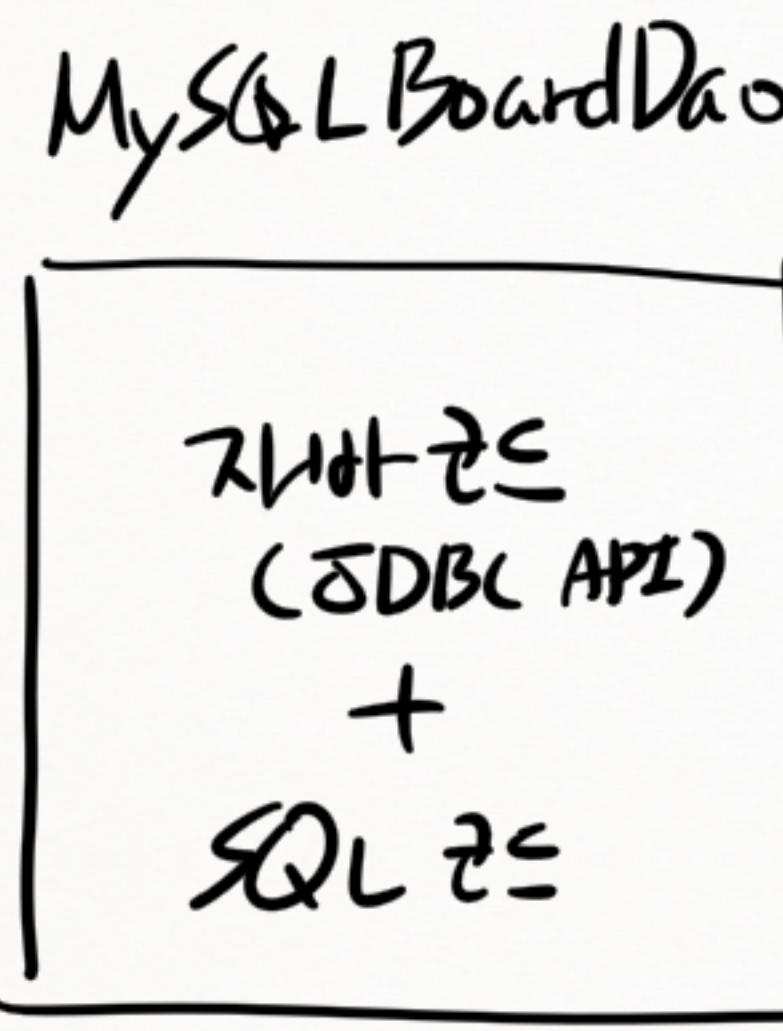
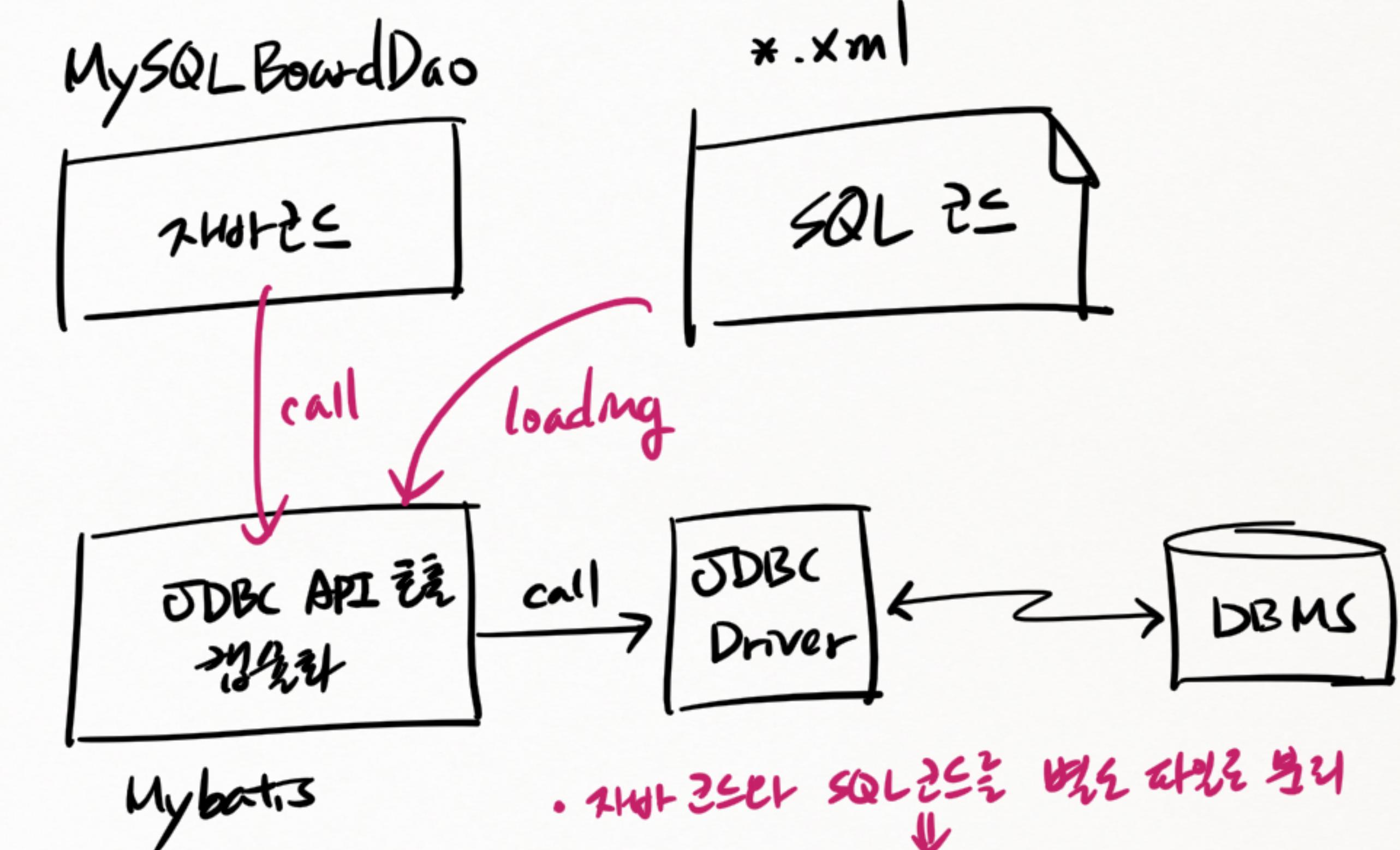


