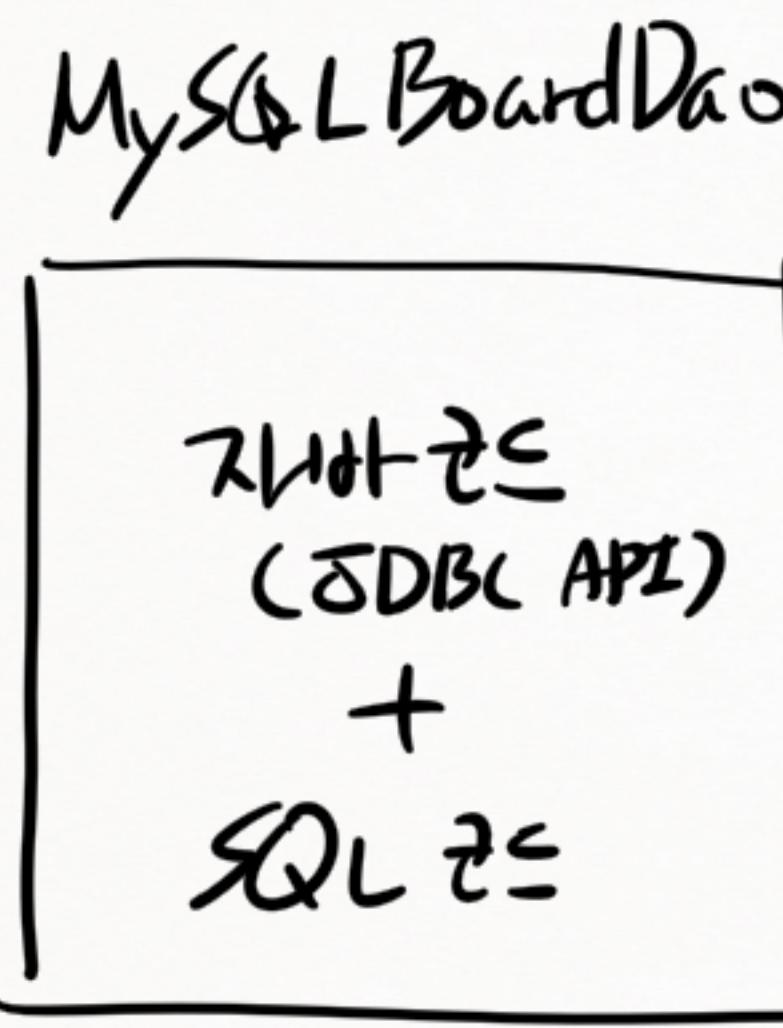
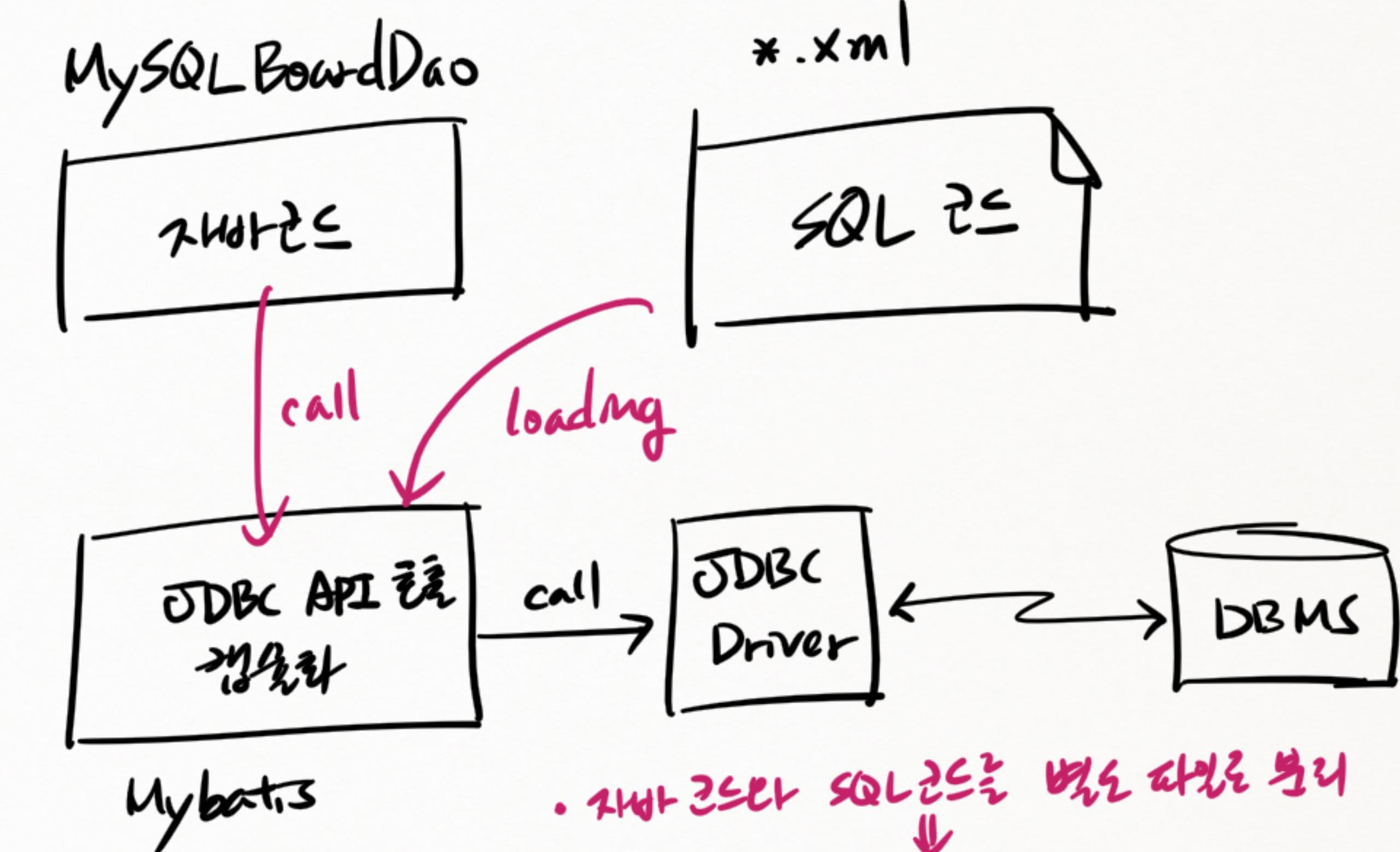


## 52. Mybatis SQL Mapper Framework

① 현황 → ② 향후

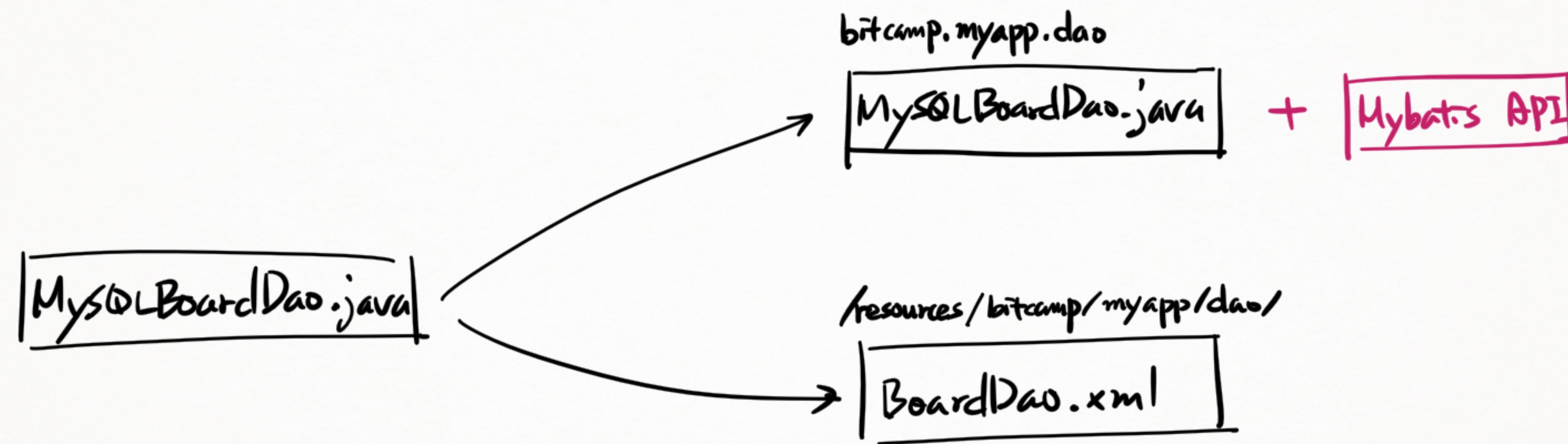


- 코드 관리가 어렵다
- JDBC API 쿼리 코드가 복잡하다.  
복잡한 코드



- 자바 코드와 SQL 구문을 별도 파일로 분리
- ↓  
• 자바 코드가 입니다  
• JDBC API 호출 코드를 정리  
• ↓  
• 자바 프로그래밍이 간결해집.

