**命令模式：将请求封装成对象，以便使用不同的请求、日志、队列等来参数化其他对象。命令模式也支持撤销操作。**

**请求以命令的形式包裹在对象中**，并传给调用对象。调用对象寻找可以处理该命令的合适的对象，并把该命令传给相应的对象，该对象执行命令。

Odp，类似于controller实现。

<http://blog.csdn.net/jialinqiang/article/details/8752111>

**客户**（非主体客户）、**命令**继承自命令接口（订单），**调度者**（服务员），**接收者**（厨师）

一个请求来，生成一个命令调用类，调用类调用相关接口处理请求。

public interface Command

{

public void execute();

}

//命令

public class ConcreteCommand implements Command

{

// 持有接收者的引用

private Receiver receiver;

public ConcreteCommand(Receiver receiver)

{

this.receiver = receiver;

}

@Override

public void execute()//将命令的执行委托给接收者

{

receiver.doAction();

}

}

//接收者 真正执行逻辑的

public class Receiver

{

public void doAction()

{

System.out.println("doAction");

}

}

//调度（调用者，服务员）者，invoker内部可以实现队列、日志等

public class **Invoker**

{

private Command command;

public Invoker(Command command)

{

this.command = command;

}

public void doInvokerAction()

{

command.execute();

}

}

public class Client

{

public static void main(String[] args)

{

Receiver receiver = new Receiver();

Command command = new ConcreteCommand(receiver);

Invoker invoker = new Invoker(command);

invoker.doInvokerAction();

}

}

客户client：发起请求，方法调用

调度者invoker：调用command

命令command：封装接受者，让相应的接受者执行相应的操作

接受者receiver：实际执行者（做菜的厨师）

Invoker的作用:

：需要做多个命令的组合、命令队列（包括对队列的操作）、增加命令日志、撤销重做 这些操作的时候就提现出Invoker的作用了

作用：客户不需要直接调用执行者，不同的client只要通过同样的标准去调用invoker即可。

**执行者和客户解耦**。Invoker内可以实现一些通用的调用过程，如：日志、队列等。