深入浅出基于Java的代理设计模式 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/144/2021_2022__E6_B7_B1_E5_85_A5_E6_B5_85_E5_c104_144659.htm 一、引子 我们去科技市场为自己的机器添加点奢侈的配件，很多DIYer都喜欢去找代理商，因为在代理商那里拿到的东西不仅质量有保证，而且价格和售后服务上都会好很多。客户通过代理商得到了自己想要的东西，而且还享受到了代理商额外的服务；而生产厂商通过代理商将自己的产品推广出去，而且可以将一些销售服务的任务交给代理商来完成（当然代理商要和厂商来共同分担风险，分配利润），这样自己就可以花更多的心思在产品的设计和生产上了。 在美国，任何企业的产品要想拿到市场上去卖就必须经过代理商这一个环节，否则就是非法的。看来代理商在商业运作中起着很关键的作用。 不小心把话题扯远了，回过头来，那么在我们的面向对象的程序设计中，会不会有代理商这样的角色呢？来看这篇文章的人肯定不会说：没有！ 那么就跟着这篇文章来看看代理模式的奇妙吧。 二、定义和分类 代理模式在设计模式中的定义就是：为其他对象提供一种代理以控制对这个对象的访问。说白了就是，在一些情况下客户不想或者不能直接引用一个对象，而代理对象可以在客户和目标对象之间起到中介作用，去掉客户不能看到的内容和服务或者增添客户需要的额外服务。 那么什么时候要使用代理模式呢？在对已有的方法进行使用的时候出现需要对原有方法进行改进或者修改，这时候有两种改进选择：修改原有方法来适应现在的使用方式，或者使用一个“ 第三者 ”方法来调用原有的方法并且对方法产生的结

果进行一定的控制。第一种方法是明显违背了“ 对扩展开放、对修改关闭 ”（开闭原则），而且在原来方法中作修改可能使得原来类的功能变得模糊和多元化（就像现在企业多元化一样），而使用第二种方式可以将功能划分的更加清晰，有助于后面的维护。所以在一定程度上第二种方式是一个比较好的选择！ 当然，话又说回来了，如果是一个很小的系统，功能也不是很繁杂，那么使用代理模式可能就显得臃肿，不如第一种方式来的快捷。这就像一个三口之家，家务活全由家庭主妇或者一个保姆来完成是比较合理的，根本不需要雇上好几个保姆层层代理:) 根据《Java与模式》书中对代理模式的分类，代理模式分为8种，这里将几种常见的、重要的列举如下： 1. 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。比如：你可以将一个在世界某个角落一台机器通过代理假象成你局域网中的一部分。 2. 虚拟（Virtual）代理：根据需要将一个资源消耗很大或者比较复杂的对象延迟的真正需要时才创建。比如：如果一个很大的图片，需要花费很长时间才能显示出来，那么当这个图片包含在文档中时，使用编辑器或浏览器打开这个文档，这个大图片可能就影响了文档的阅读，这时需要做个图片Proxy来代替真正的图片。 3. 保护（Protect or Access）代理：控制对一个对象的访问权限。比如：在论坛中，不同的身份登陆，拥有的权限是不同的，使用代理模式可以控制权限（当然，使用别的方式也可以实现）。 4. 智能引用（Smart Reference）代理：提供比对目标对象额外的服务。比如：纪录访问的流量（这是个再简单不过的例子），提供一些友情提示等等。 代理模式是一种比较有用的模式，从几个类的“ 小结构 ”

到庞大系统的“ 大结构 ”都可以看到它的影子。 三、结构 代理模式中的“ 代理商 ”要想实现代理任务，就必须和被代理的“ 厂商 ”使用共同的接口（你可以想象为产品）。所以自然而然你会想到在java中使用一个抽象类或者接口（推荐）来实现这个共同的接口。于是代理模式就有三个角色组成了：1. 抽象主题角色：声明了真实主题和代理主题的共同接口。2. 代理主题角色：内部包含对真实主题的引用，并且提供和真实主题角色相同的接口。 3. 真实主题角色：定义真实的对象。 使用类图来表示下三者间的关系如下：当然，图上所示的是代理模式中的一个具体情况。而代理模式可以非常灵活的使用其他方式来实现，这样就与图上所示有很大的区别。 也许，现在你已经对代理模式已经有了一个宏观的认识了，下面我们来看看怎么实际的使用代理模式。 