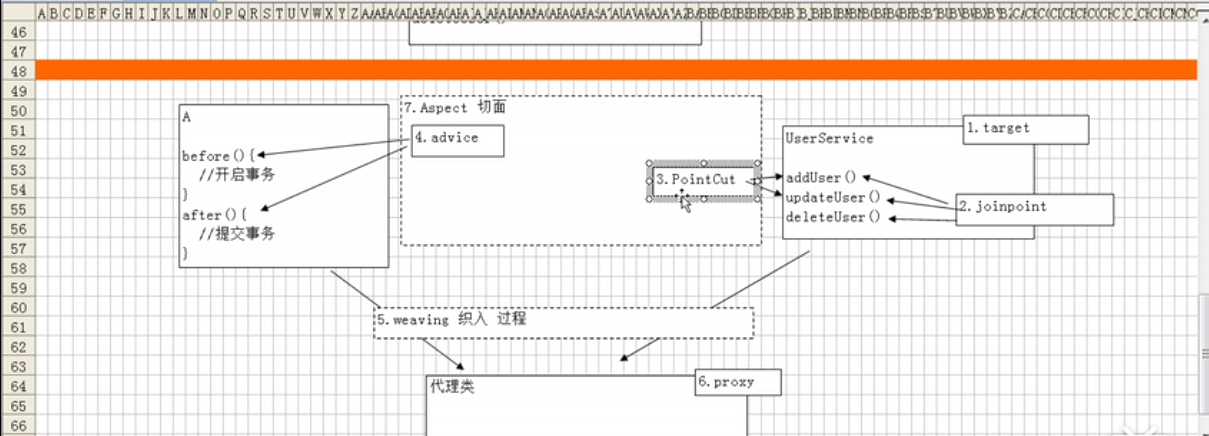
springAop

1. target需要被代理的类 （userservice）
2. Jointpoint（连接点）：指可能被拦截/增强的方法 （uservice的所有方法）
3. Pointcut（切入点）：已经被增强的连接点（已经被使用/增强的方法）
4. advice（通知/增强）：（连接点需要被增强），通过为被增强的代码。列如after，before
5. Weaving（织入）：吧增强的advice应用到目标对象tartget***来创建新的代理对象（proxy）的过程***
6. Proxy（代理）
7. Aspect（切面），切入点与通知的结合



JDK 动态代理

1. 必须要有接口
2. 目标类：接口+实现类
3. 切面类：用于存放通知
4. 工厂类：生成对象（代理对象）
5. 代理类：结合目标类与切面类

CGLIB的字节码增强（框架）

运行时创建目标类的子类，来对目标类进行增强

spring通知类型

五类通知

前置通知：可以阻止目标方法的执行（前置通知抛出异常，就会不执行目标方法）

后置通知

环绕通知：必须手动执行目标方法

抛出异常通知

Spring编写代理：半自动

Spring创建代理对象，从容器中手动获取代理对象

Spring编写代理：全自动

<!--配置  
 ***<aop:pointcut 切入点：其实就是获取要指定的目标方法，用作被增强的方法,在这里代表insert↓与delete方法  
 <aop:advisor 特殊的切面：配置通知与切入点，实现对切入点方法的增强***

proxy-target-class="true 声明式CGLIB  
 -->  
<aop:config proxy-target-class="true">  
 <aop:pointcut id="spring\_autoproxy" expression="execution(\* com.ibm.springautoproxy.impl.SpringAutoServiceImpl.\*(..))"></aop:pointcut>  
 <aop:advisor advice-ref="AspectOId" pointcut-ref="spring\_autoproxy"></aop:advisor>  
</aop:config>

切入点表达式： 第一个\*表示配置所有返回值（无返回值也可以），第二个\*表示匹配所有方法，最后表达式（..）表示配置所有参数（无参数也行）

expression="execution(\* com.ibm.springautoproxy.impl.SpringAutoServiceImpl.\*(..))"

采用aspectj方式：

1. execution（）用于描述方法

语法：execution(修饰符 返回值 包 类 方法（参数） throws 异常)

返回值与方法不能省略（其他的可省略），可用\*来表示任意返回值与方法

Execution（\* com.ibm.crm.\*.service..\*.\*()）

返回值为任意，.\* .中间的\*表示crm包下面任意包下面的service包

..\*表示Service其子包下面所有的实现类

.\* 表示类下面的所有方法

1. 五种通知类型

前置：before

后置：afterReturning

最终通知：after-----跑出异常需要执行的（相当于try catch中finally里面需要执行的语句）

1. XML
2. 注解