52. Mybatis SQL Mapper Framework

① 현황 → ② 향후

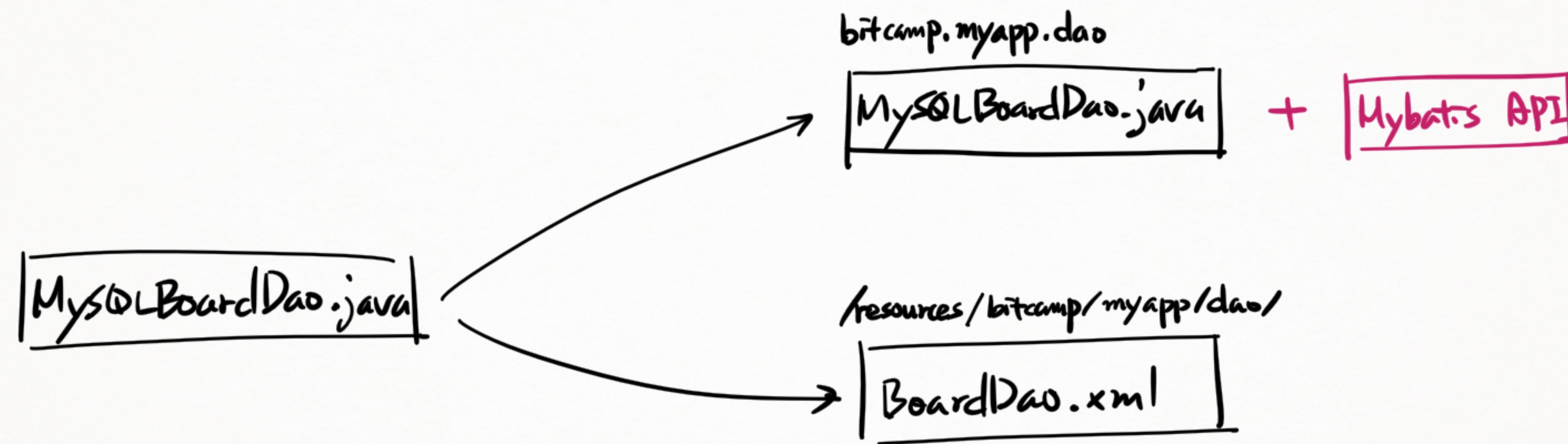


- 코드 관리가 어렵다
- JDBC API 쿼리 코드가 복잡하다.
복잡한 코드



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

52. Mybatis SQL Mapper Framework



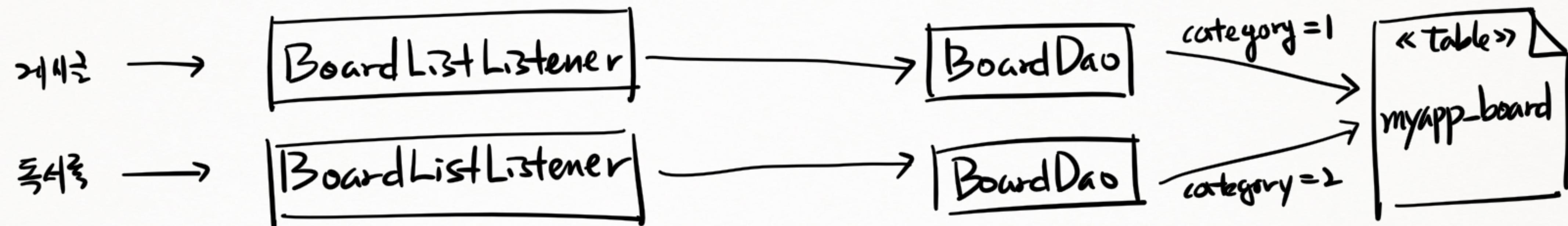
이전 대체 시



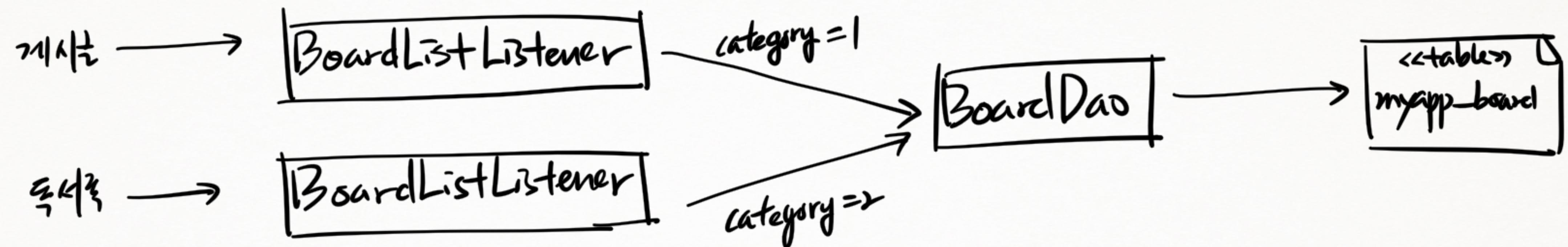
Mybatis 사용

* Listener - DAO - Table 구조 변경

① 현황

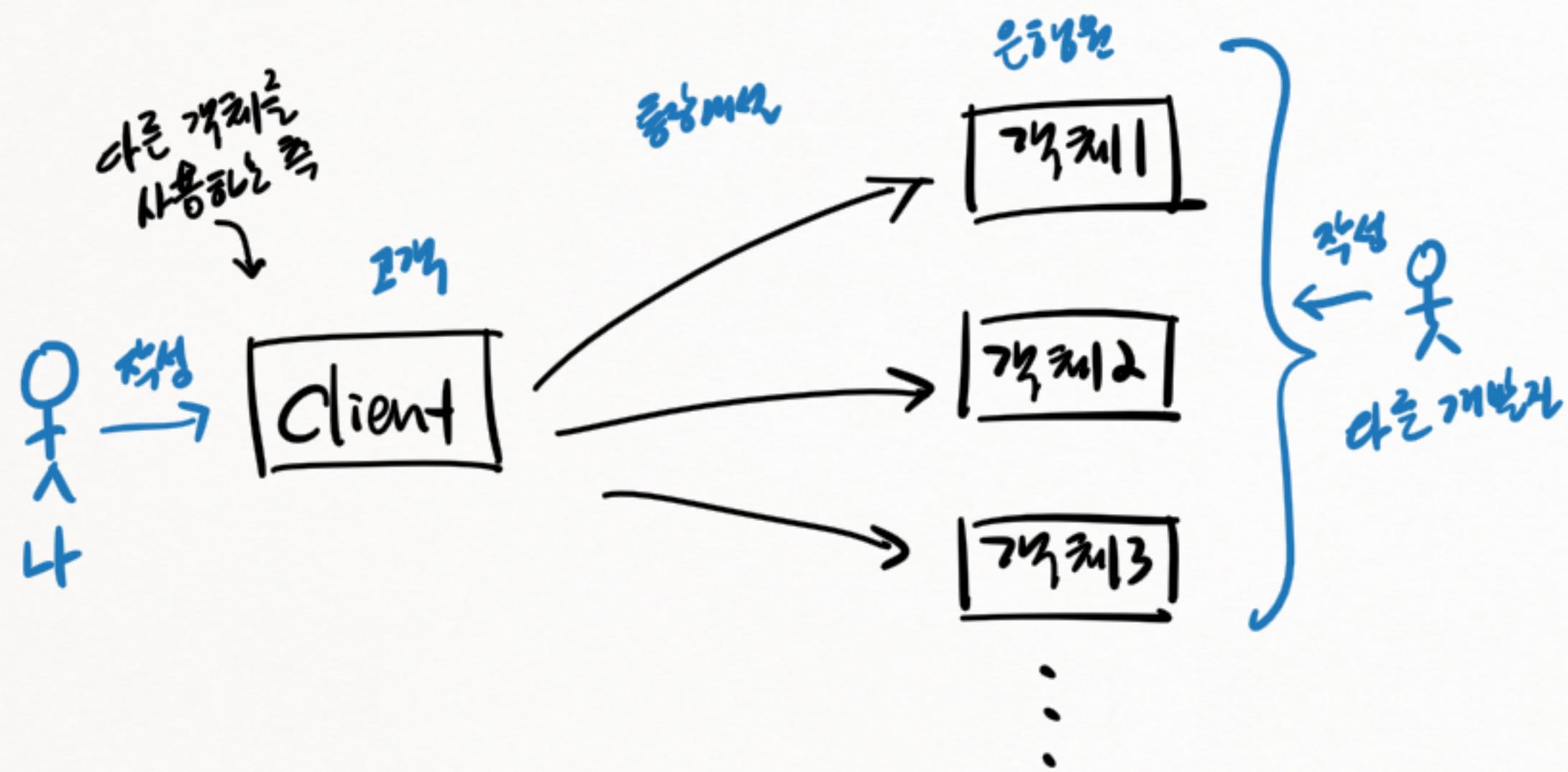


② 변경