## 52. Mybatis SQL Mapper Framework



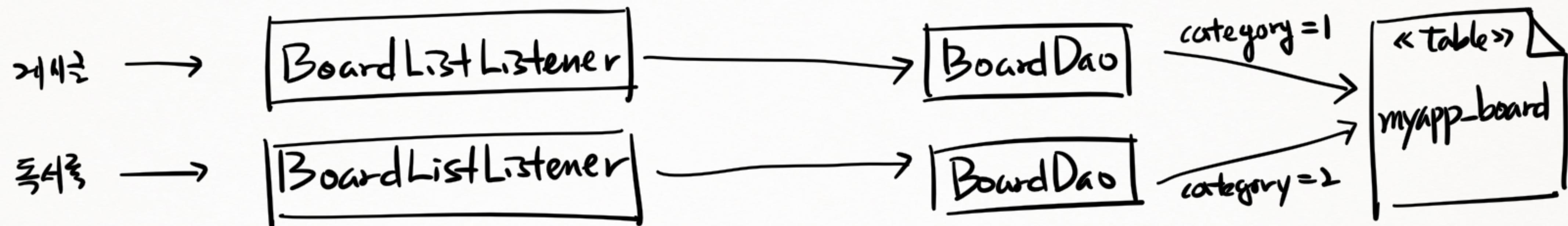
이전 대체 시



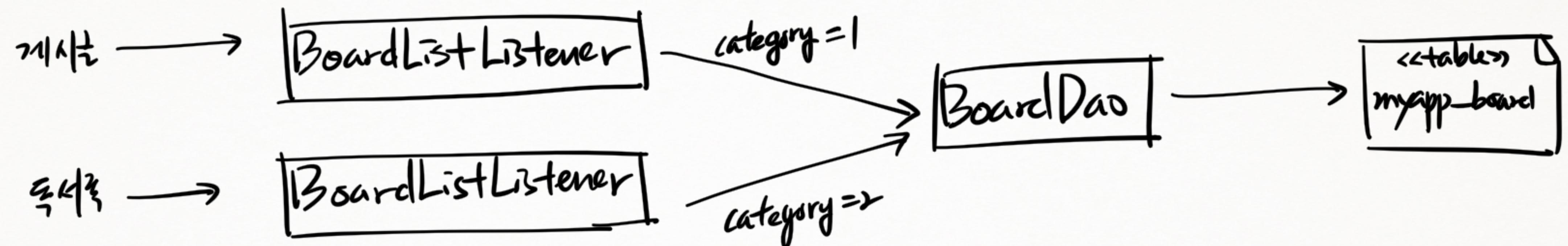
Mybatis 사용

\* Listener - DAO - Table 구조 변경

① 현황

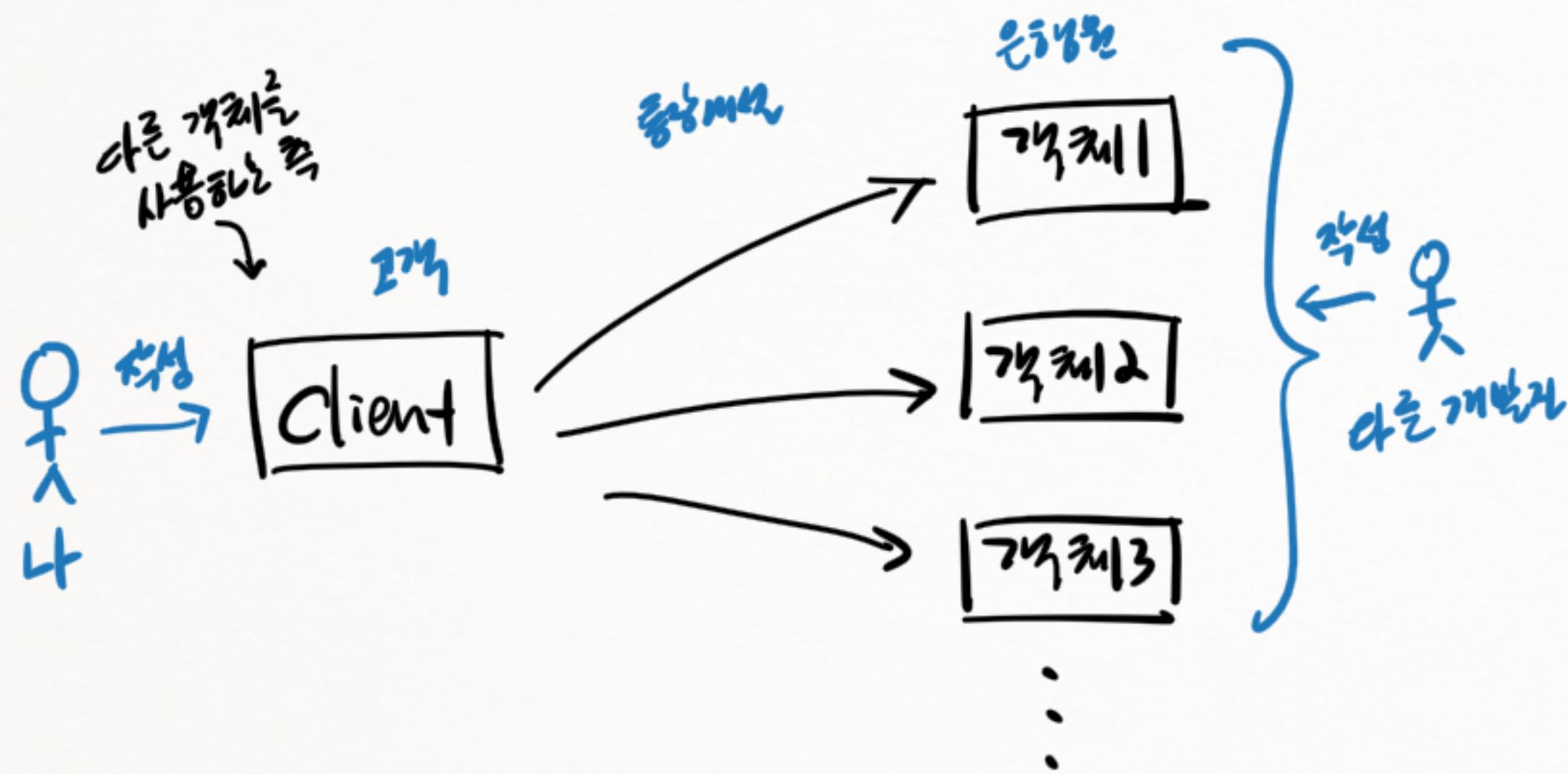


② 변경



## 53. Facade 패턴 적용

① 기존 구조



- Client는 여러 개의 Façade에 의존한다

의존 객체 변경에 영향을 자주 받는다

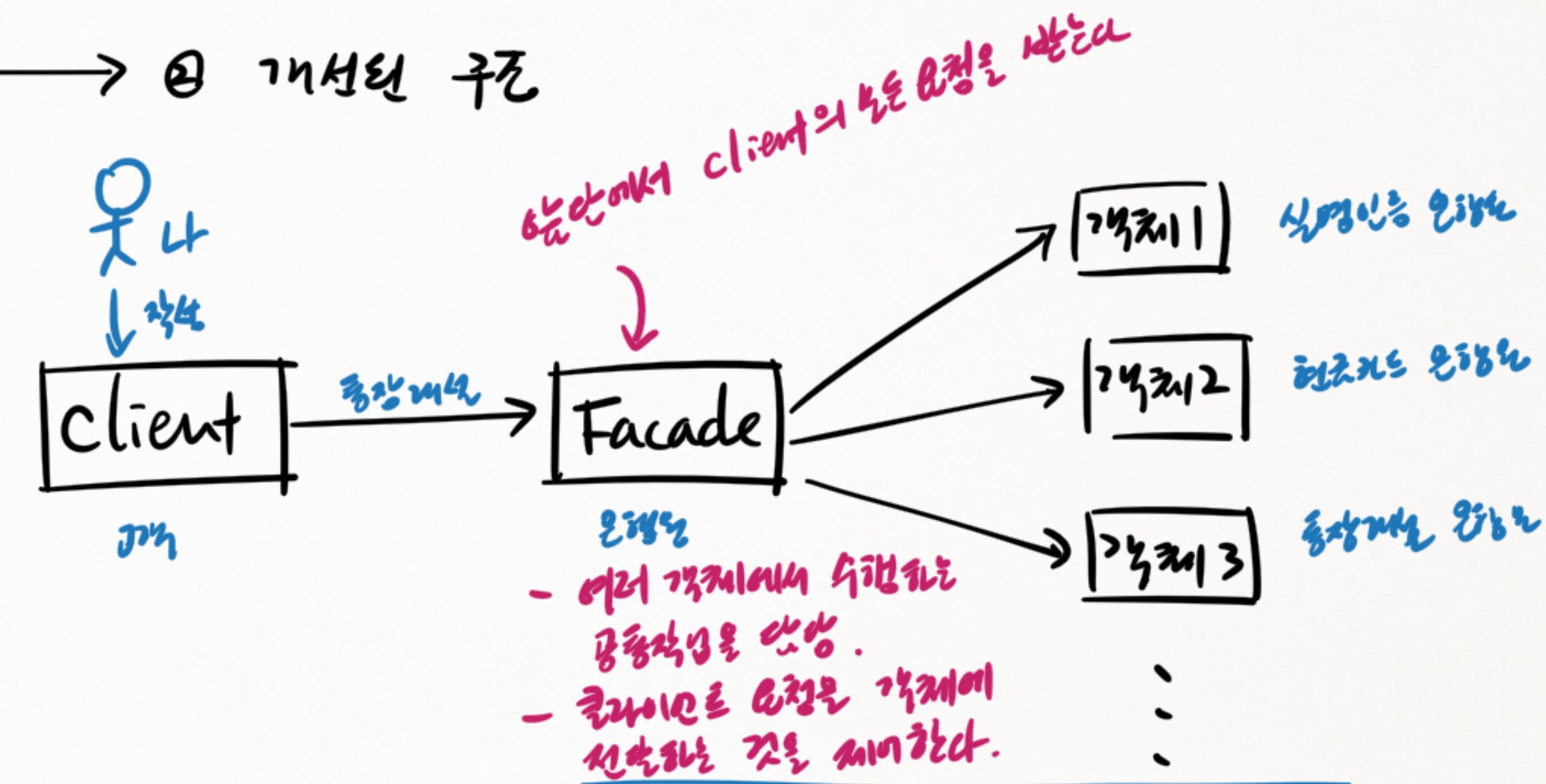
↓ 개선

GRASP

패턴의 "Low Coupling" 유지

다른 객체와의 관계를 줄이기

② 개선된 구조



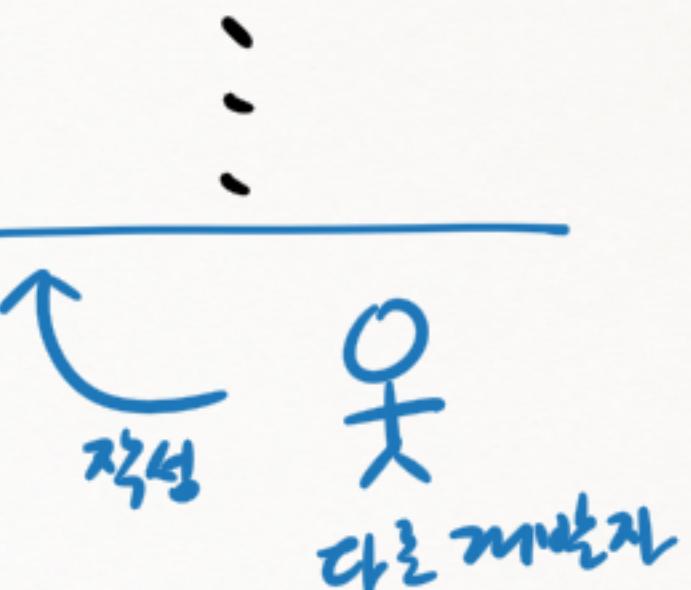
- 여러 Façade에 의존하는 공통작업을 단행.
- 클라이언트로 요청을 Façade에 전달하는 것을 제거함.

