# 代理模式

在代理模式（Proxy Pattern）中，一个类代表另一个类的功能。这种类型的设计模式属于结构型模式。

在代理模式中，我们创建具有现有对象的对象，以便向外界提供功能接口。

## 介绍

**意图：**为其他对象提供一种代理以控制对这个对象的访问。

**主要解决：**在直接访问对象时带来的问题，比如说：要访问的对象在远程的机器上。在面向对象系统中，有些对象由于某些原因（比如对象创建开销很大，或者某些操作需要安全控制，或者需要进程外的访问），直接访问会给使用者或者系统结构带来很多麻烦，我们可以在访问此对象时加上一个对此对象的访问层。

**何时使用：**想在访问一个类时做一些控制。

**如何解决：**增加中间层。

**关键代码：**实现与被代理类组合。

**应用实例：** 1、Windows 里面的快捷方式。 2、猪八戒去找高翠兰结果是孙悟空变的，可以这样理解：把高翠兰的外貌抽象出来，高翠兰本人和孙悟空都实现了这个接口，猪八戒访问高翠兰的时候看不出来这个是孙悟空，所以说孙悟空是高翠兰代理类。 3、买火车票不一定在火车站买，也可以去代售点。 4、一张支票或银行存单是账户中资金的代理。支票在市场交易中用来代替现金，并提供对签发人账号上资金的控制。 5、spring aop。

**优点：** 1、职责清晰。 2、高扩展性。 3、智能化。

**缺点：** 1、由于在客户端和真实主题之间增加了代理对象，因此有些类型的代理模式可能会造成请求的处理速度变慢。 2、实现代理模式需要额外的工作，有些代理模式的实现非常复杂。

**使用场景：**按职责来划分，通常有以下使用场景： 1、远程代理。 2、虚拟代理。 3、Copy-on-Write 代理。 4、保护（Protect or Access）代理。 5、Cache代理。 6、防火墙（Firewall）代理。 7、同步化（Synchronization）代理。 8、智能引用（Smart Reference）代理。

**注意事项：** 1、和适配器模式的区别：适配器模式主要改变所考虑对象的接口，而代理模式不能改变所代理类的接口。 2、和装饰器模式的区别：装饰器模式为了增强功能，而代理模式是为了加以控制。

**总结**

 代理模式（Proxy）根据种类不同，效果也不尽相同：

* 远程（Remote）代理：为一个位于不同的地址空间的对象提供一个局域代表对象。这个不同的地址空间可以是在本机器中，也可是在另一台机器中。远程代理又叫做大使（Ambassador）。好处是系统可以将网络的细节隐藏起来，使得客户端不必考虑网络的存在。客户完全可以认为被代理的对象是局域的而不是远程的，而代理对象承担了大部份的网络通讯工作。由于客户可能没有意识到会启动一个耗费时间的远程调用，因此客户没有必要的思想准备。
* 虚拟（Virtual）代理（图片延迟加载的例子）：根据需要创建一个资源消耗较大的对象，使得此对象只在需要时才会被真正创建。使用虚拟代理模式的好处就是代理对象可以在必要的时候才将被代理的对象加载；代理可以对加载的过程加以必要的优化。当一个模块的加载十分耗费资源的情况下，虚拟代理的好处就非常明显。
* Copy-on-Write代理：虚拟代理的一种。把复制（克隆）拖延到只有在客户端需要时，才真正采取行动。
* 智能引用（Smart Reference）代理：当一个对象被引用时，提供一些额外的操作，比如将对此对象调用的次数记录下来等。

     代理模式（Proxy）VS [装饰者（Decorator）](http://www.cnblogs.com/rush/archive/2011/05/08/Decorator_DesignPattern_NETFramework.html" \t "http://www.cnblogs.com/rush/archive/2011/06/19/_blank)

     意图：它们都提供间接访问对象层，都保存被调用对象的引用。

     代理模式（Proxy）：为另一个对象提供一个替身或占位符以控制对这个对象的访问，简而言之就是用一个对象来代表另一个对象。

    装饰者（Decorator）：动态地给一个对象添加一些额外的职责，就增加功能来说，Decorator模式比生成子类更为灵活，它避免了类爆炸问题，像装饰者（Decorator），代理模式（Proxy）组成一个对象并提供相同的接口，但代理模式并不关心对象动态职能的增减。

    在代理模式（Proxy）中Subject定义了主要的功能，而且Proxy根据Subject提供功能控制对象的访问权限。在装饰者（Decorator）中Component只是提供了其中的一些功能，需要通过装饰链动态给对象增加职能。