53. Facade 패턴 적용

① 기존 구조



- Client는 여러 개의 Façade에 의존한다

의존 객체 변경에 영향을 자주 받는다

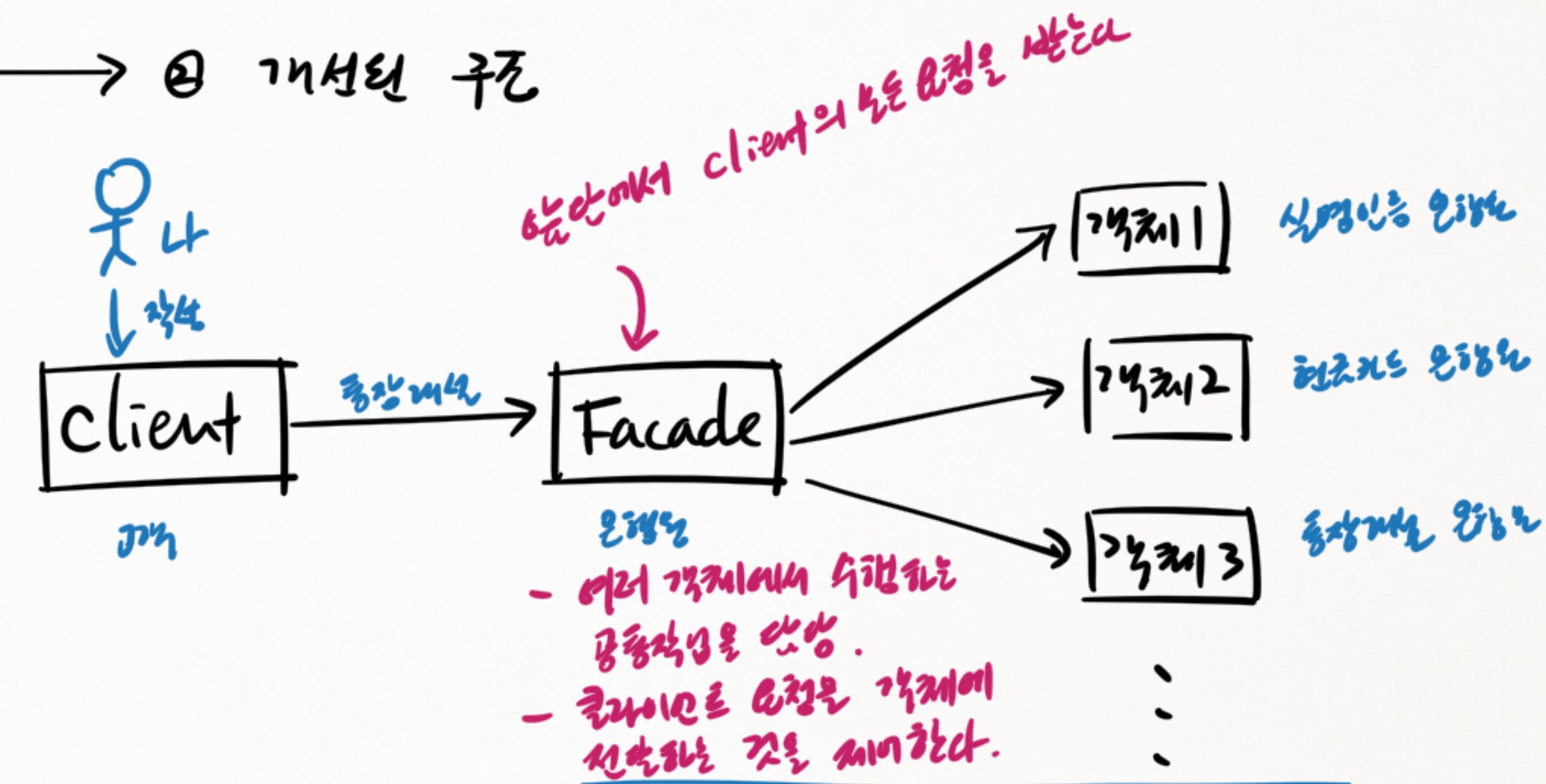
↓ 개선

GRASP

패턴의 "Low Coupling" 유지

다른 객체와의 관계를 줄이기

② 개선된 구조

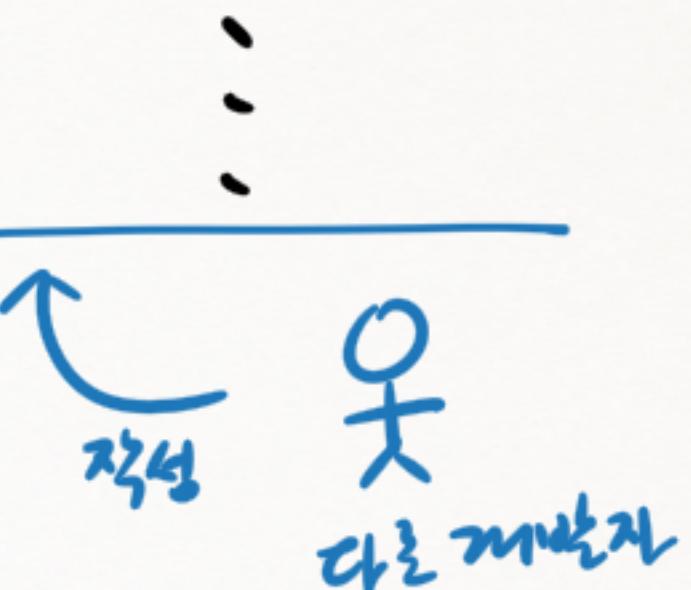


- 여러 Façade에서 수행되는 공동작업을 단행.
- 클라이언트로 요청을 Façade에 전달하는 것을 제거함.

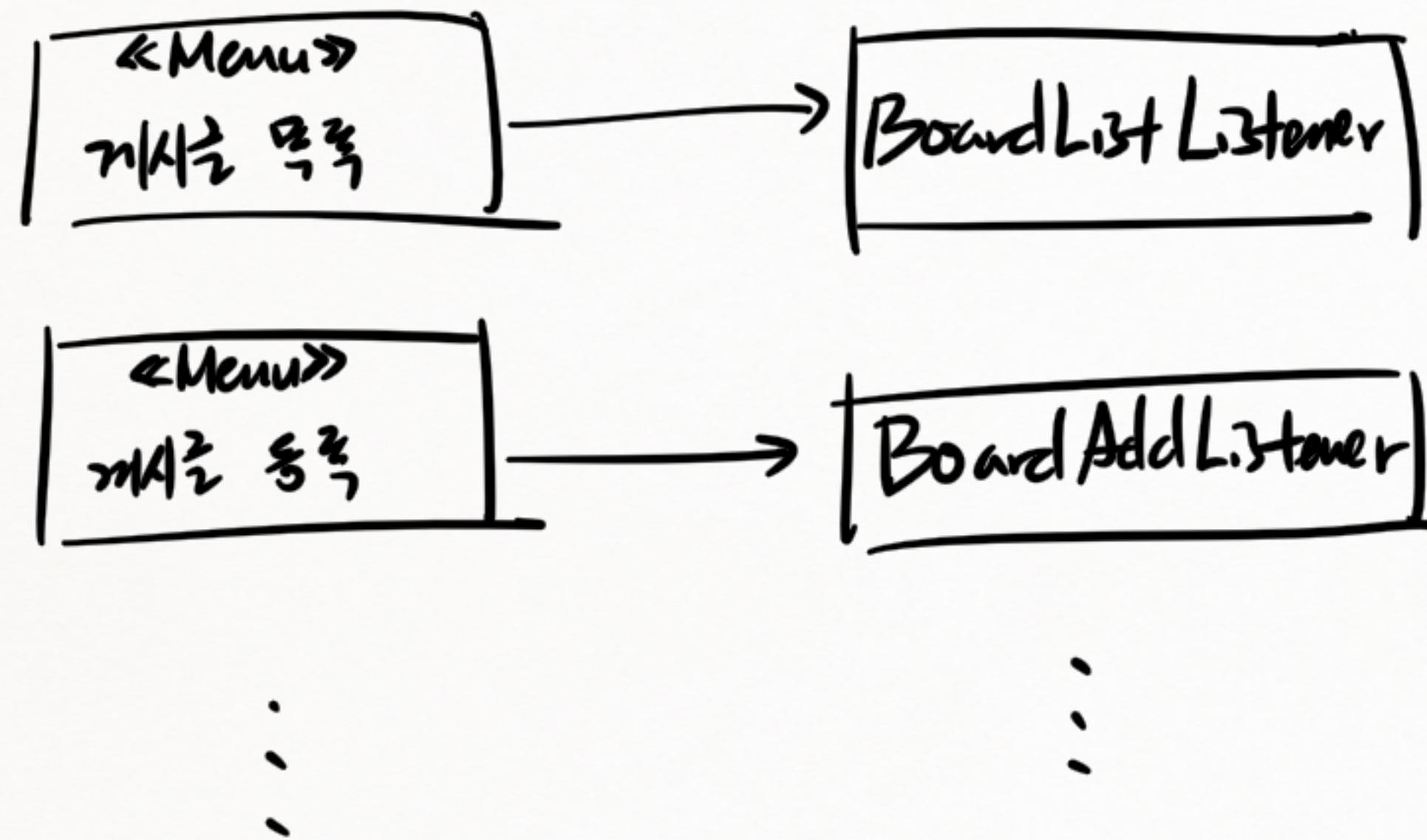
- Client는 Façade 역할의 객체만 사용.

원래의 객체를 사용하는 것은
Facade가 한다.

원래의 객체의 변화가 발생하는데
Client는 영향 끼치지 않는다.