- Client는 Façade 역할의 객체만 사용.

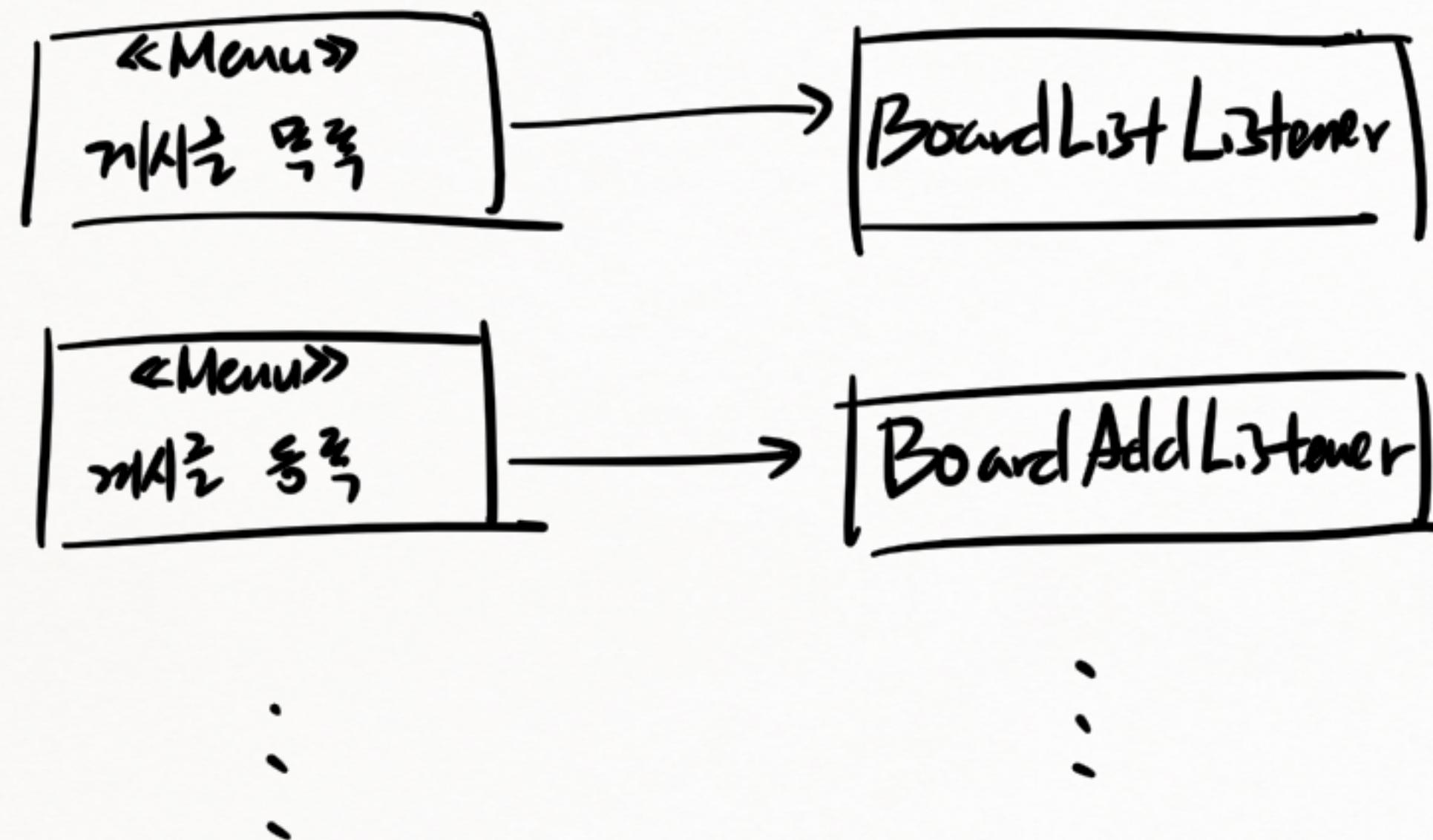
원래의 객체를 사용하는 것은 Façade가 한다.

↓

원래의 객체의 변화가 발생하는데도 Client는 영향 끼치지 않는다.

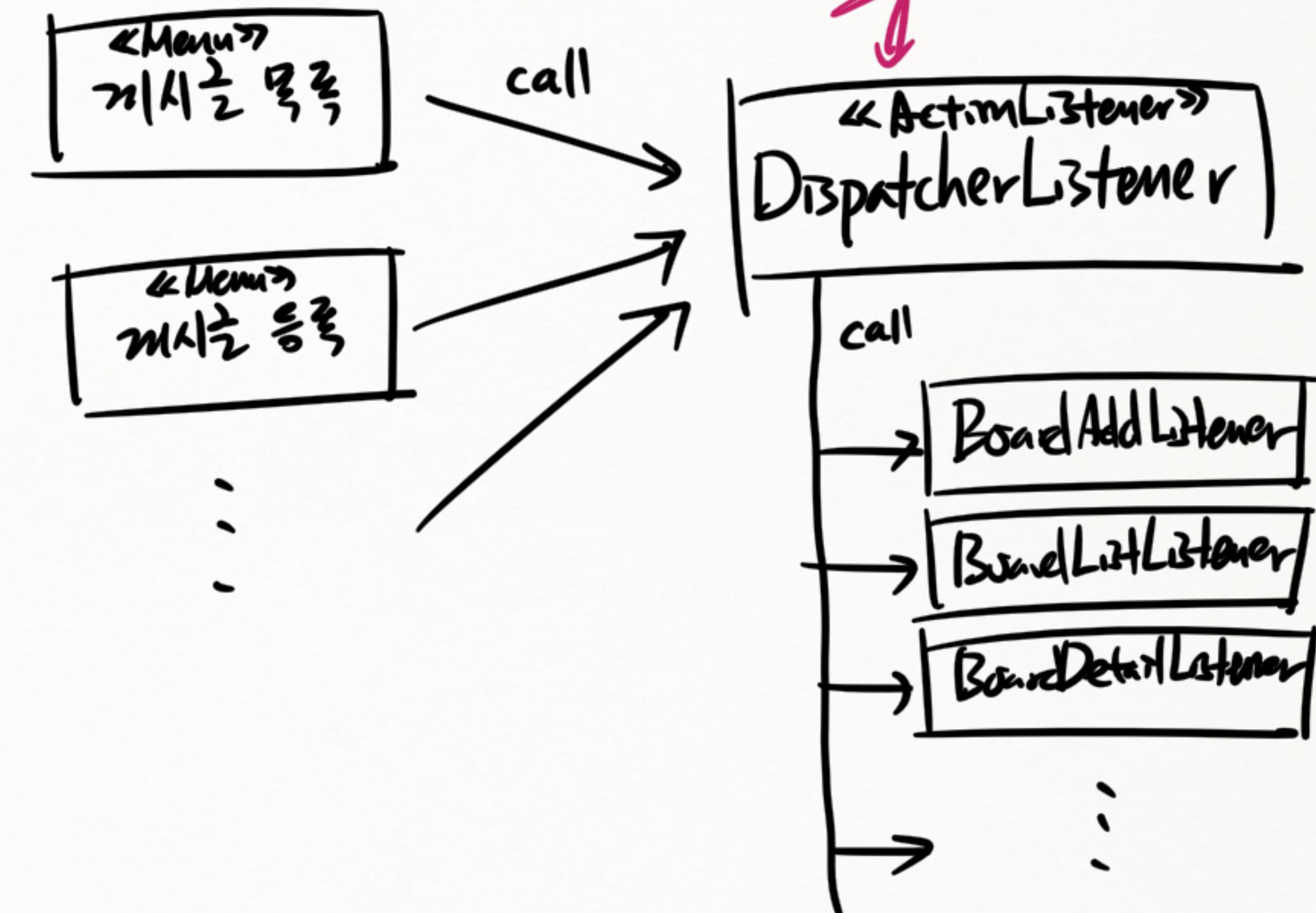


① 오전 구조



"Front Controller" 대신  
"

② Facade 구조



"Facade 구조"

## \* 54. IoC 컨테이너 활용하기

### IoC (Inversion of Control)

제어의 역전 = 역제어

#### ① Dependency Injection (의존객체 주입) 의존성 주입

{ 일반 : 필요한 객체는 만들어 쓴다.  
역제어 : 필요한 객체를 외부에서 끌어온다.  
    ↓

- 고체화 가능
- 콘트랙트에 맞는  
복잡한 시스템
- 풀업(mockup) 객체를 주입하여  
단위 테스트를 쉽게 할 수 있다

#### ② Event Listener

{ 일반 : 위 → 아래 순차적으로 코드를 실행  
역제어 : 특정 이벤트가 발생했을 때  
동록된 리스너가 실행된다.

“ 실행 코드가 실행된다.”

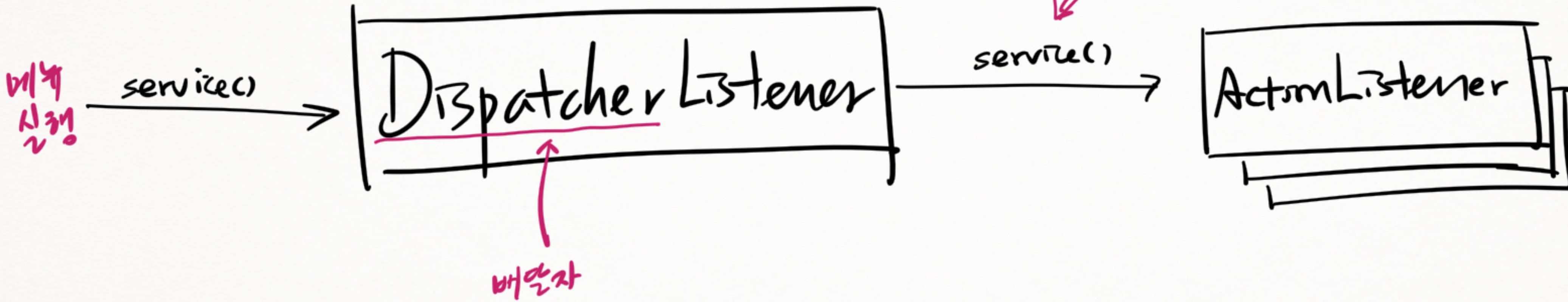
### \* IoC 컨테이너

= Bean Container + Dependency Injection

||  
Object의  
애칭? } Java(기자) object (⇒ = bean)

