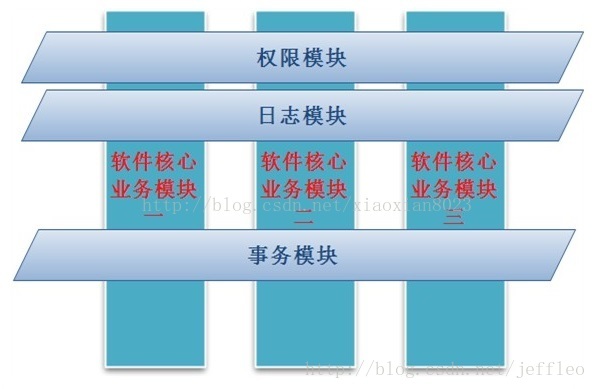
# 动态代理

1. 
2. 切面，就是通用的业务逻辑代码，若日志模块，权限，事务模块
3. 通知，前置，后置，最终，环绕，异常五类
4. 连接点(joinpoint)：就是被拦截的方法
5. 切入点（pointcut）：被拦截的方法，连接点被拦截后变成切入点
6. 目标对象： 指要织入的对象模块，
7. 织入：通过切入点切入，将切面应用到目标对象并导致代理对象创建的过程
8. AOP代理：aop框架创建的对象，包括通知。AOP代理可以使JDK代理或是CGLIB代理
9. 使用AOP的步骤
10. 定义具体的业务逻辑模块
11. 定义切面，实现通知逻辑

aop创建代理后会返回一个连接点， 然后在通知中可以通过该连接点实现我们的切面逻辑

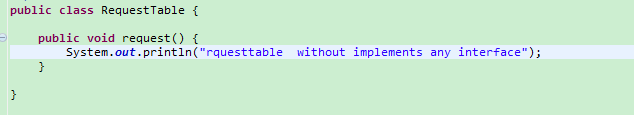
1. 实现切面逻辑

# CGLIB动态字节码生成

CGLIB的原理就是：将目标对象也就是业务逻辑类进行继承扩展，为其生成子类，而子类中可以覆写来扩展父类的行为，只要将横切逻辑放到子类中，然后让系统使用扩展后的目标对象的子类，就可以达到与代理模式相同的效果

例：

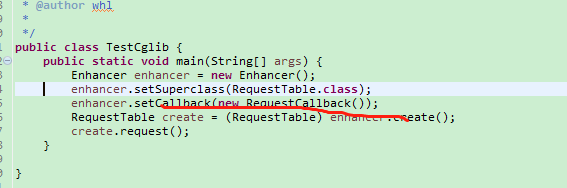
1. 定义业务逻辑类



1. 定义切面，就是对父类进行扩展成子类



1. 通过Enhancer为目标对象动态生成一个子类，将RequestCallback的横切逻辑附加到该子类中

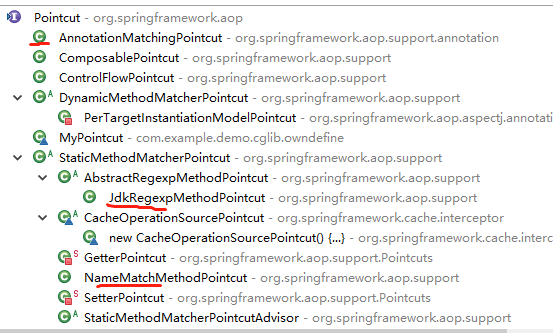


# 3.AOP的底层实现原理

1.Pointcut底层是一个Pointcut接口，里面有对象匹配方法getClassFilter和方法匹配方法getMethodMatcher

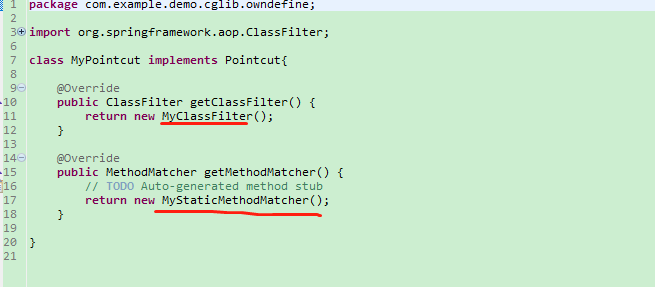
2.MethodMatcher接口中三个方法，isRuntime()方法返回值决定是否匹配是否校验参数

3.Pointcut分为NameMatchMethodPointcut， JdkRegexpMethodPointcut

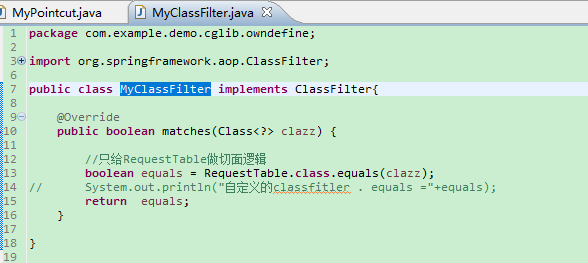


# 自定义的Pointcut , ClassFilter, MethodMatcher

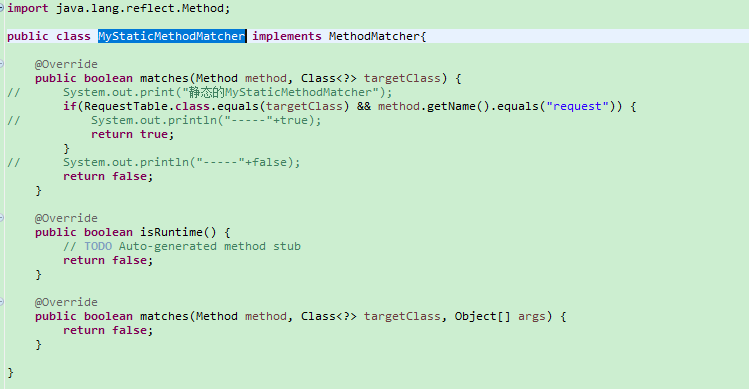
1. 自定义的pointcut



1. 自定义的ClassFilter



1. 自定义的MethodMatcher



1. 测试自定义