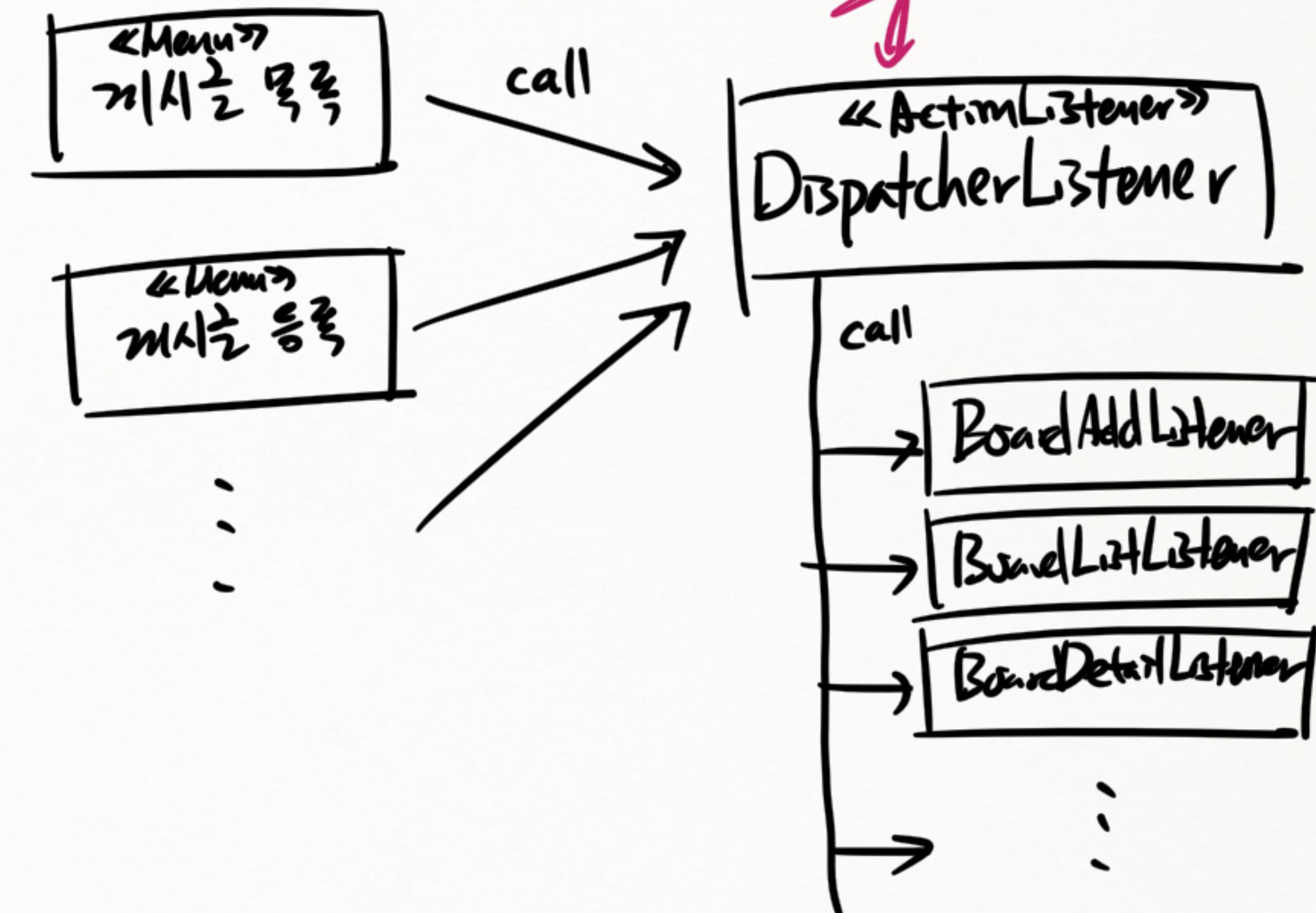


① 오전 구조



"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.”

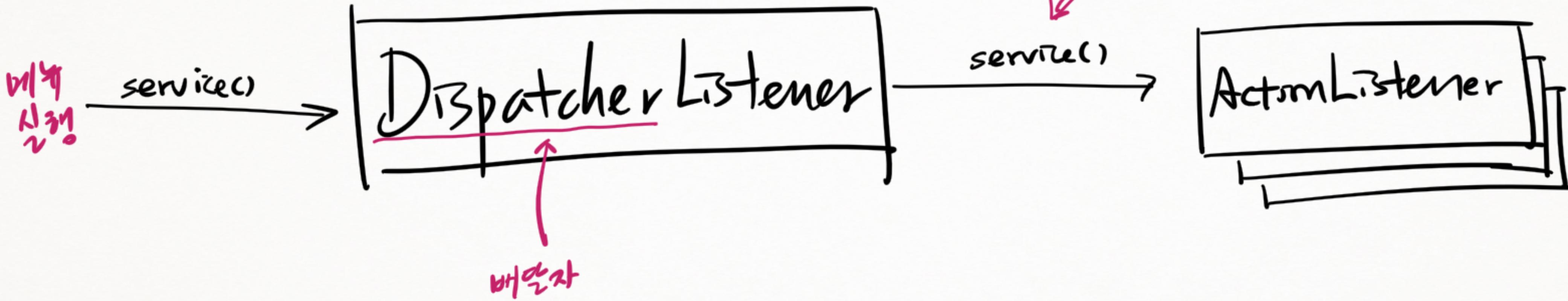
* IoC 컨테이너

= Bean Container + Dependency Injection

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

* IoC 컨테이너 예제의 문제

GRASP 의 "High Cohesion" = IoC 컨테이너 예제



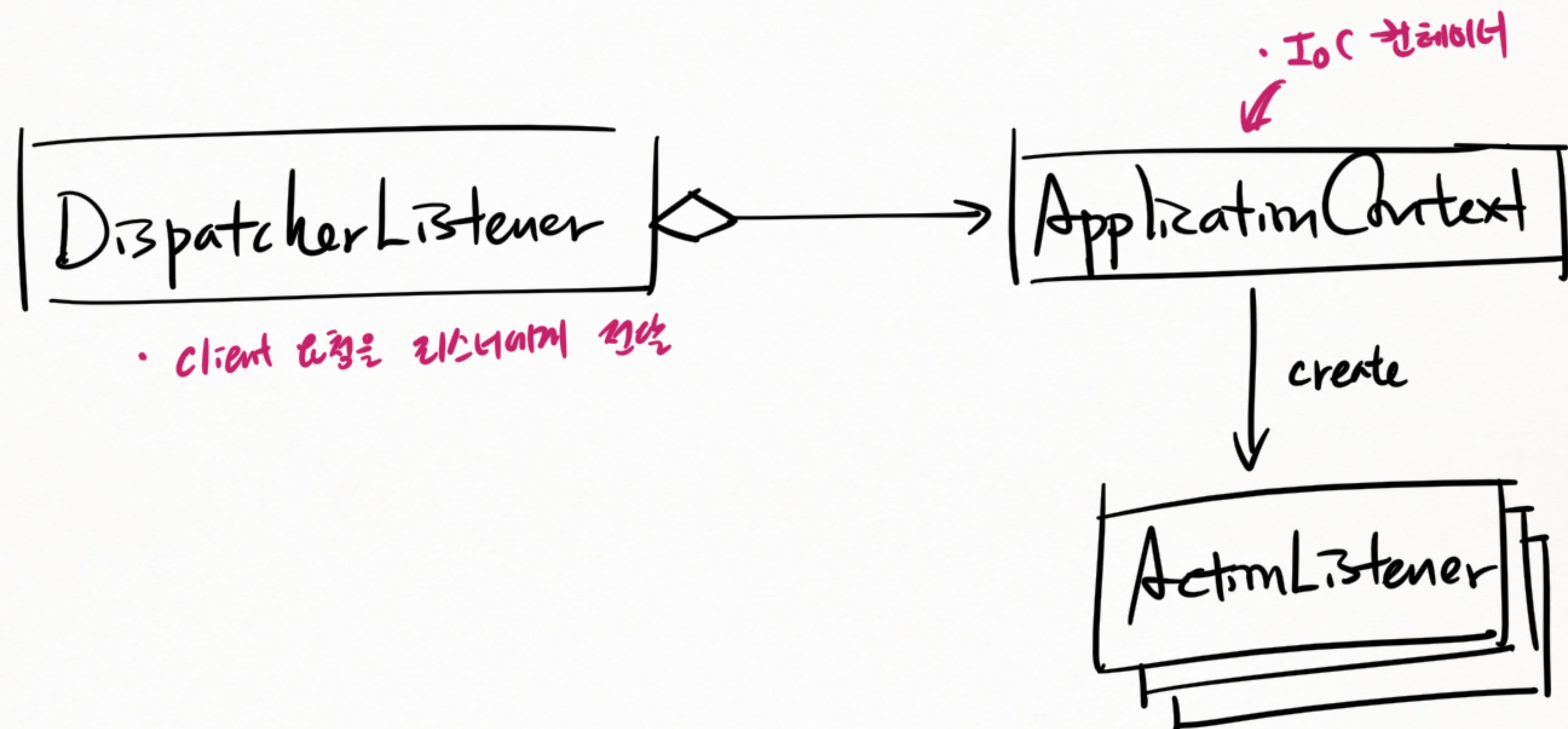
* 하는 일

① Listener 구현 준비 = IoC 컨테이너 예제

② Listener의 Facade 예제

"Front Controller"

* IoC 컨테이너 부리



IoC 컨테이너 = Bean 컨테이너

"자연-인간"