\* IoC 컨테이너 예제의 문제

GRASP 의 "High Cohesion" = IoC 컨테이너 예제



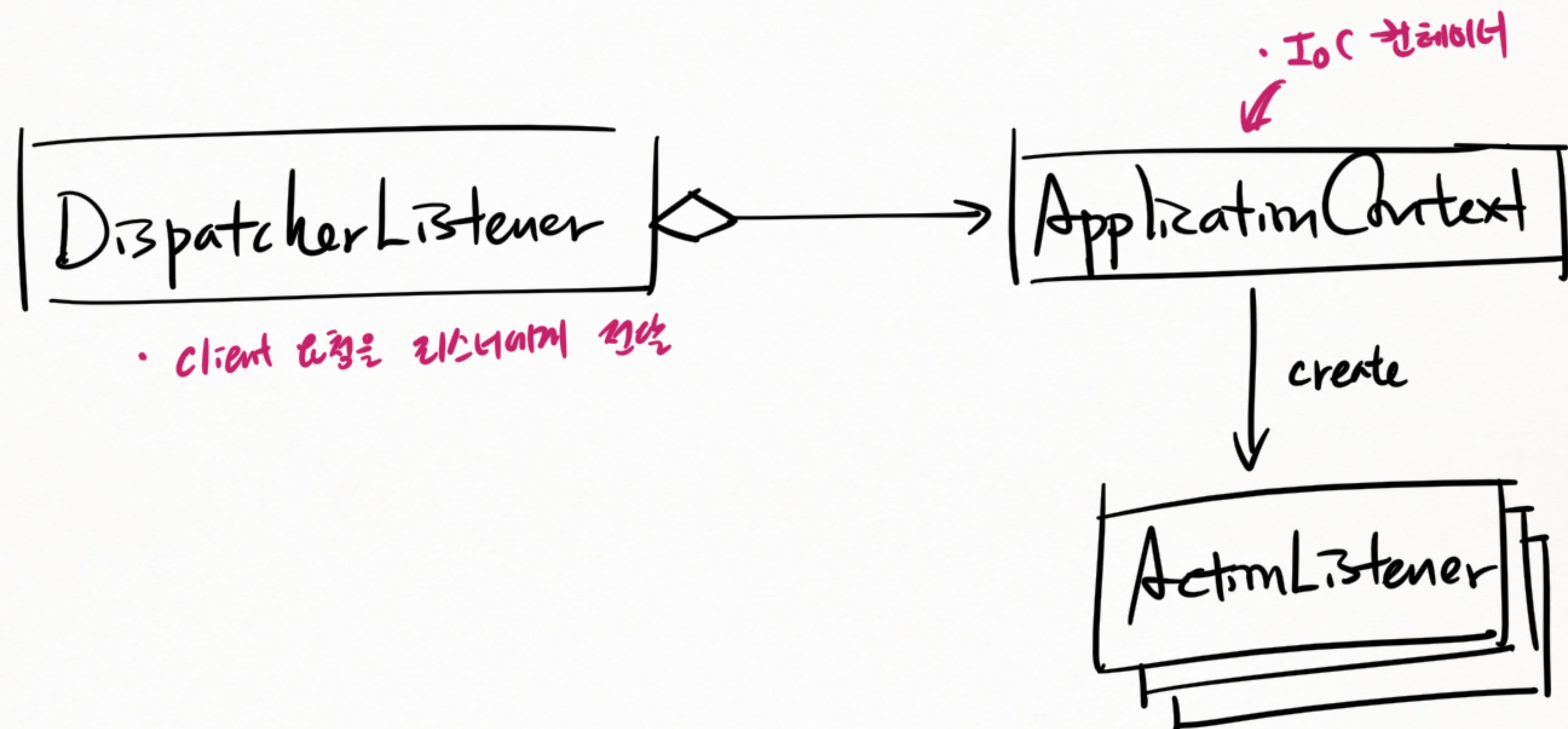
\* 하는 일

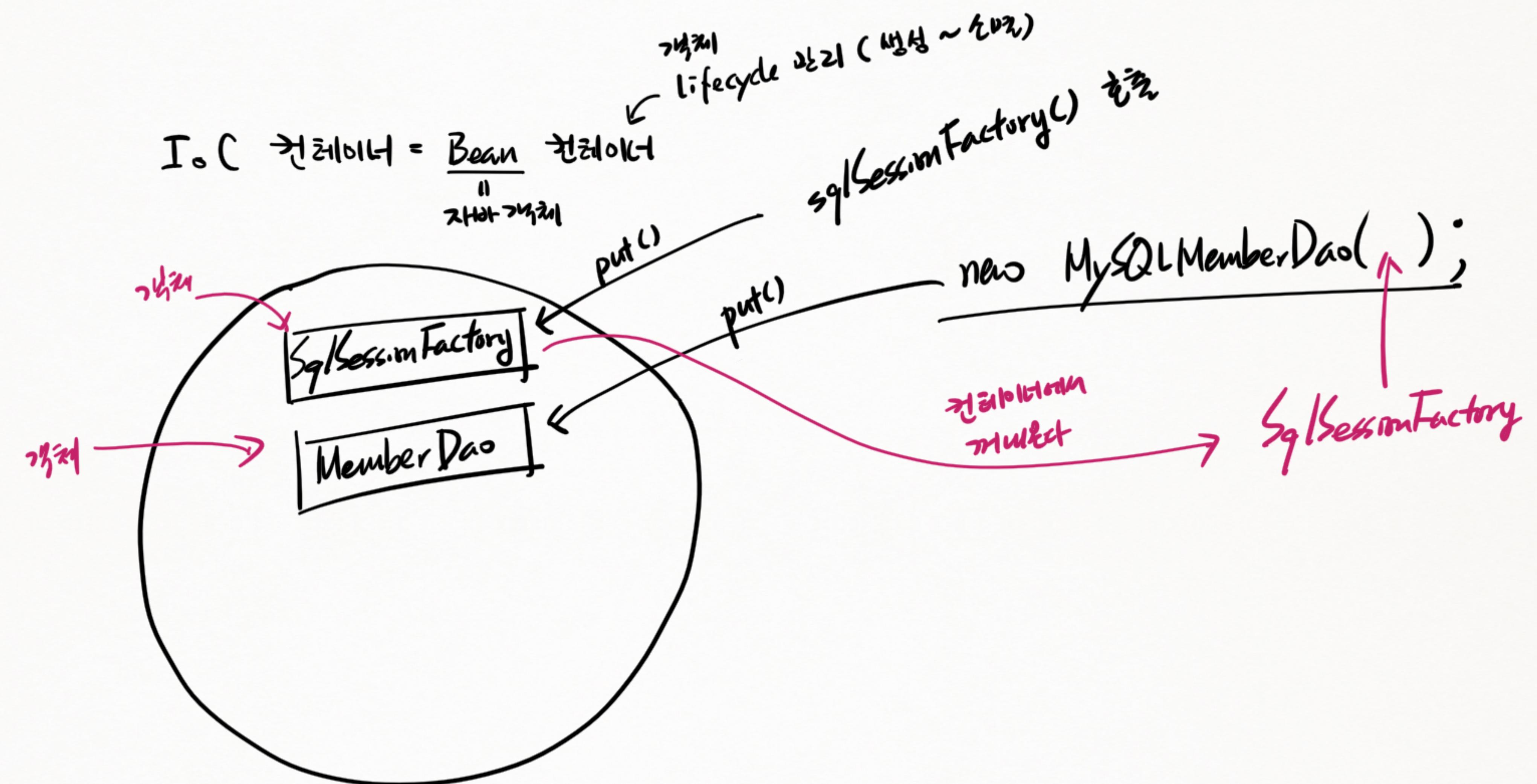
① Listener 구현 준비 = IoC 컨테이너 예제

② Listener의 Facade 예제

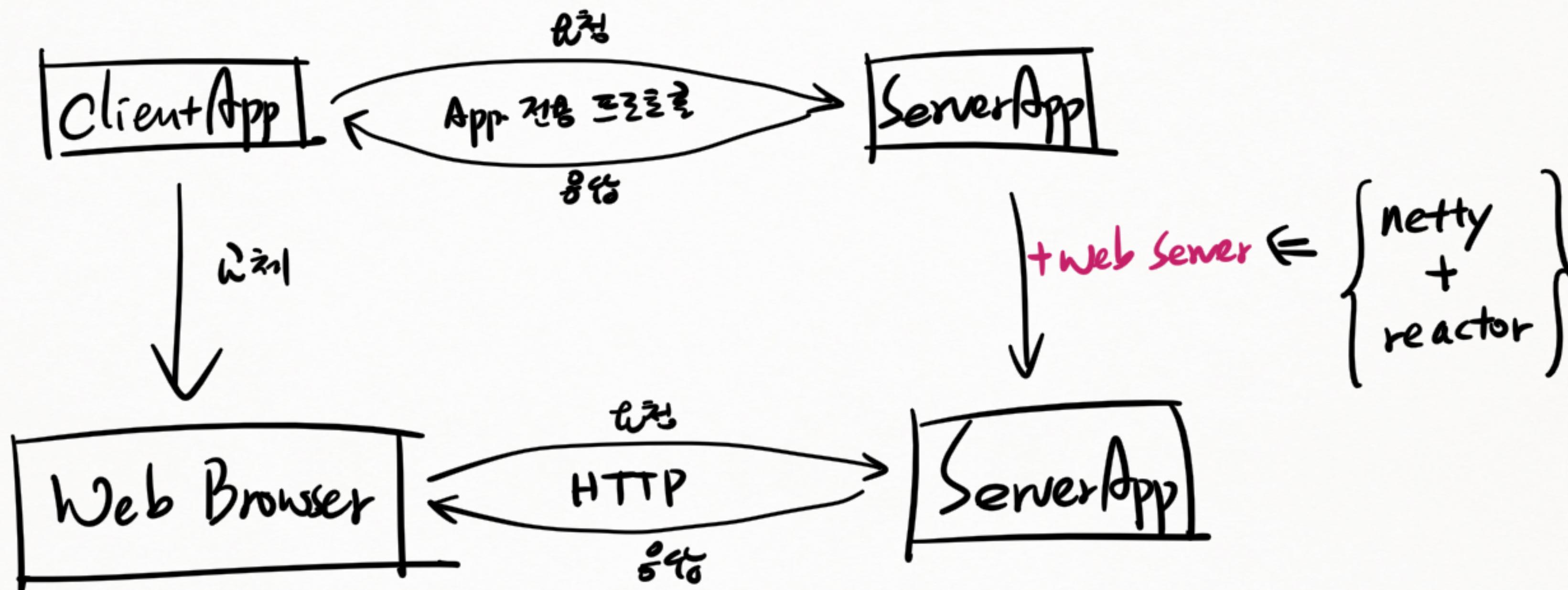
"Front Controller"

