# 代理模式

## 问题引入

如果你想租房，你可能会找租房中介；如果你想找对象，你可能会找婚姻中介……，生活中有很多中介的例子，人们选择中介无一例外的是为了省事，找租房中介是为了避免自己找房的麻烦，找婚姻中介是为了避免自己找对象的麻烦……。

在软件系统中同样有中介的存在，只不过在软件系统中我们称之为“代理”。假如直接访问某个对象很麻烦，或者某个对象不想被直接外部直接访问，可以通过添加代理来解决。

## 定义

代理模式的定义：由于某些原因需要给某对象提供一个代理以控制外部对该对象的访问。这时，访问对象不适合或者不能直接引用目标对象，代理对象将作为访问对象和目标对象之间的中介。

## 特点

### 优点

1. 代理模式在目标对象和客户端间设置一个中介（代理对象），能够起到保护目标对象的作用。
2. 代理对象能够拓展目标对象的功能。比如，突然需要记录某个类的调用日志，可以为这个类设置代理类，在代理类中实现日志功能。
3. 目标对象与客户端分离，一定程度上降低了系统的耦合程度。

### 缺点

1. 对目标对象的访问需要经过代理，造成请求处理速度变慢。
2. 系统复杂程度增加，维护难度增加。

## 结构

1. 抽象主题类（Subject）：通过接口或抽象类声明真实主题类和代理类应该具有什么方法。
2. 真实主题类（Real Subject）：实现抽象主题类中的方法，是代理类的代理目标，是客户端的访问目标。
3. 代理类（Proxy）：实现抽象主题类中的方法，包含真实主题对象的引用，可以访问、控制、扩展真实主题对象的功能。

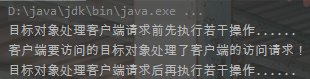
## 实例

实例结构图：

（需要注意一点：被代理对象（realSubject）是代理类（ProxySubject）自己创建的。这是代理模式和许多类似模式的最大区别。）



运行结果：



## 应用场景

* 远程代理，这种方式通常是为了隐藏目标对象存在于不同地址空间的事实，方便客户端访问。例如，用户申请某些网盘空间时，会在用户的文件系统中建立一个虚拟的硬盘，用户访问虚拟硬盘时实际访问的是网盘空间。
* 虚拟代理，这种方式通常用于要创建的目标对象开销很大时。例如，下载一幅很大的图像需要很长时间，因某种计算比较复杂而短时间无法完成，这时可以先用小比例的虚拟代理替换真实的对象，消除用户对服务器慢的感觉。
* 安全代理，这种方式通常用于控制不同种类客户对真实对象的访问权限。
* 智能指引，主要用于调用目标对象时，代理附加一些额外的处理功能。例如，增加计算真实对象的引用次数的功能，这样当该对象没有被引用时，就可以自动释放它。
* 延迟加载，指为了提高系统的性能，延迟对目标的加载。例如，[Hibernate](http://c.biancheng.net/hibernate/) 中就存在属性的延迟加载和关联表的延时加载。

## 拓展

在前面介绍的代理模式中，代理类中包含了对真实主题的引用，这种方式存在两个缺点。

1. 真实主题与代理主题一一对应，增加真实主题也要增加代理。
2. 设计代理以前真实主题必须事先存在，不太灵活。

动态代理模式解决了以上问题，动态代理模式的使用实例有Spring AOP,其结构图如下所示：



由于动态代理模式比较复杂，所以另外开一个文档记录动态代理的内容。