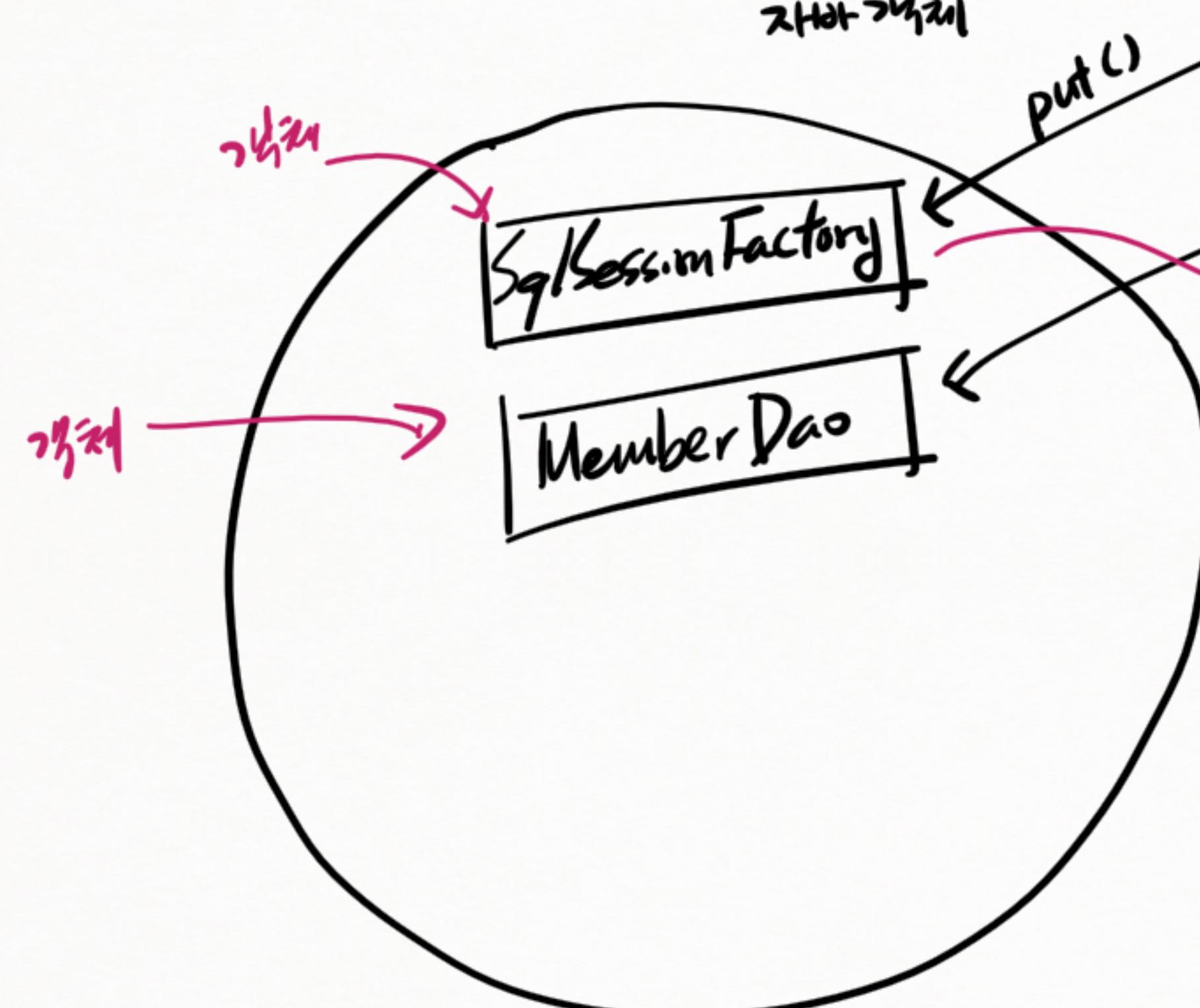
생명주기 \rightarrow lifecycle (생성 ~ 100%)

sqlSessionFactory() $\frac{2}{2}$

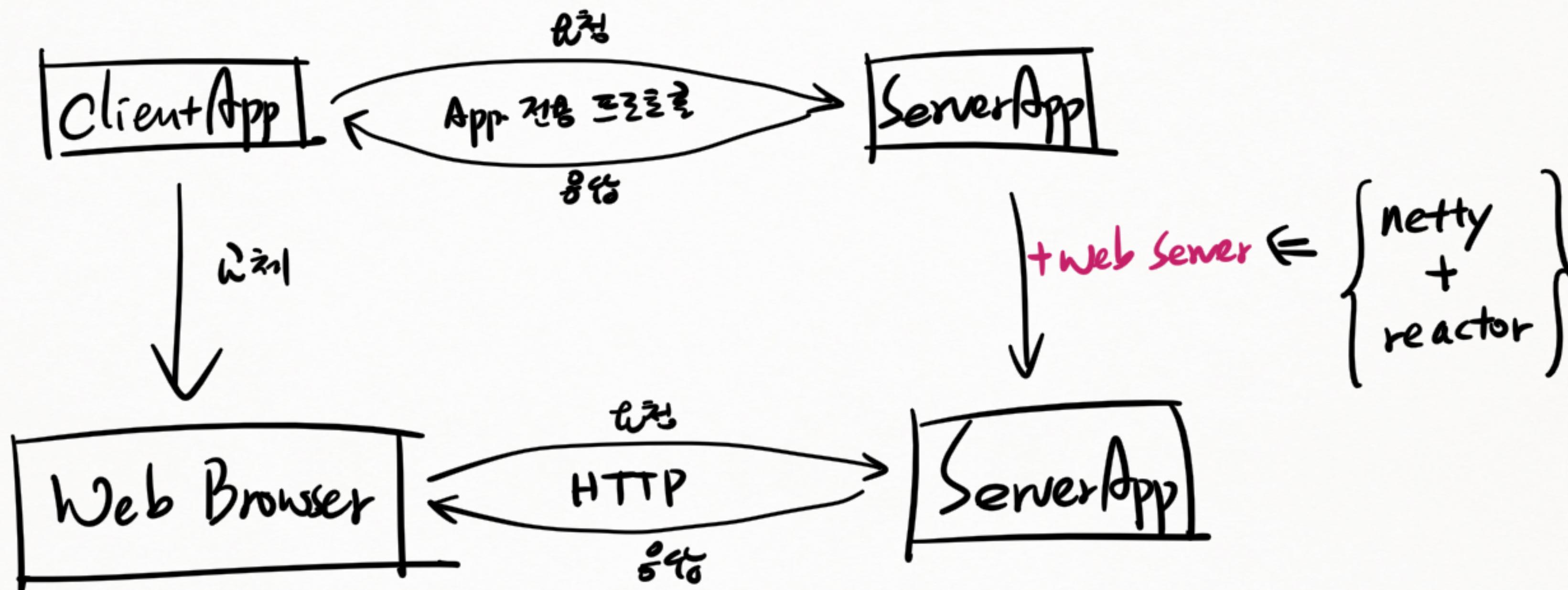
new MySQLMemberDao();