\* IoC 컨테이너 부리

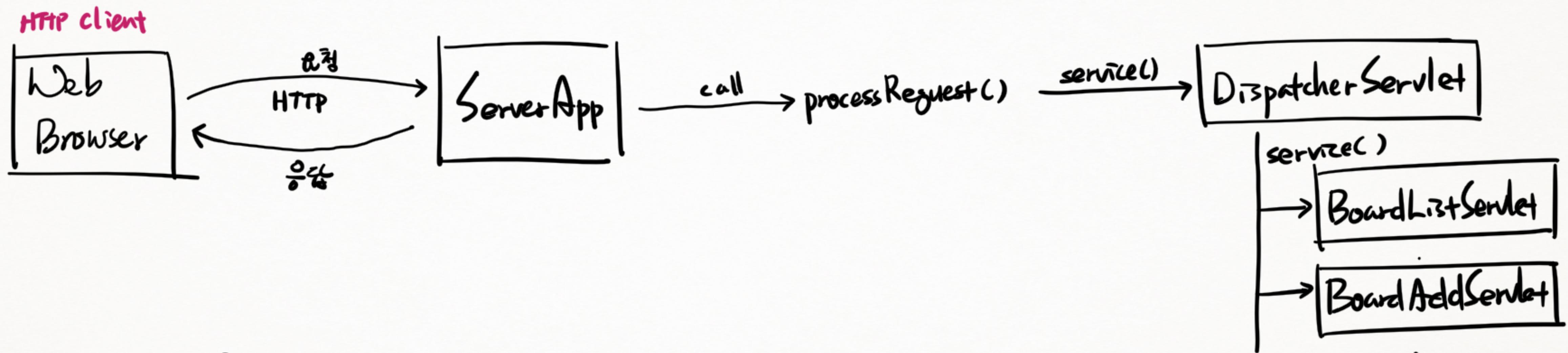




## 55. 웹기반 애플리케이션



\* 요청 - 응답 실행 과정



## Hyper-Text Transfer Protocol

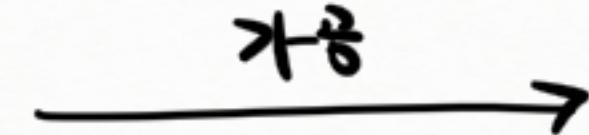
- 문서데이터 마크업이 존재
- 다른 문서와의 연결정보



HTML  
Markup  
Language

\* processRequest () 하는 일

HttpServletRequest



HttpServletRequest

- getServletPath()
- getParameter()
- getParameterValues()
- getAttribute()
- setAttribute()

http://localhost: port /board/detail ? category=1 & no=207

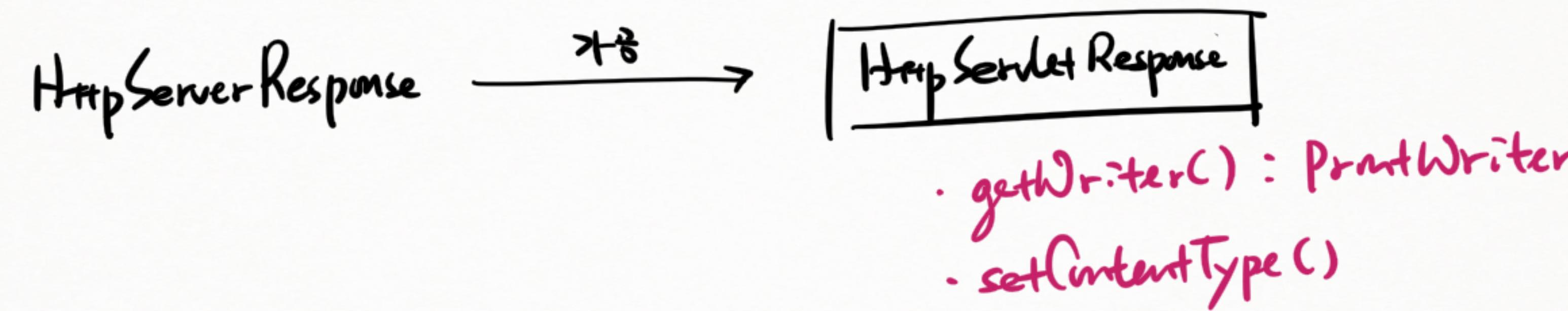
Servlet Path

↓  
getServletPath()

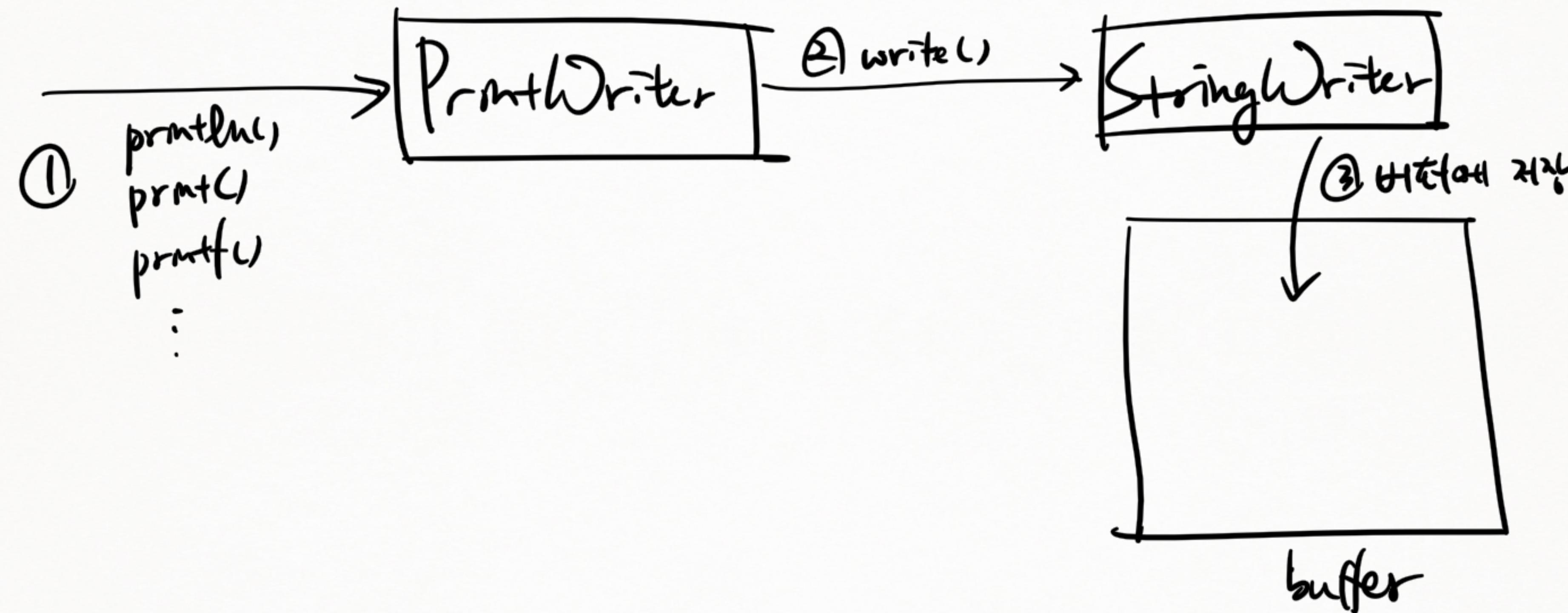
Query String

↓  
getParameter()

