http://blog.csdn.net/janice0529/article/details/42583481

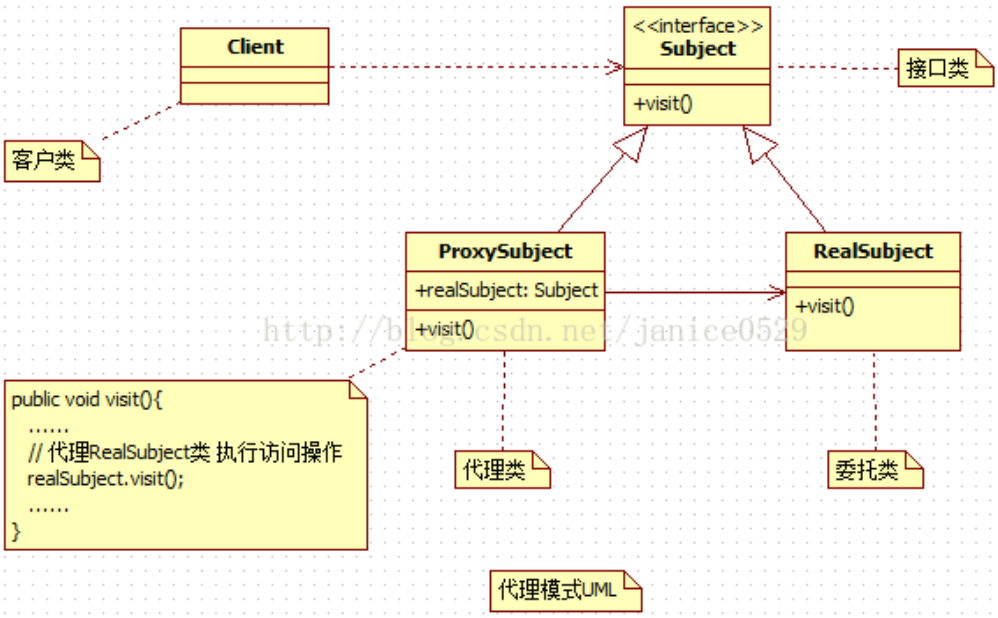
一、概述

给某一个对象提供一个代理，并由代理对象来完成对原对象的访问。代理模式是一种对象结构型模式。

二、适用场景

当无法直接访问某个对象或访问某个对象存在困难时可以通过一个代理对象来间接访问，为了保证客户端使用的透明性，委托对象与代理对象需要实现相同的接口。

三、UML类图



四、参与者

1、接口类：Subject

它声明了真实访问者和代理访问者的共同接口,客户端通常需要针对接口角色进行编程。

2、代理类：ProxySubject

包含了对真实（委托）对象(RealSubject)的引用，在实现的接口方法中调用引用对象的接口方法执行，从而达到代理的作用。看似是代理对象(ProxySubject)在操作，但其实真正的操作者是委托对象（RealSubject）。

3、委托类/真实访问类：RealSubject

它定义了代理角色所代表的真实对象，在真实角色中实现了真实的业务操作，客户端可以通过代理角色间接调用真实角色中定义的操作。

1、接口类：Subject.Java

```
1. /**
2.  * 接口类
3.  * @author lvzb.software@qq.com
4.  *
5.  */
6. public interface Subject {
7.
8.     public void visit();
9. }
```

2、接口实现类，真实访问对象/委托对象：RealSubject.java

```
1. /** 接口实现类，真实访问对象/委托对象 */
2. public class RealSubject implements Subject {
3.
4.     @Override
5.     public void visit() {
6.         System.out.println("I am 'RealSubject',I am the execution
method");
7.     }
8. }
```

3、接口实现类，代理对象：ProxySubject.java

```
1. /** 接口实现类，代理对象 **/
2. public class ProxySubject implements Subject {
3.     // 维持对真实委托对象的引用，该对象才是真正的执行者
```

```
4.      private Subject realSubject;
5.      public ProxySubject(Subject subject){
6.          this.realSubject = subject;
7.      }
8.
9.      @Override
10.      public void visit() {
11.          // 真实委托对象 通过 代理对象的引用 间接的实现了对目标对象的访问执行
12.          System.out.println("before real visit ");
13.          realSubject.visit();
14.          System.out.println("after real visit ");
15.      }
16. }
```

4、客户类 Client.java

```
1. /** 客户类 **/
2. public class Client {
3.      public static void main(String[] args) {
4.          Subject proxySubject = new ProxySubject(new RealSubject());
5.          proxySubject.visit();
6.      }
7. }
```

六、其他/扩展

按照代理类的创建时期，代理类可以分为两种:

1、静态代理：由程序员创建或特定工具自动生成源代码，再对其编译。在程序运行前，代理类的.class文件就已经存在了。（上面用例介绍的就是 静态代理技术）

静态代理的优劣分析:

优点：客户端面向接口编程，符合开闭原则，使系统具有好的灵活性和扩展性。

缺点：从上面代码中我们可以发现 每一种代理类都是实现了特定的接口，及每一种代理类只能为特定接口下的实现类做代理。如

果是不同接口下的其他实现类，则需要重新定义新接口下的代理类。

那么是否可以通过一个代理类完成不同接口下实现类的代理操作呢？那么此时就必须使用动态代理来完成。

2、动态代理：在程序运行时，运用JAVA反射机制动态创建代理实例。