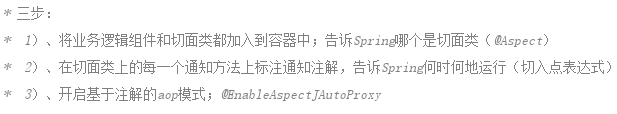
# Aop和动态代理

## 动态代理



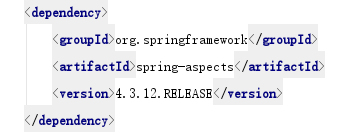
## 三部流程



# Demo流程

## Pom导入aop模块

*1、导入aop模块；Spring AOP：(spring-aspects)*



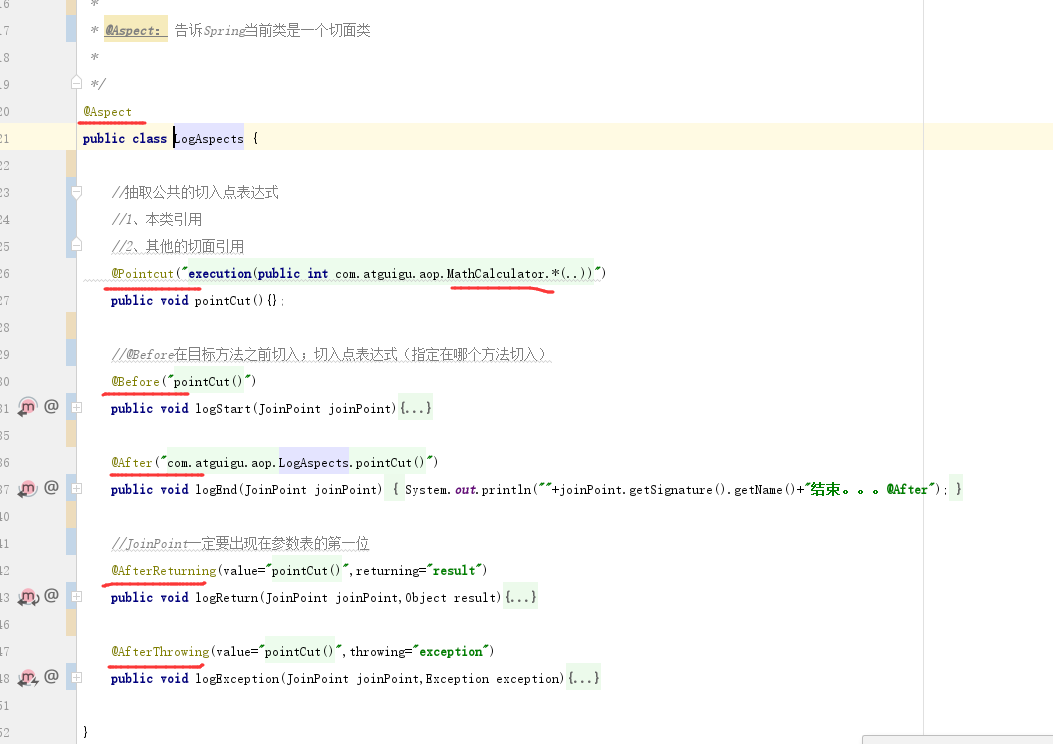
## 业务逻辑类（MathCalculator）

*2、定义一个业务逻辑类（MathCalculator）；在业务逻辑运行的时候将日志进行打印（方法之前、方法运行结束、方法出现异常，xxx）*



## 日志切面类（LogAspects）

*3、定义一个日志切面类（LogAspects）：切面类里面的方法需要动态感知MathCalculator.div运行到哪里然后执行；  
 通知方法：  
 前置通知(@Before)：logStart：在目标方法(div)运行之前运行  
 后置通知(@After)：logEnd：在目标方法(div)运行结束之后运行（无论方法正常结束还是异常结束）  
 返回通知(@AfterReturning)：logReturn：在目标方法(div)正常返回之后运行  
 异常通知(@AfterThrowing)：logException：在目标方法(div)出现异常以后运行  
 环绕通知(@Around)：动态代理，手动推进目标方法运行（joinPoint.procced()）*



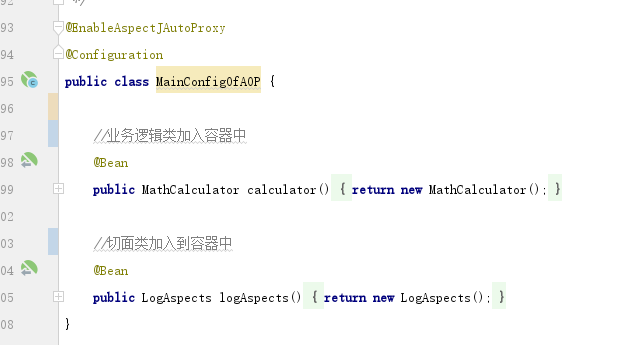
## 切面类的目标方法

1. *给切面类的目标方法标注何时何地运行（通知注解）；*



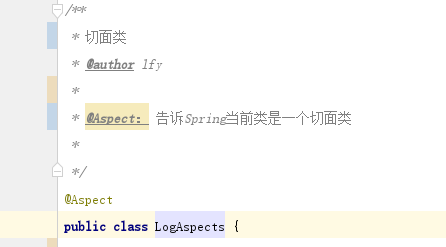
## 切面类和业务逻辑类（目标方法所在类）都加入到容器

*5、将切面类和业务逻辑类（目标方法所在类）都加入到容器中;*



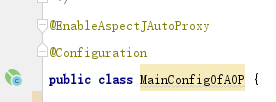
## 切面类(给切面类上加一个注解：@Aspect

*6、必须告诉Spring哪个类是切面类(给切面类上加一个注解：@Aspect)*



## @EnableAspectJAutoProxy 【开启基于注解的aop模式】

*[7]、给配置类中加 @EnableAspectJAutoProxy 【开启基于注解的aop模式】  
 在Spring中很多的 @EnableXXX;*



# Demo

## 环绕通知



*/\*\*  
 \* deleteByApiId 同步redis  
 \* 1.删除redis 里面所有授权表的信息， 2.删除redis 里统计访问次数信息 3.删除redis 里统计访问频率的信息  
 \*/*@Around(**"execution(\* com.kun.openapi.entity.service.KunApiService.deleteKunApi(..))"**)  
**public** Object deleteByApiIdKunApi(ProceedingJoinPoint pjp) {  
 Object resValue = **null**;  
 **try** {  
 Object[] agrs = pjp.getArgs();  
  
 *// 前置通知* String apiId = (String) agrs[0];  
 RedisUseUtils.*del*(apiId);  
  
 *// 执行被拦截的方法* resValue = pjp.proceed(agrs);  
 *// 后置通知* } **catch** (Throwable e) {  
 *// 异常通知* } **finally** {  
 *// 最终通知* }  
 *// 返回通知* **return** resValue;  
}