

## 57. 파일 업로드

\* 텍스트 프로토콜

POST /board/add HTTP/1.1 ↗ request line  
Content-Type: multipart/form-data; boundary=----Webxxxxx  
↑  
<form enctype="multipart/form-data" ...>  
:  
<input type='file' ...>  
:  
</form>

자바에서 구동 문자열

header

↑  
→ ← 파일-문자열 <boundary>  
→ ← 파일  
→ ← 파일-문자열  
:  
} message-body      ~~O=OK O=O~~

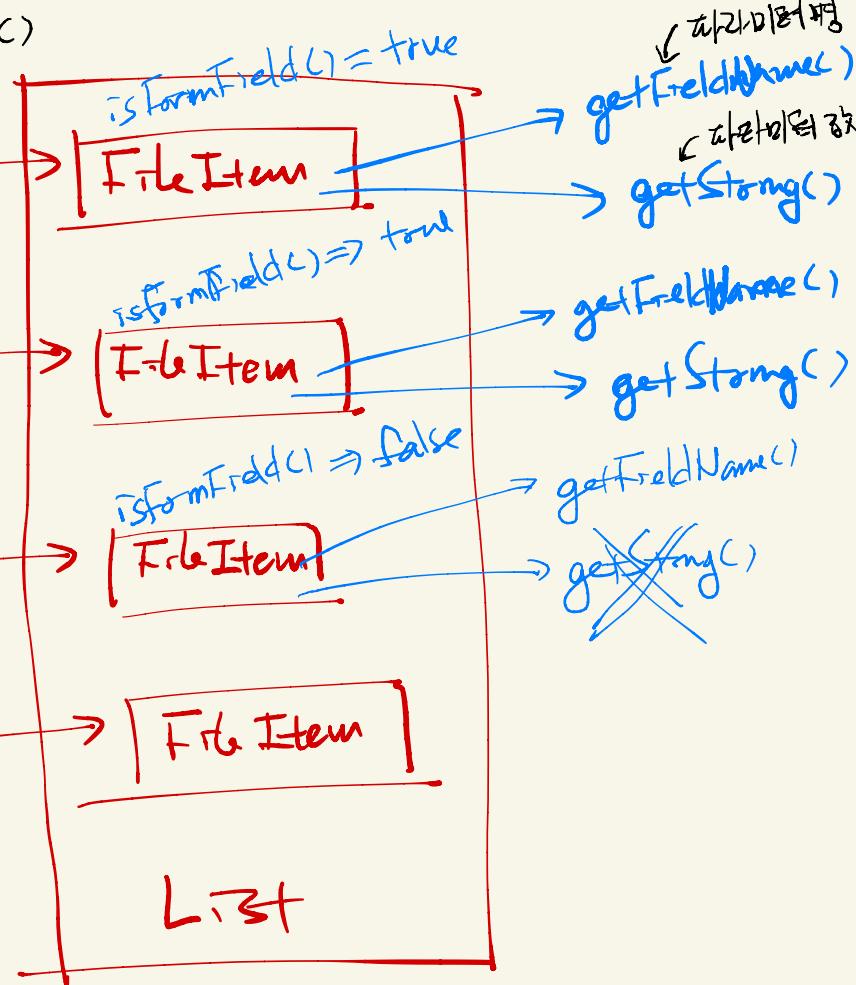
\* 요청 파싱  $\Rightarrow$  parseRequest()

aaaa  
name = "title"

