## 简介

将行为action和命令command进行封装，n个action组合封装成一个命令command，n个command封装成一个invoker。Invoker发出一个方法，执行n个命令，执行n\*n个行为。实现请求发出者与请求执行者的解耦。

## 角色

1. 抽象接收者（receiver）:接受者的抽象接口，
2. 接受者（ConcreteReceiver）：接受者具体角色，执行具体action。
3. 抽象命令角色（command）：具体命令类的抽象就接口
4. 具体命令角色（concreteCommand）：调用接收者的操作
5. 请求者（invoker）：调用命令
6. 客户端（client）：创建一个具体的命令对象，并确定其接收者

## 优缺点

1. 低耦合：命令请求者和命令执行者的耦合低
2. 灵活：灵活地更换接收者，灵活地更改命令，灵活地组合命令，灵活地组合行为
3. 易扩展：容易实现新的命令，不会影响已有的命令

## 使用场景

在下面的情况下应当考虑使用命令模式：

1. 使用命令模式作为"CallBack"在面向对象系统中的替代。"CallBack"讲的便是先将一个函数登记上，然后在以后调用此函数。
2. 需要在不同的时间指定请求、将请求排队。一个命令对象和原先的请求发出者可以有不同的生命期。换言之，原先的请求发出者可能已经不在了，而命令对象本身仍然是活动的。这时命令的接收者可以是在本地，也可以在网络的另外一个地址。命令对象可以在串形化之后传送到另外一台机器上去。
3. 系统需要支持命令的撤消(undo)。命令对象可以把状态存储起来，等到客户端需要撤销命令所产生的效果时，可以调用undo()方法，把命令所产生的效果撤销掉。命令对象还可以提供redo()方法，以供客户端在需要时，再重新实施命令效果。
4. 如果一个系统要将系统中所有的数据更新到日志里，以便在系统崩溃时，可以根据日志里读回所有的数据更新命令，重新调用Execute()方法一条一条执行这些命令，从而恢复系统在崩溃前所做的数据更新。

## 参考

<http://www.cnblogs.com/java-my-life/archive/2012/06/01/2526972.html>