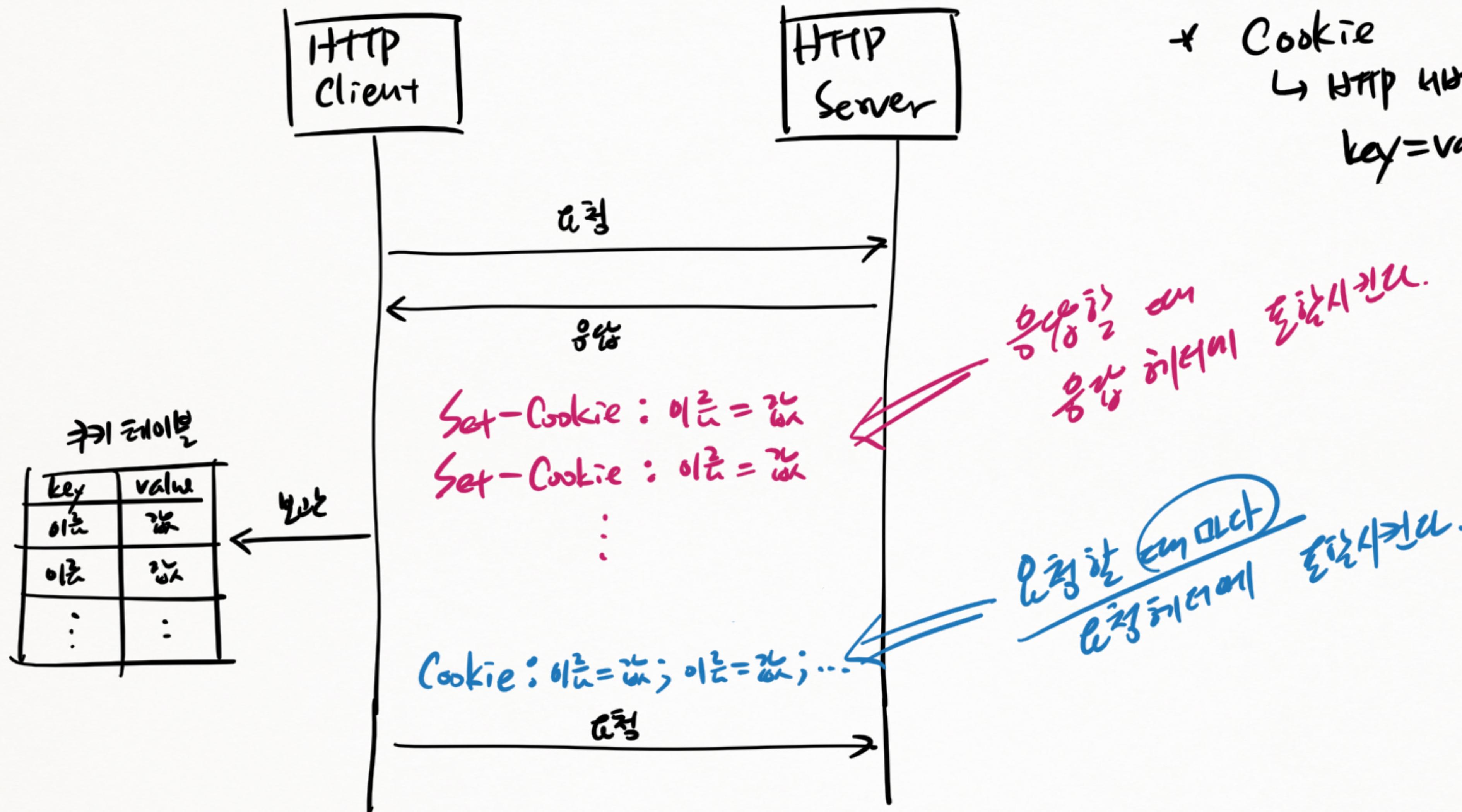
\* processRequest () 하기 위한



\* StringWriter의 흐름



## \* HTTP Client et Cookie

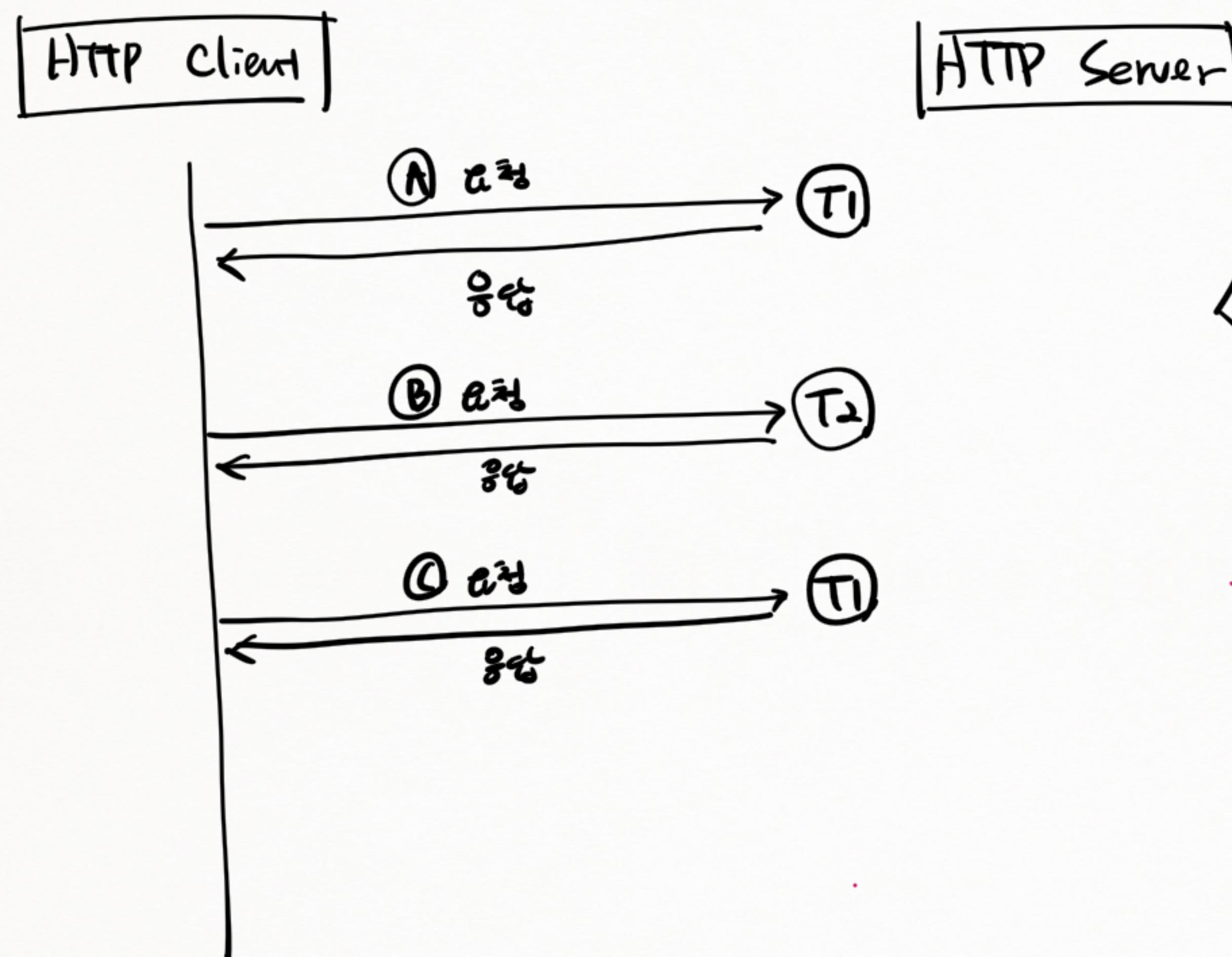


## \* Cookie

↳ HTTP 헤더가 HTTP 클라이언트에  
key=value 형식으로 전달시킨다!

HTTP 클라이언트는  
 요청할 때마다  
 키를 뒷면附加值로  
 추가로 전달한다.

