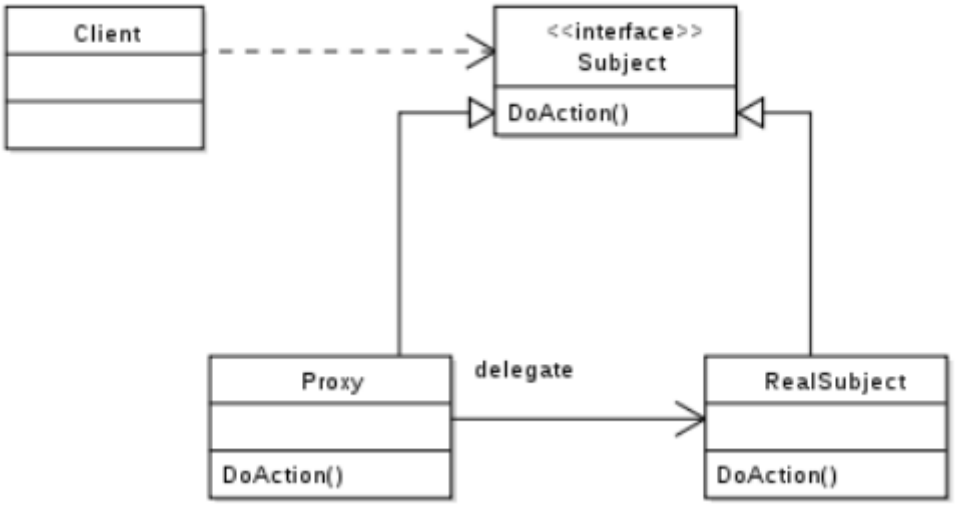
代理机制

笔记本:	java基础		
创建时间:	2019/5/4 15:30	**更新时间:**	2019/5/4 16:20
作者:	ANNER		
URL:	https://segmentfault.com/a/1190000011291179		

代理模式

代理模式是一种设计模式，提供了对目标对象额外的访问方式，即通过代理对象访问目标对象，这样可以在不修改原目标对象的前提下，提供额外的功能操作，扩展目标对象的功能。



静态代理

这种代理方式需要代理对象和目标对象实现一样的接口。
优点：可以在不修改目标对象的前提下扩展目标对象的功能

缺点：

- 冗余。由于代理对象要实现与目标对象一致的接口，会产生过多的代理类。
- 不易维护。一旦接口增加方法，目标对象与代理对象都要进行修改。

动态代理

动态代理利用了JDK API，动态地在内存中构建代理对象，从而实现对目标对象的代理功能。动态代理又被称为JDK代理或接口代理。

静态代理与动态代理的区别主要在：

- 静态代理在编译时就已经实现，编译完成后代理类是一个实际的class文件
- 动态代理是在运行时动态生成的，即编译完成后没有实际的class文件，而是在运行时动态生成类字节码，并加载到JVM中

特点：**动态代理对象不需要实现接口，但是要求目标对象必须实现接口，否则不能使用动态代理。**

cglib代理

cglib (Code Generation Library)是一个第三方代码生成类库，运行时在内存中动态生成一个子类对象从而实现对目标对象功能的扩展。

cglib特点:

- JDK的动态代理有一个限制，就是使用动态代理的对象必须实现一个或多个接口。如果想代理没有实现接口的类，就可以使用CGLIB实现。
- CGLIB是一个强大的高性能的代码生成包，它可以在运行期扩展Java类与实现Java接口。它广泛的被许多AOP的框架使用，例如Spring AOP和dynaop，为他们提供方法的interception（拦截）。
- CGLIB包的底层是通过使用一个小而快的字节码处理框架ASM，来转换字节码并生成新的类。不鼓励直接使用ASM，因为它需要你对JVM内部结构包括class文件的格式和指令集都很熟悉。

cglib与动态代理**最大的区别**就是

- 使用动态代理的对象必须实现一个或多个接口使用
- cglib代理的对象则无需实现接口，达到代理类无侵入

maven依赖:

```
<dependency>
    <groupId>cglib</groupId>
    <artifactId>cglib</artifactId>
    <version>3.2.5</version>
 </dependency>
```