CommandPattern

- 요청을 객체 형태로 캡슐화
- command를 저장하거나 메서드에 전달하거나 다른 객체들처럼 반환 할수 있게 해주는 디자인 패턴

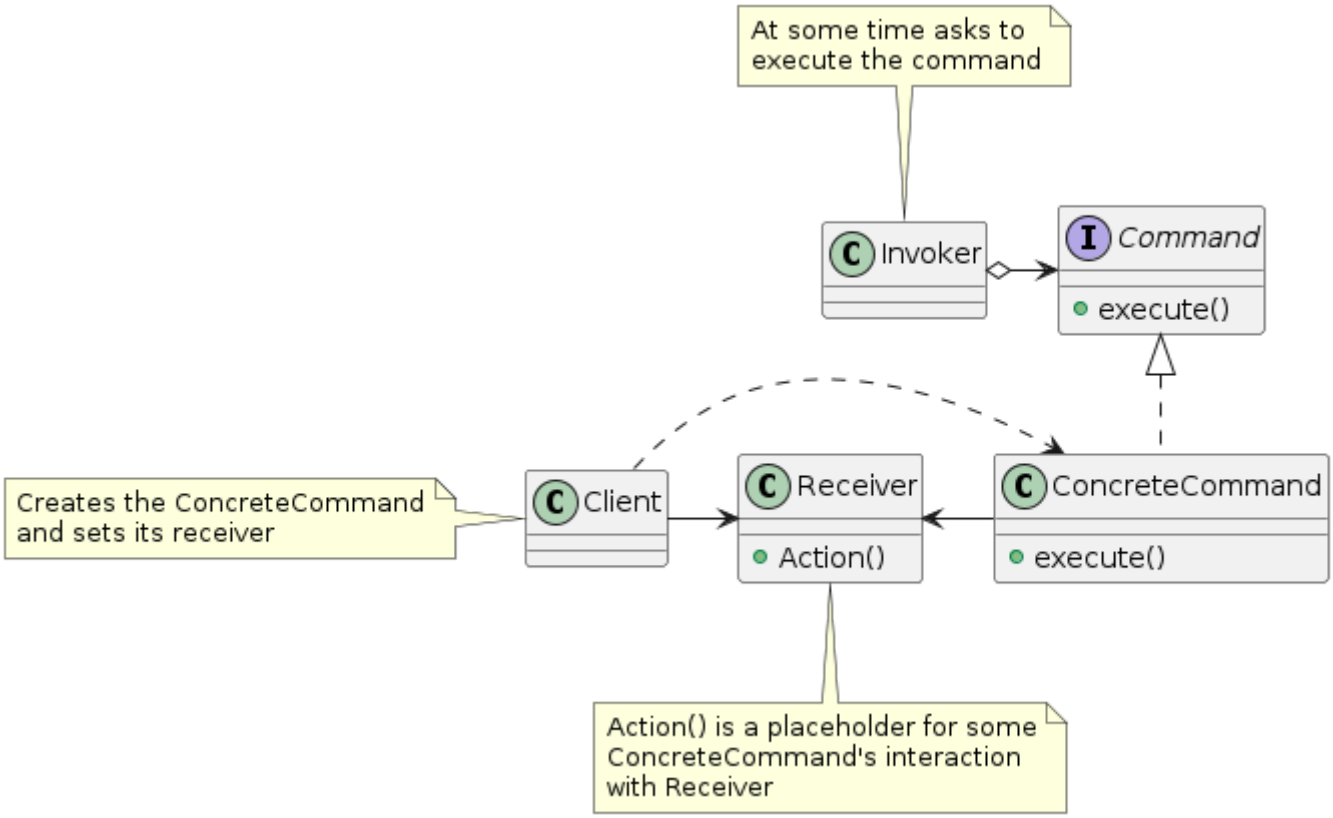
구성요소

- Command : 요청을 캡슐화하는 인터페이스 정의, execute()메서드를 선언
- ConcreteCommand : Command 인터페이스를 구현한 구체적인 클래스. execute()메서드를 구현해 실제 요처을 처리
- Invoker: 요청을 수신하는 객체를 정의합니다. Invoker는 Command 객체를 유지하고, execute() 메서드를 호출하여 요청을 처리합니다.
- Receiver: 실제 요청 처리를 수행하는 객체입니다. ConcreteCommand는 Receiver 객체를 호출하여 실제 요청 처리를 수행합니다.

Command 패턴을 사용하면 다음과 같은 이점이 있습니다.

- 요청 처리 과정을 캡슐화하므로, 요청 처리 과정의 변경이나 확장에 유연하게 대응할 수 있습니다.
- 요청 처리에 대한 로깅, 취소, 다시 실행 등의 기능을 구현하기 쉽습니다.
- 객체 간의 의존성을 줄일 수 있으며, 객체 간의 결합도를 낮출 수 있습니다.

Command 패턴은 GUI 애플리케이션에서는 메뉴나 버튼 등의 요소들과 이벤트 핸들러 등에서 사용됩니다. 또한, 웹 애플리케이션에서도 HTTP 요청과 응답 처리에 Command 패턴을 적용할 수 있습니다.



실제 요청을 처리하는 Servlet들

- 기존 서블릿에서 -> Controller라는 접미사를 사용
- Command interface

기존 코드

- FontServlet 으로 공통 처리 부분 모으고
    - response content-type, character encoding 지정
    - view 처리
    - 예외 처리
    - ...
- 실제 요청 처리 Servlet은
    - 호출 규칙을 추상화하기 위해 Command 패턴 적용
    - 실제 요청을 처리하던 Servlet은 Command interface를 구현하는 Servlet이 아닌 일반 클래스
        - HttpServlet에 대한 의존성 X
        - 재사용성 높아짐

NEXT LEVEL ==> FrontControllerPattern

- 웹 사이트의 모든 요청을 처리하는 컨트롤러 (A controller that handles all requests for a web site)
- 보안, 국제화, 뷰 제공 등의 공통적인 작업을 수행
- cf.) Front Controller vs Page Controller

BUT...

- Command 구현체가 n개
- resolve가 if에 의한 분기처리