# 概述

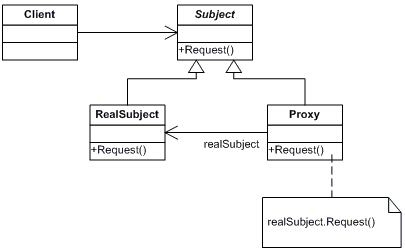
## 定义

代理模式指给目标对象设置一个代理对象，通过代理对象来控制对目标对象的访问。

## 分类

* 静态代理
* 动态代理

## UML



# 知识点

## 静态代理

静态代理是指代理类Proxy在编译期就已经实现好的。

自我理解：

* Proxy类中有一个Subject的对象Target
* Client使用Proxy之前，需要创建一个RealSubjec类型的目标对象Target，并传递给Proxy类中Target；
* Client通过Proxy提供的接口，间接访问Target的方法

参考Demo项目中的类Subject、Proxy、RealSubject，ProxyClient；

静态代理中，每一个 RealSubject的代理类Proxy在编译期就已经确定了。

## 动态代理

动态代理是指运行期动态的生成代理类。

和静态代理一样，需要定义Subject接口，RealSubject目标类（委托类），不同的是动态代理需要定义一个代理类的句柄Handler，继承自InvocationHandler接口，并实现接口中的invoke方法，Handler代理类句柄中需要包含一个Subject对象。

动态代理客户端使用步骤：

* 创建RealSubject对象（委托对象）；
* 创建代理句柄ProxyHandler对象，同时将RealSubject对象作为其构造参数；
* 通过java.lang.reflect.Proxy类中newProxyInstance方法创建一个动态代理类的对象，该方法需要的参数包括RealSubject对象的类加载器，RealSubject的接口集合，以及代理句柄

参考Demo项目中的DynamicProxyClient类的执行过程，同时查看生成的com.sun.proxy包下生成的class文件。动态生成的代理类具有如下特点：

* 继承自java.lang.reflect.Proxy类
* 实现Subject接口

为了理清动态代理是如何生成代理类的，首先需要了解java.lang.reflect.Proxy类

### Proxy

此处的Proxy类是java.lang.reflect.Proxy类，是所有动态代理类的父类。

* 包含一个InvocationHandler对象h；
* getProxyClass方法生成动态代理类；
* newProxyInstance方法生成动态代理类的一个对象dynamicProxy；
* 通过动态代理对象dynamicProxy访问Subject中的接口时，实际上是通过h对象调用InvocationHandler中的invoke方法，方法的参数是目标类对象realSubject，接口名，接口参数；

动态代理的核心就是：通过反射调用InvocationHandler中的invoke方法。

# 参考

<https://www.jianshu.com/p/6f6bb2f0ece9>

http://blog.jobbole.com/104433/