**介绍**

命令模式(Command)的定义是：用于将一个请求封装成一个对象，从而使你可用不同的请求对客户进行参数化；对请求排队或者记录请求日志，以及执行可撤销的操作。也就是说改模式旨在将函数的调用、请求和操作封装成一个单一的对象，然后对这个对象进行一系列的处理。此外，可以通过调用实现具体函数的对象来解耦命令对象与接收对象。

**正文**

我们来通过车辆购买程序来展示这个模式，首先定义车辆购买的具体操作类：

$(function () {  
  
 var CarManager = {  
  
 // 请求信息  
 requestInfo: function (model, id) {  
 return 'The information for ' + model +  
 ' with ID ' + id + ' is foobar';  
 },  
  
 // 购买汽车  
 buyVehicle: function (model, id) {  
 return 'You have successfully purchased Item '  
 + id + ', a ' + model;  
 },  
  
 // 组织view  
 arrangeViewing: function (model, id) {  
 return 'You have successfully booked a viewing of '  
 + model + ' ( ' + id + ' ) ';  
 }  
 };  
})();

来看一下上述代码，通过调用函数来简单执行manager的命令，然而在一些情况下，我们并不想直接调用对象内部的方法。这样会增加对象与对象间的依赖。现在我们来扩展一下这个CarManager 使其能够接受任何来自包括model和car ID 的CarManager对象的处理请求。根据命令模式的定义，我们希望实现如下这种功能的调用：

CarManager.execute({ commandType: "buyVehicle", operand1: 'Ford Escort', operand2: '453543' });

根据这样的需求，我们可以这样啦实现CarManager.execute方法：

CarManager.execute = function (command) {  
 return CarManager[command.request](command.model, command.carID);  
};

改造以后，调用就简单多了，如下调用都可以实现（当然有些异常细节还是需要再完善一下的）：

CarManager.execute({ request: "arrangeViewing", model: 'Ferrari', carID: '145523' });  
CarManager.execute({ request: "requestInfo", model: 'Ford Mondeo', carID: '543434' });  
CarManager.execute({ request: "requestInfo", model: 'Ford Escort', carID: '543434' });  
CarManager.execute({ request: "buyVehicle", model: 'Ford Escort', carID: '543434' });

**总结**

命令模式比较容易设计一个命令队列，在需求的情况下比较容易将命令计入日志，并且允许接受请求的一方决定是否需要调用，而且可以实现对请求的撤销和重设，而且由于新增的具体类不影响其他的类，所以很容易实现。

但敏捷开发原则告诉我们，不要为代码添加基于猜测的、实际不需要的功能，如果不清楚一个系统是否需要命令模式，一般就不要着急去实现它，事实上，在需求的时通过重构实现这个模式并不困难，只有在真正需求如撤销、恢复操作等功能时，把原来的代码重构为命令模式才有意义。