전화번호
가족

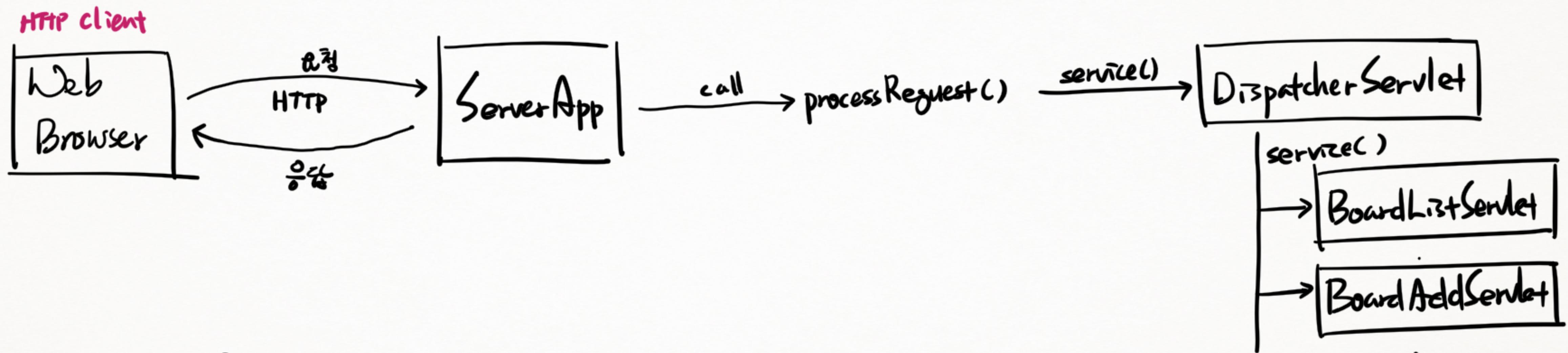
SqSessionFactory



55. 웹기반 애플리케이션



* 요청 - 응답 실행 과정



Hyper-Text Transfer Protocol

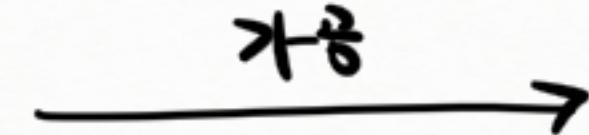
- 문서데이터 마크업이 존재
- 다른 문서와의 연결정보



HTML
Markup
Language

* processRequest () 하는 일

HttpServletRequest



HttpServletRequest

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

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

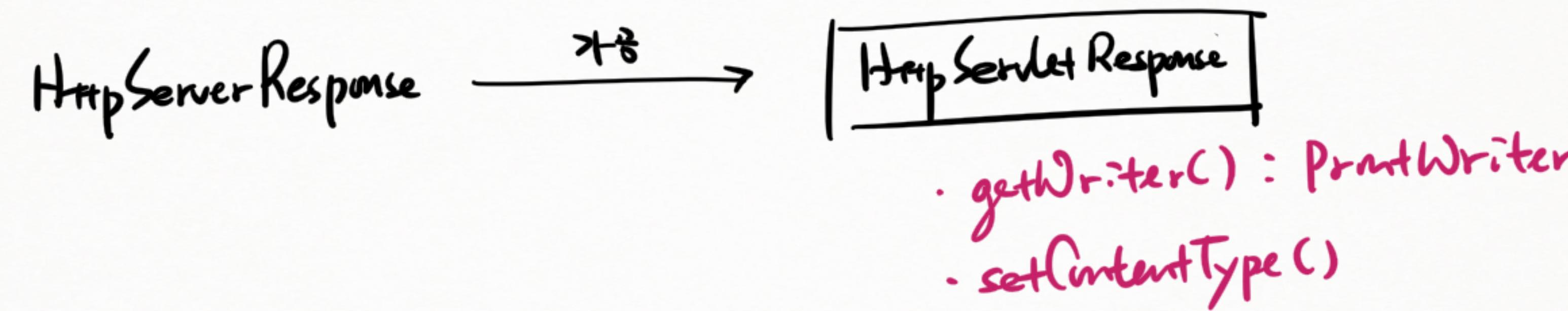
Servlet Path

↓
getServletPath()

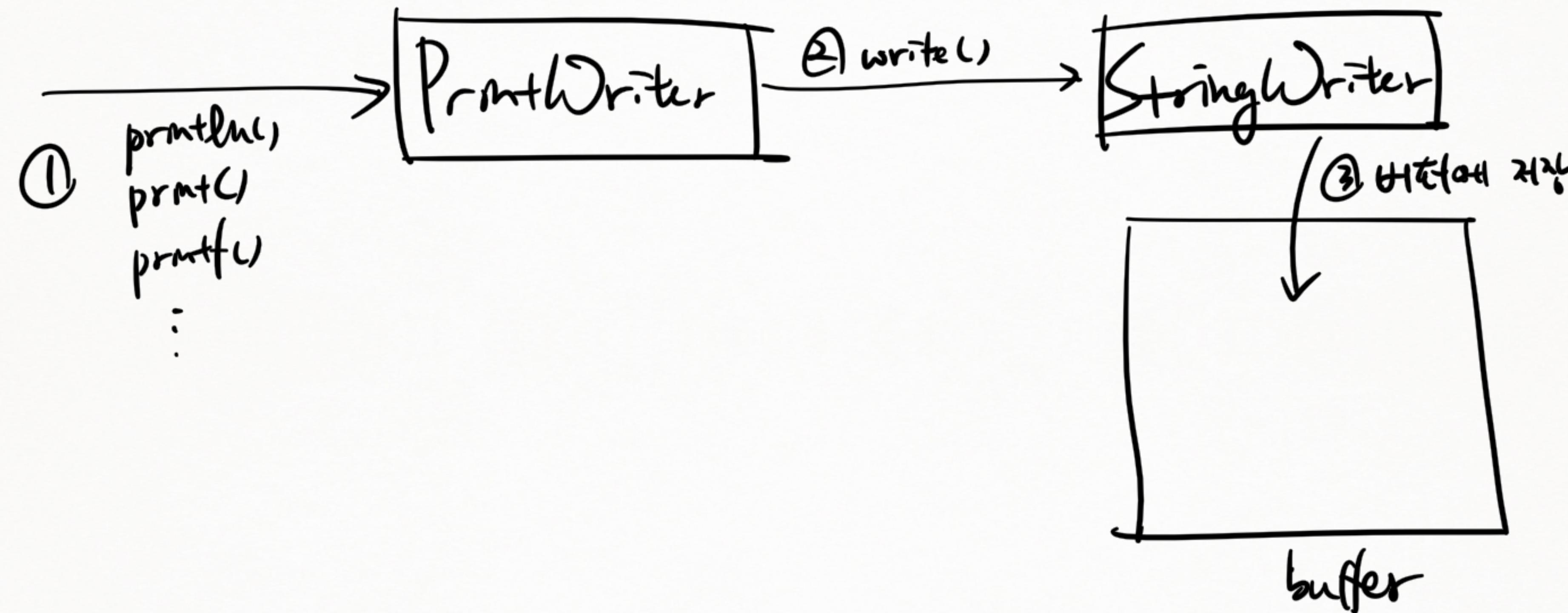
Query String

↓
getParameter()