## \* HTTP client 와 스레드 관계



\* 통신방식

connection-oriented

TCP: 연결후통신

예) telnet, ftp, http, smtp

connectionless

UDP: 연결없이통신

예) ping, 방송

stateful  
연결 - 요청 - 응답 - 끝기  
반복

stateless  
예) HTTP

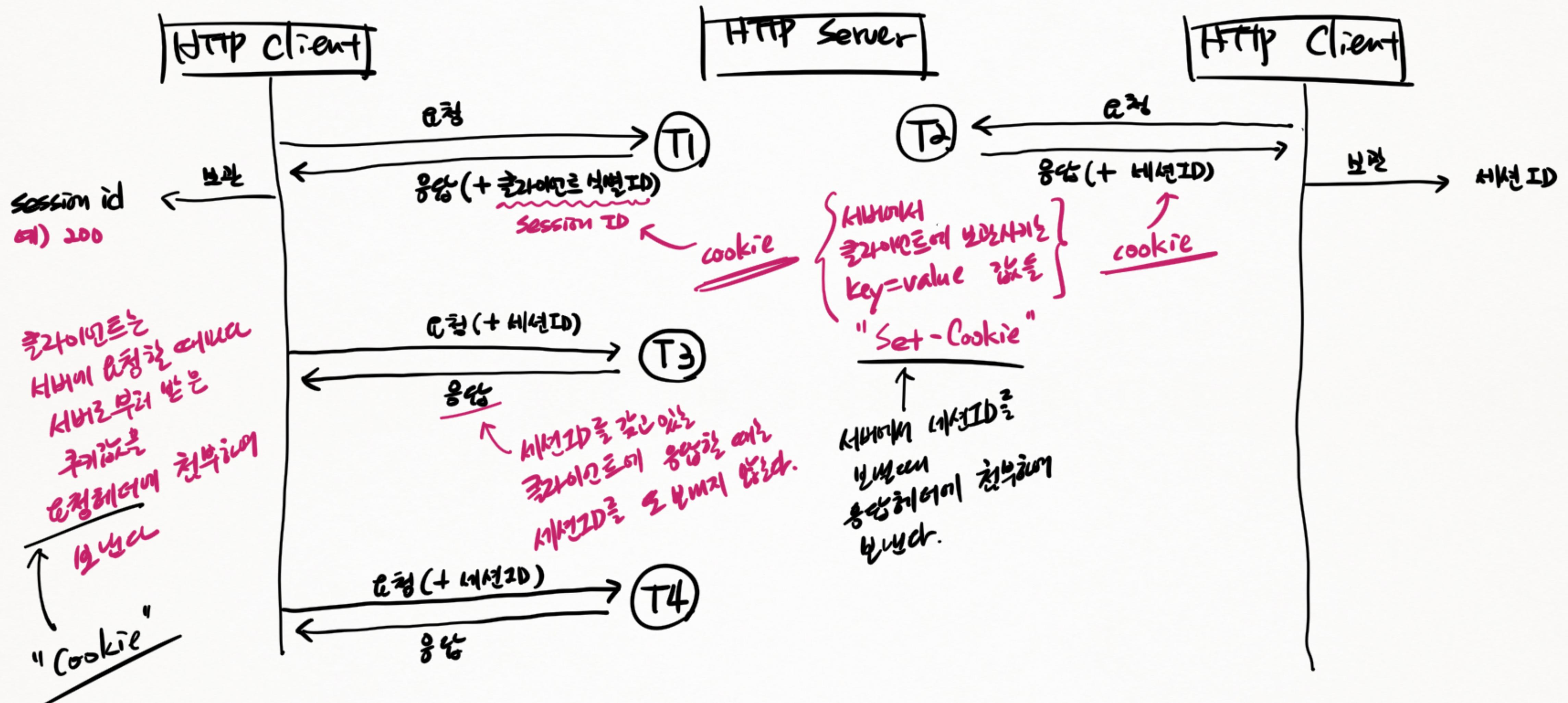
연결 - 요청 - 응답 - 끝기

연결 - 요청 - 응답 - 끝기

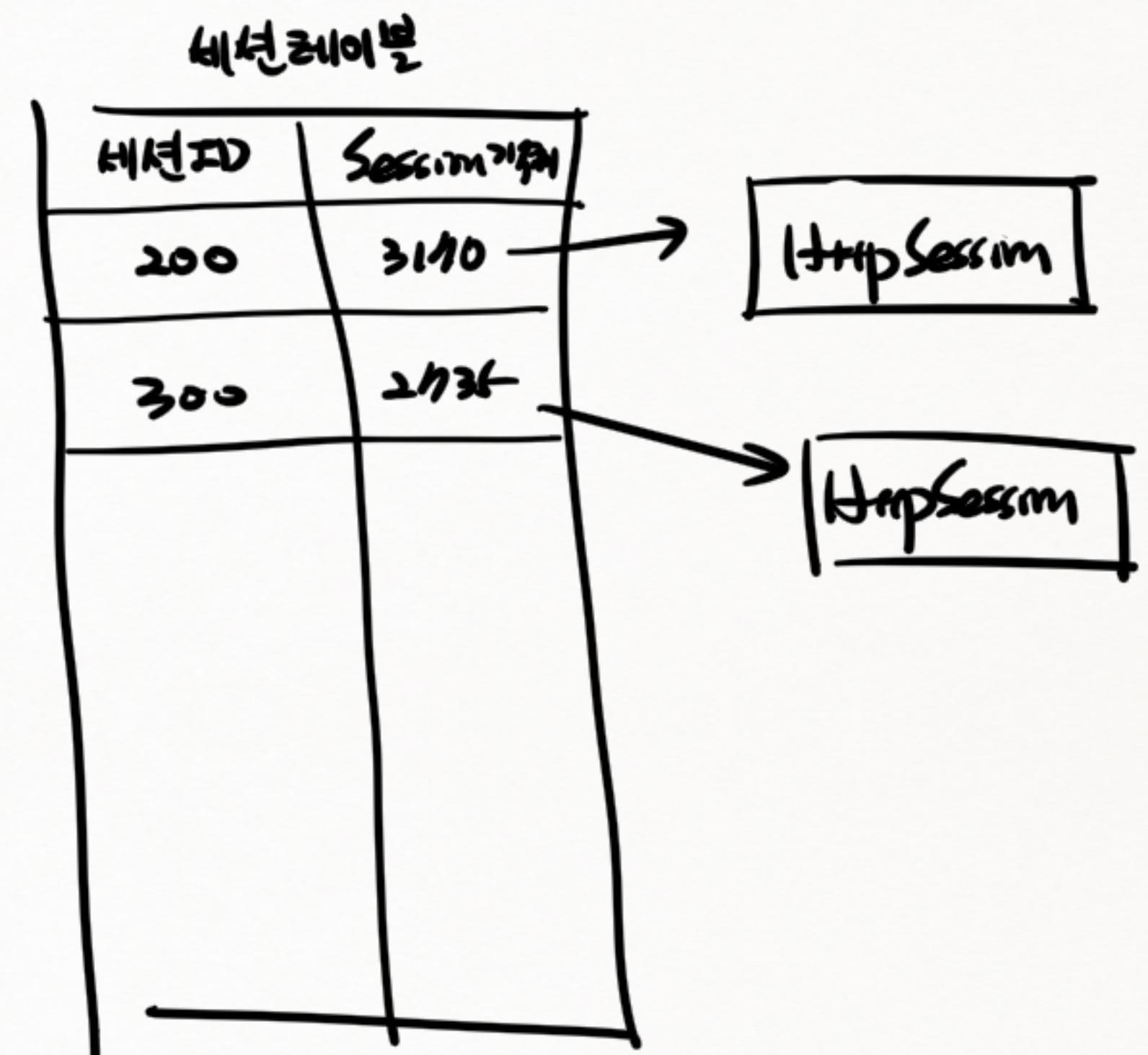
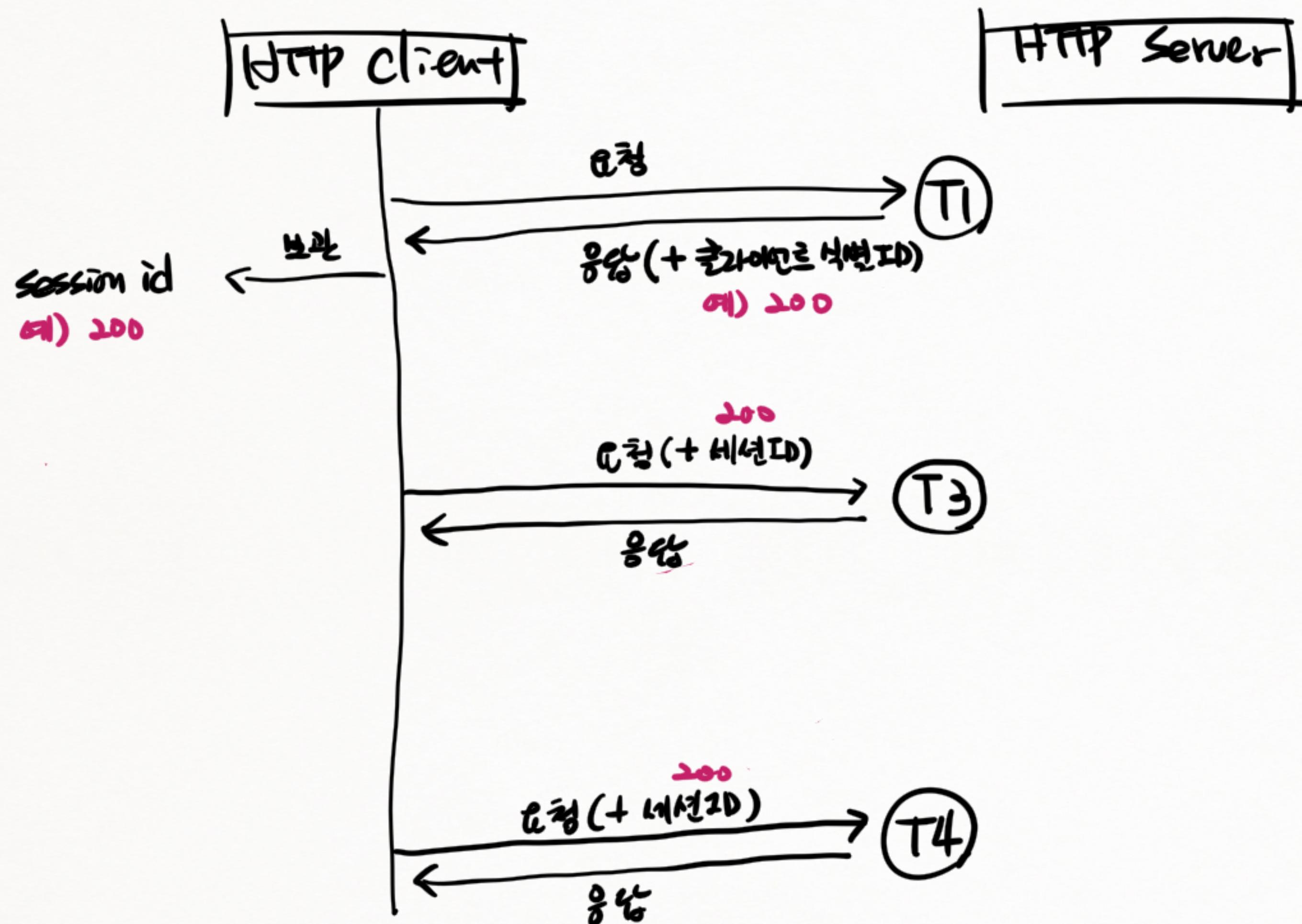
## \* HTTP 통신

- stateless 방식으로 통신한다
- 요청할 때마다 새로 연결한다
- 요청을 처리하는 스레드가 다른 수 있다
- 요청과 요청 사이엔 데이터 공유 불가!
  - 스레드에 보관해보아 사용한다.
  - HttpServletRequest 가 요청데이터를 사용한다.

\* HTTP client et Session — ① 쿠키를 이용하여 클라이언트를 구분하기



\* HTTP client et Session - ② 같은 클라이언트의 요청을 사이에서 태이러링하기



## \* HTTP 요청 방식 - GET vs POST

① GET 요청 : 데이터 조회

GET /board/list?category=1 HTTP/1.1 ↳ Request Line

Host : localhost:8888 ↳  
User-Agent : 웹브라우저 혹은 앱 ↳  
Cookie : 쿠키 정보  
⋮  
⋮

↳ ← 맨 끝에 뒷주소를 보낸다.

/board/list?category=1



Get 방식으로 요청하는 경우

서버가 보내는 파라미터는 URL의 포함된다. "Query String"

② POST : 데이터 생성, 변경 (REST에서 PUT 요청)

POST /auth/login HTTP/1.1 ↳

Host : localhost:8888 ↳  
User-Agent : 웹브라우저 혹은 앱 ↳  
Cookie : 쿠키 정보 ↳  
⋮

Content-Length : 34 ↳  
Content-Type : application/x-www-form-urlencoded  
⋮

{email=aaa%40test.com&password=1111}

↑  
post 방식으로 요청하는 경우를 Entity

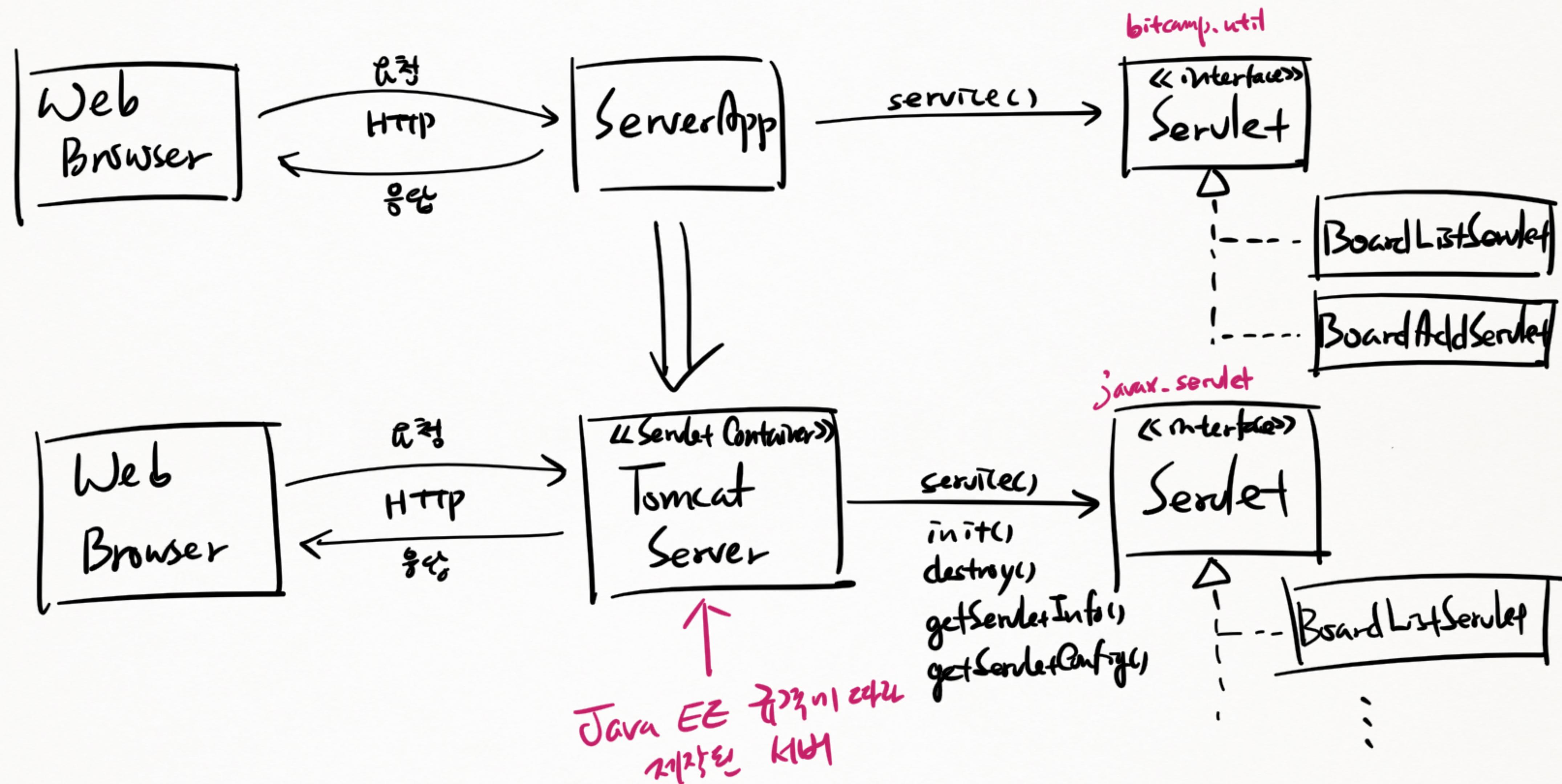
"Entity Body"

