

命令模式 Lab 文档

17302010049 刘佳兴

1 模式设计

1.1 命令模式

将一个请求封装为一个对象，从而使你可用不同的请求对客户进行参数化，对请求排队或记录请求日志。以及支持可撤销的操作。它一般涉及到 4 类对象：

1. Command 和 ConcreteCommand 表示抽象名接口和具体命令。
2. Invoker 是命令的请求者。对于用户的命令，客户端委托 Invoker 进行请求。
3. Reciver 是命令的接受者。接收者执行与请求相关的操作，真正执行命令的对象。具体实现对请求的业务处理。
4. Client 是客户类。在客户类中需要创建调用者对象，具体命令类对象，在创建具体命令对象时指定对应的接收者。发送者和接收者之间没有直接关系，都通过命令对象来调用。

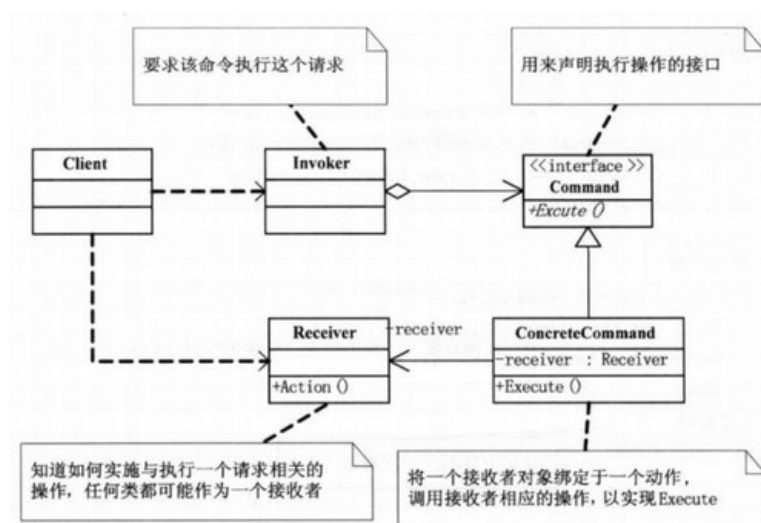


图 1: 命令模式 UML 图.

1.2 代码设计

根据命令模式的设计模式和本应用的特点，我进行了如下的设计：

1. Editor：它属于 Invoker 对象，其中含有方法 `executeCommand(Command):String`。这个方法会请求命令的执行，与抽象接口 `Command` 进行绑定。
2. EditorModel：它属于 Reciver 对象，是本编辑器与文本相关命令的实际执行者。它是一个 model 对象，拥有数据和对数据的操作方法。
3. Command：它属于抽象 `Command` 对象，含有抽象方法 `execute(): String`, `redo(): String`, `undo(): String` 和 `myClone(): String`。在具体的命令类中，`execute` 方法表示执行本命令的逻辑，`myClone` 表示对本命令进行一个拷贝。
4. Client：它属于 Client 对象，会持有 Editor 对象，EditorModel 对象和 `Command` 对象。

1.3 模式的运用

1.3.1 命令执行流程

Client 解析出具体的 `Command` 后，会调用 Editor 对象的 `executeCommand` 方法进行执行。在 `executeCommand` 方法中，Editor 对象只是简单的调用具体 `Command` 对象的 `execute` 方法执行请求。对于每一个具体的命令，它的 `execute` 方法并不会直接操作文本，而是让请求接收者 EditorModel 对象进行处理。这就是命令模式的执行流程。

1.3.2 撤销命令

命令的撤销与执行是一堆相反的操作，它的实现也是让请求接收者 EditorModel 进行处理。只不过对于删除命令，我们需要在本命令中记录下删除的文本内容，以方便进行删除命令的撤销。

1.3.3 宏命令

宏命令虽然含有多个命令，但它本质上依旧是一个普通的命令。它的执行是根据时序，依次执行命令列表中的所有命令。而宏命令的撤销也是调用命令列表所有命令的撤销操作，但它一般是逆序进行调用。在 Java 中，引用的共享是一件比较危险的操作。它可能会在不经意间改变对象的属性。为了实现宏命令中的子命令对象和原命令对象的分离，需要采用深拷贝到方法。这里是通过让每一个具体的类实现 `myClone` 方法实现。