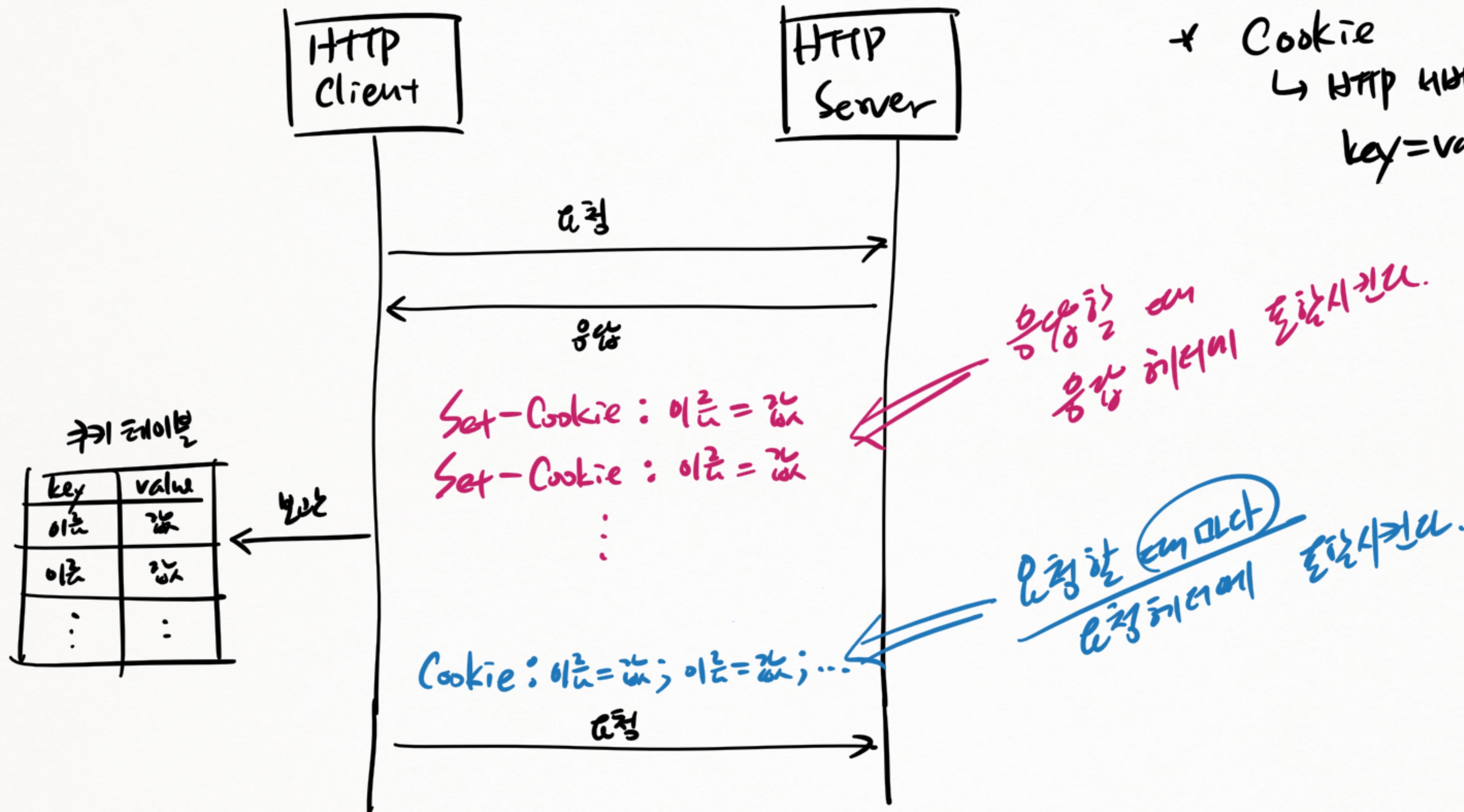
* processRequest () 하기 위한



* StringWriter의 흐름



* HTTP Client et Cookie



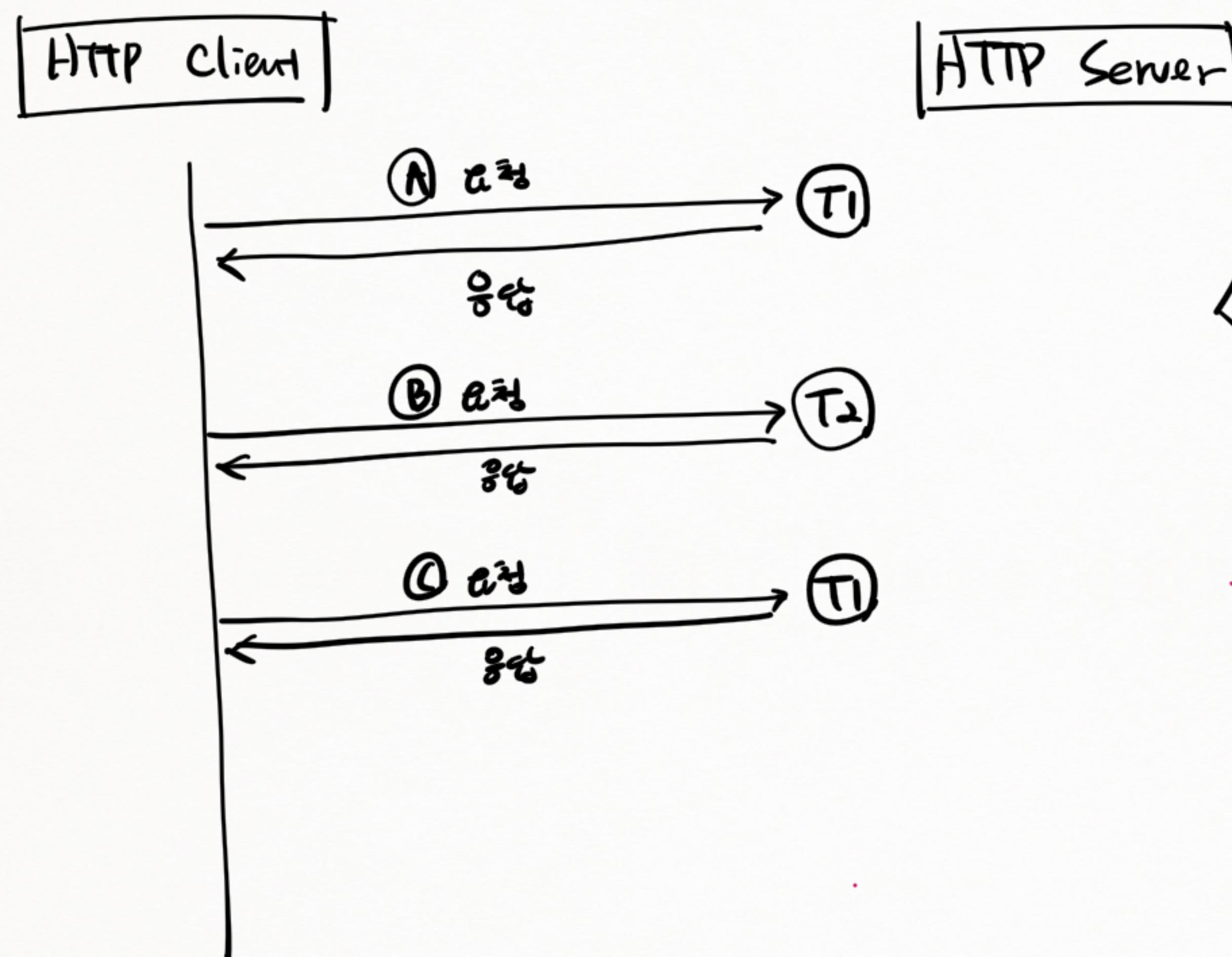
* Cookie

↳ HTTP 헤더가 HTTP 클라이언트에
key=value 형식으로 전달시킨다!