(Message Body)

## \* HTTP 요청 방식 - GET vs POST

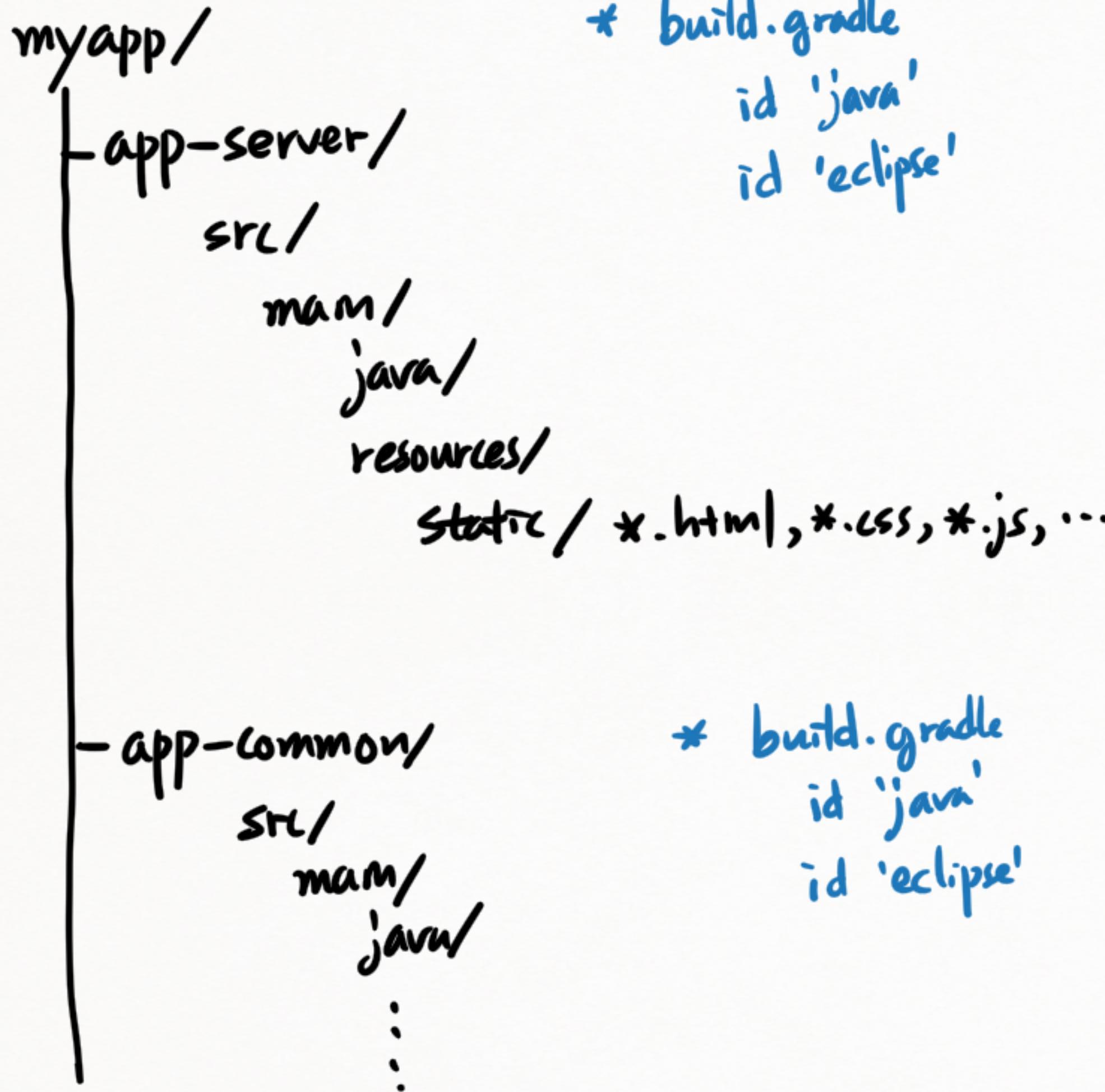
| GET 방식              |   | POST 방식   |
|---------------------|---|---|
| Binary 데이터 전송<br>크기 | 제한된다<br>64KB, 8KB<br><br>그러나 크고<br>데이터의<br>때문에<br>제한되는<br>파일 전송이<br>제한된다. | 제한없다  |
| 문자<br>데이터 전송        | 문자 . 단 Base64로 인코딩 했으면<br>문자로 변환되어 전송할 수 있다.                              | 요청 데이터 끝 부분, message body 부분이<br>첨부하기 때문에 전송 가능!  |
| 보안                  | 유형별로는 보안을 개선해 보았지만<br>↓<br>파라미터 값이 노출                                     | 파라미터 값이 부분이 첨부되는<br>↓<br>사용하기 별로 좋지 않다  |
| 성능                  | URL과 데이터를 함께 보내야 하는데<br>기본<br>↓<br>조회량이 많아                                | <ul style="list-style-type: none"><li>· 대량의 데이터 전송</li><li>· 데이터의 데이터 전송</li><li>· URL에 파라미터를 넣기로 삼지 않을 때</li></ul> |

## 56. JavaEE 도입

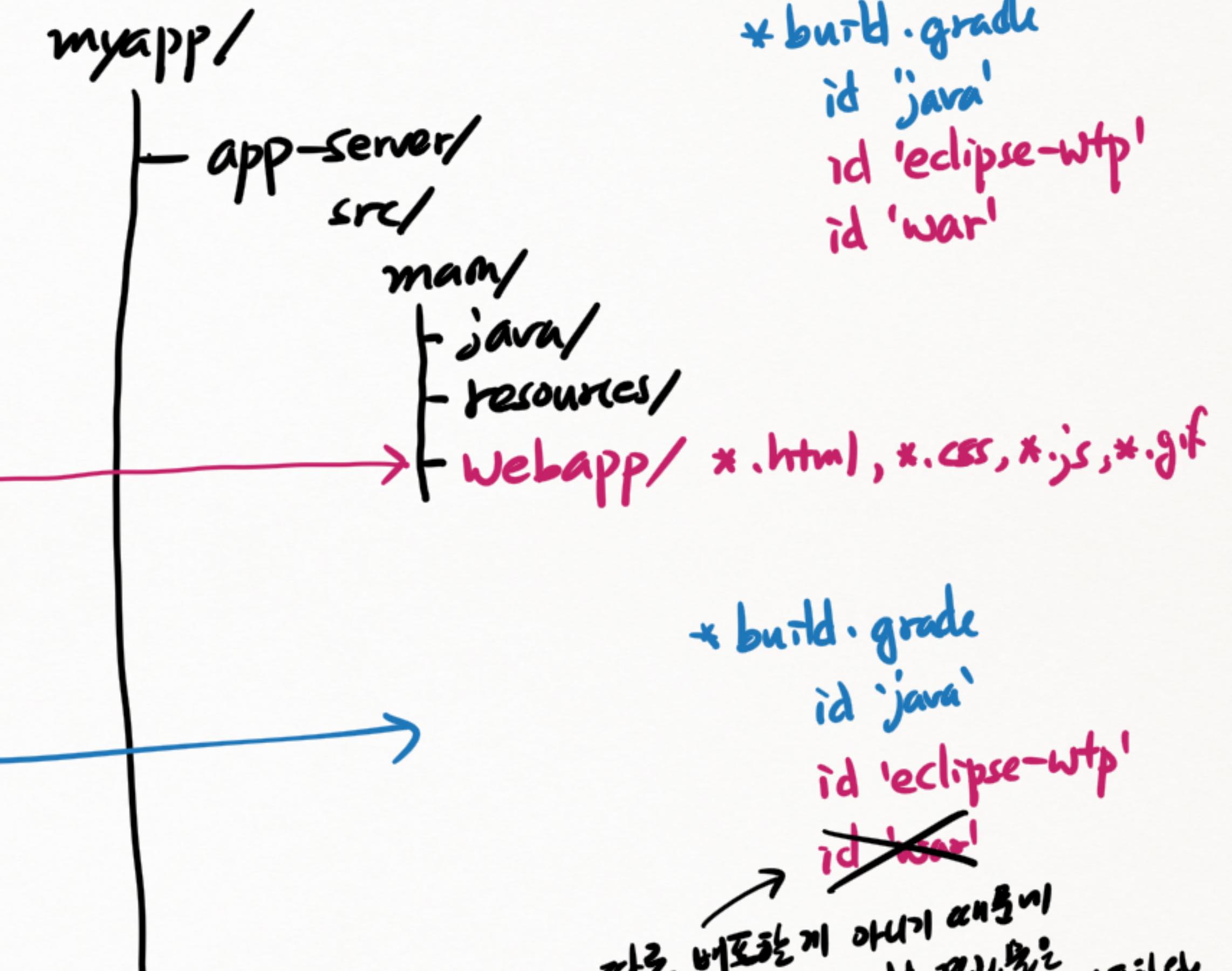


\* 기존 프로젝트 → Servlet 프로젝트 migration(이전)

## ① 기존 프로젝트



## ② Servlet 프로젝트



- \* war을 비롯한 파일은 제거되었습니다.
- 이제는 웹app의 build.gradle을  
app-server의 편집기에서 편집하세요

\* 서버주로 설정

서버 주로

컨텍스트 주로

http://localhost:8888/myapp/index.html  
application /board/list?category=1  
/init

\* build.gradle

eclipse {

:

wtp {

facet { ... }

component {

contextPath = "/"

}

↑  
서버주로 지정

http://localhost:8888/index.html

↑  
myapp 주로를 서버주로로 설정

위에 예술과 미술전의 contextPath를 지정하기 때문

NAR {

archiveBaseName = "myapp"

}

war 파일명이 사용된다.