要导入aop命名空间）



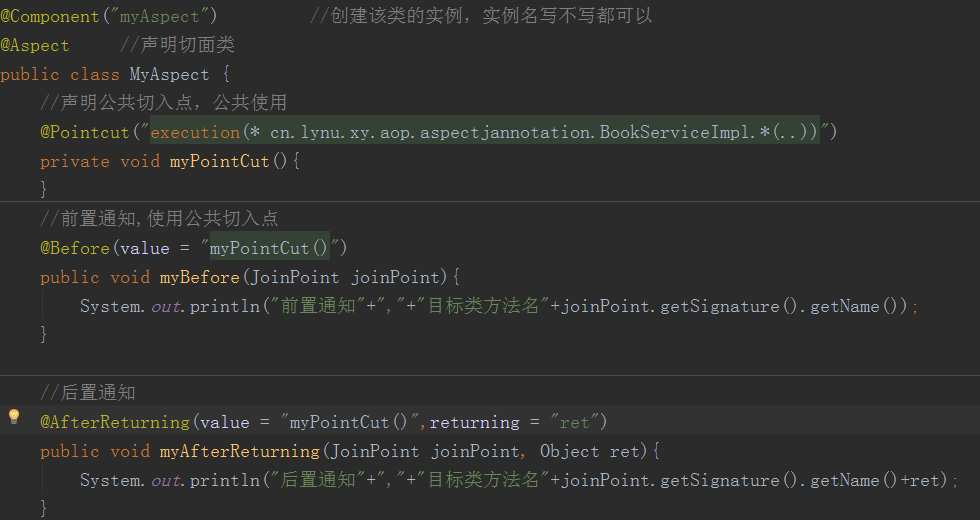




1. **ASpectJ基于注解开发**
2. 编写目标类（接口和实现类，接口省略）



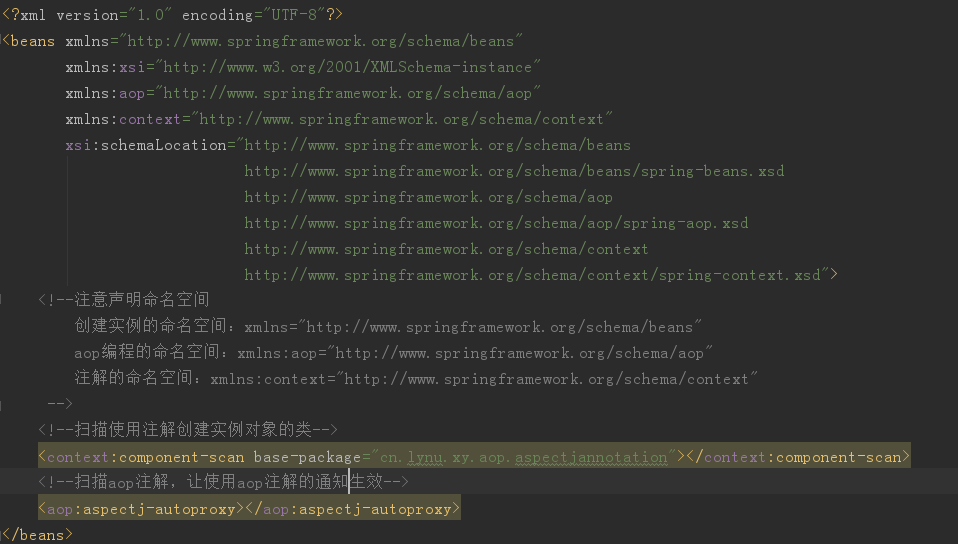
1. 编写切面类



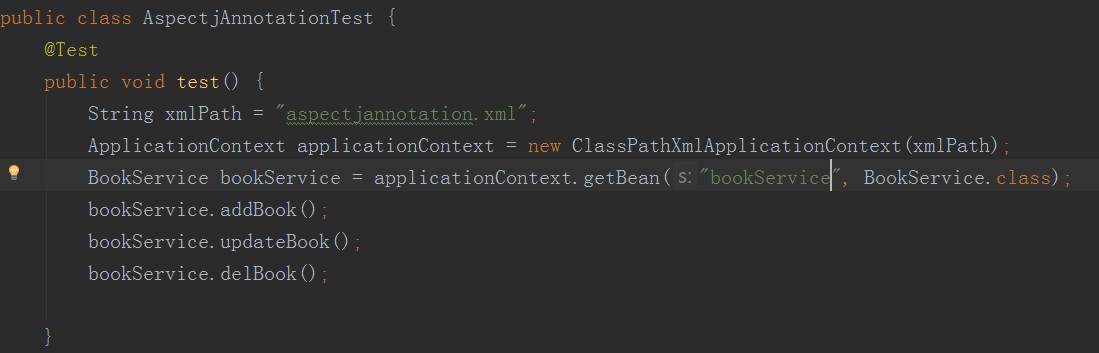
1. 配置切面类，使用注解



1. 编写配置文件，扫描注解



1. 测试



1. aop注解总结