HTTP 클라이언트는
요청한 쿠키를
제거할 때는
제거할 때는
제거할 때는
제거할 때는

* 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

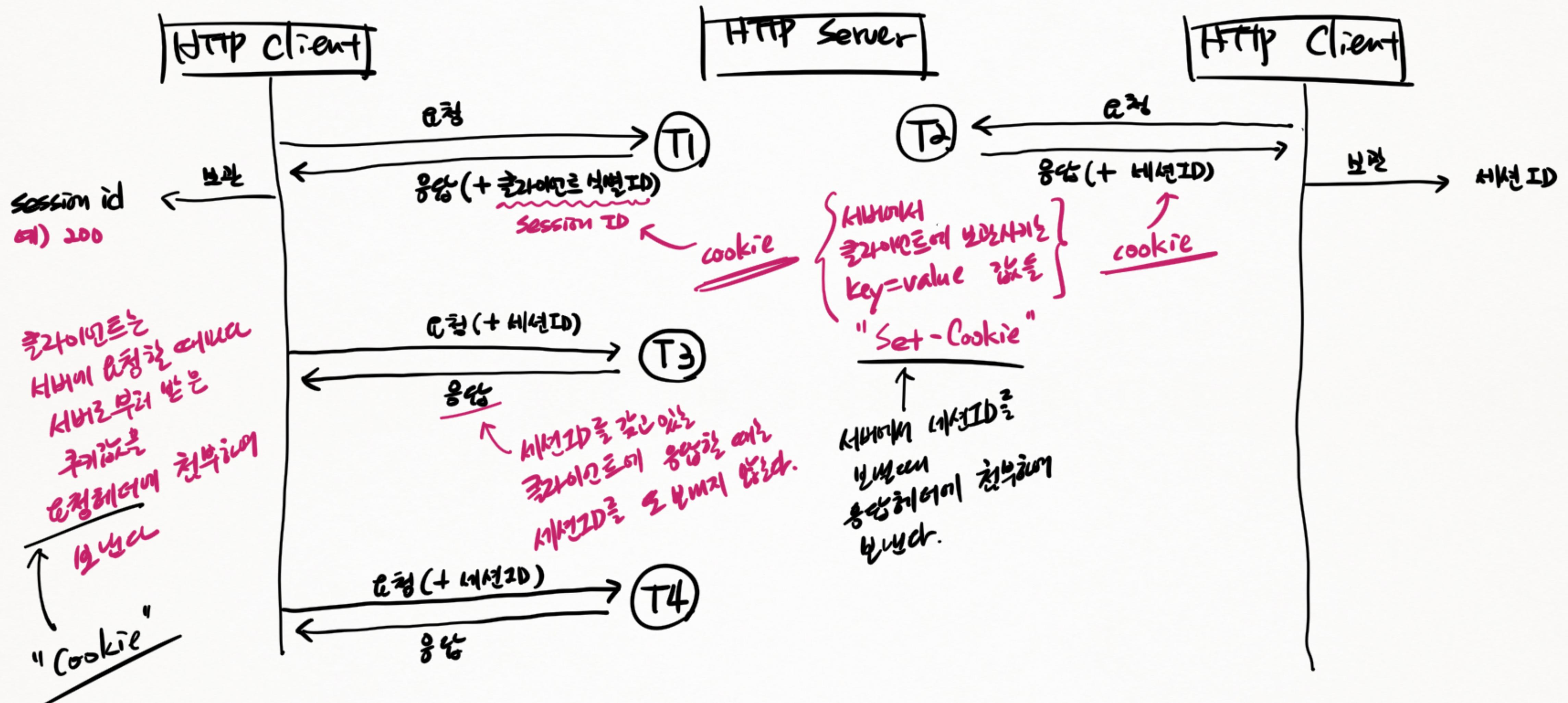
stateless
예) HTTP
연결 - 요청 - 응답 - 끝기

연결 - 요청 - 응답 - 끝기

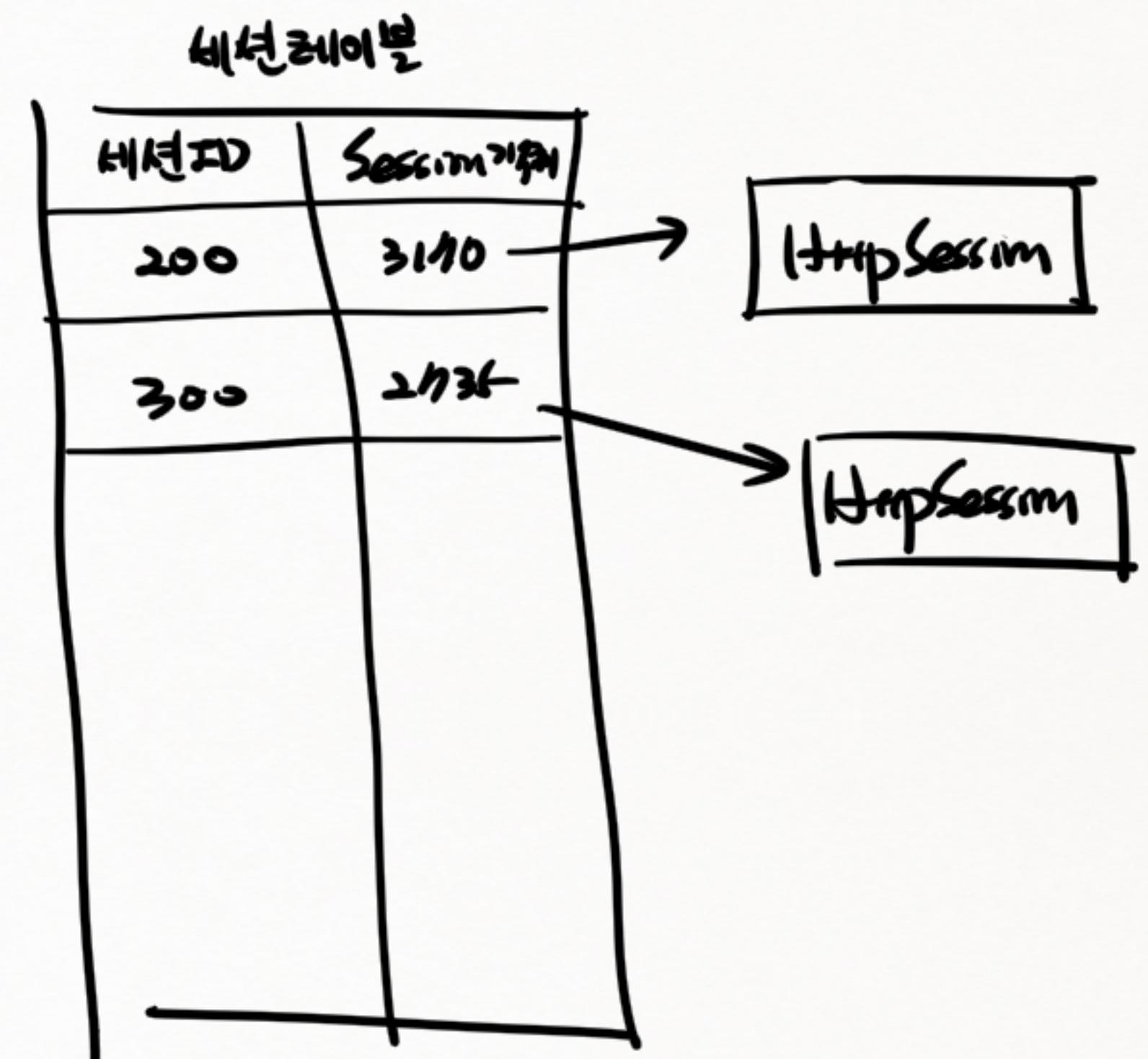
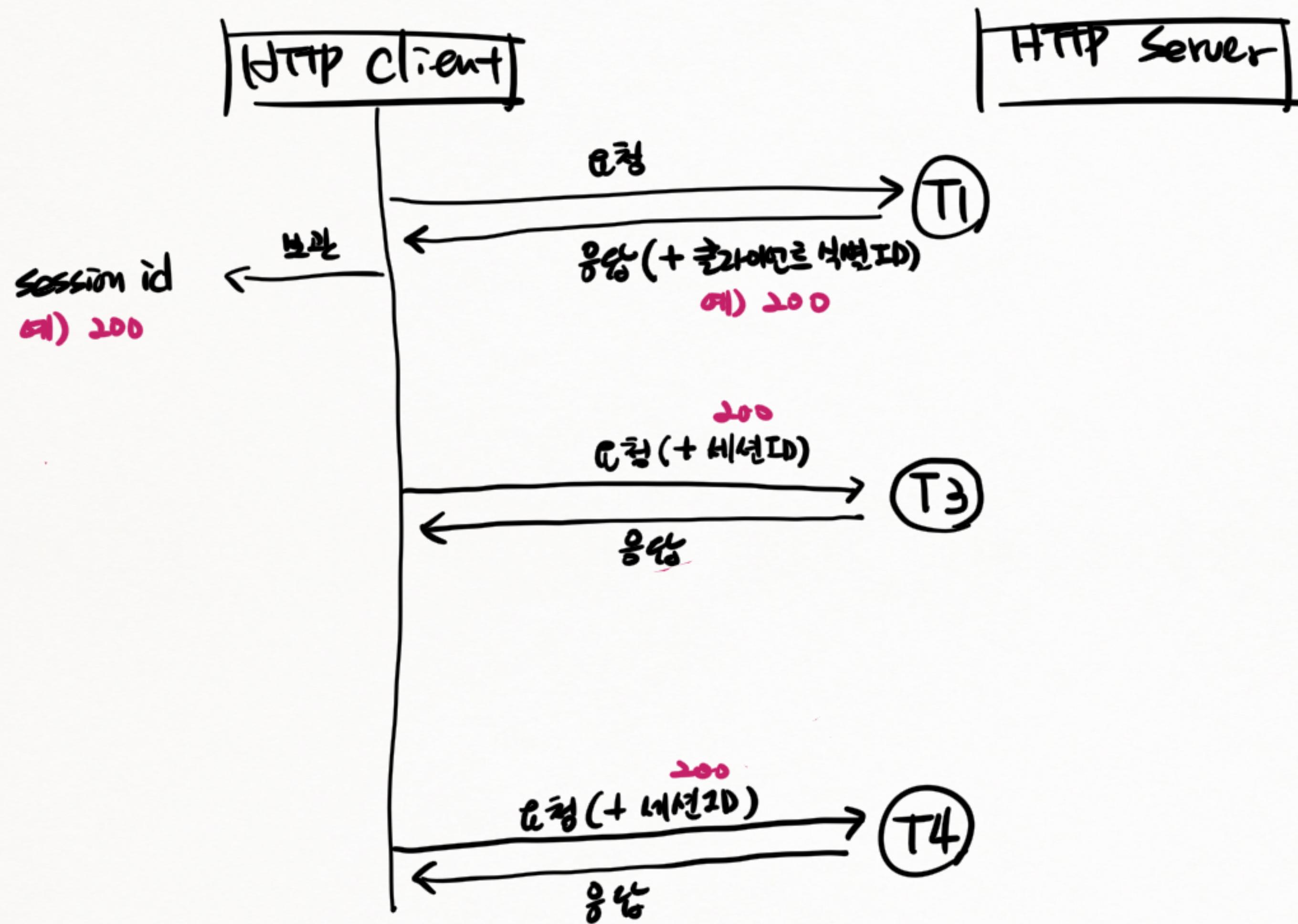
* HTTP 통신

- stateless 방식으로 통신한다
- 요청할 때마다 새로 연결한다
- 요청을 처리하는 스레드가 다른 수 있다
- 요청과 요청 사이엔 데이터 공유 불가!
 - ↓
 - 스레드에 보관해보야 사용된다.
 - HttpServletRequest 가 요청과 함께 사용된다.

* 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 뒤집은 &를 갖다.

/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 경우 Entity Body

"Entity Body"
(Message Body)

* HTTP 요청 방식 - GET vs POST

GET 방식		POST 방식
Binary 데이터 전송 크기	제한된다 64KB, 8KB 그러나 크고 데이터의 때문에 제한되는 파일 전송이 제한된다.	제한없다
문자 데이터 전송	평가. 단 Base64로 인코딩하면 문자로 변환되어 전송할 수 있다.	요청 데이터 끝 부분, message body 부분이 첨부하기 때문에 전송 가능!
보안	유형별로는 보안을 개선해 보았지만 ↓ 파라미터 값이 노출	파라미터 값이 부분이 첨부되는 ↓ 사용하기 험しい
성능	URL과 데이터를 함께 보내야 하는데 제한 ↓ 조회량 처리	<ul style="list-style-type: none">대량의 데이터 전송비슷한 데이터 전송URL에 파라미터를 넣기로 삼지 않을 때