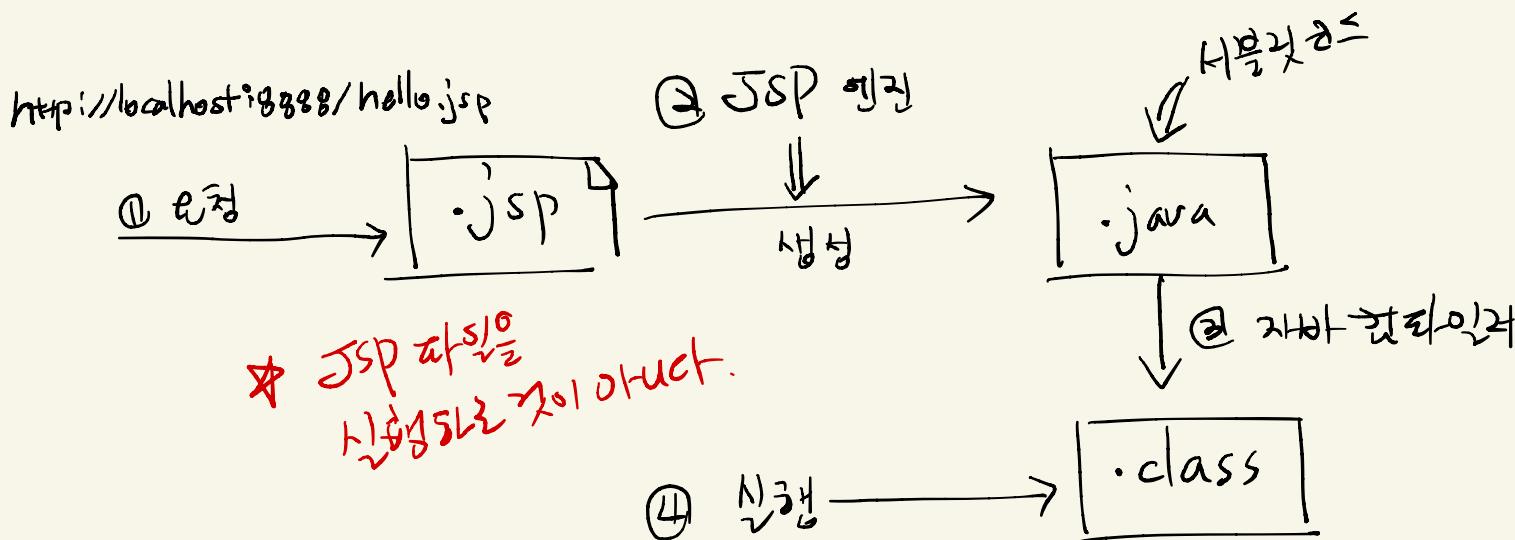
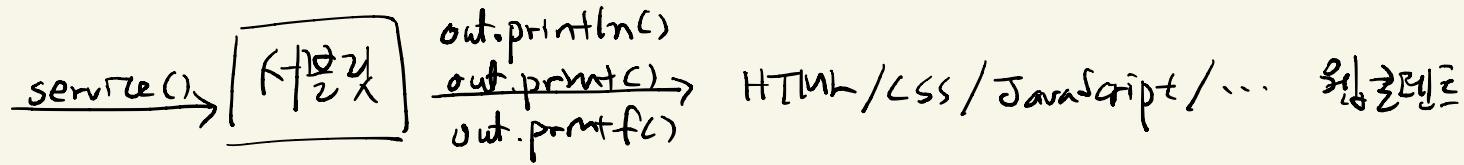
bbbbb  
name = "content"

dd  
name = "files"

ee  
name = "category"



\* JSP → 서블릿 코드 자동생성



\* JSP 자원 평로

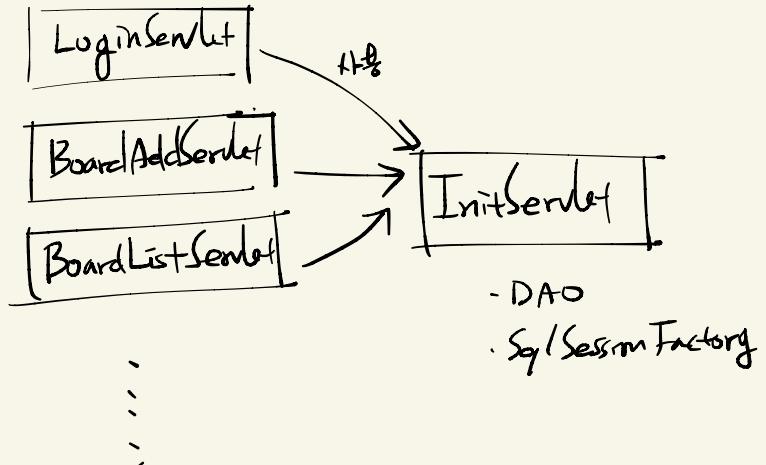
src/main/webapp/\*.jsp

↓ 자동생성

.. /temp/\*:java

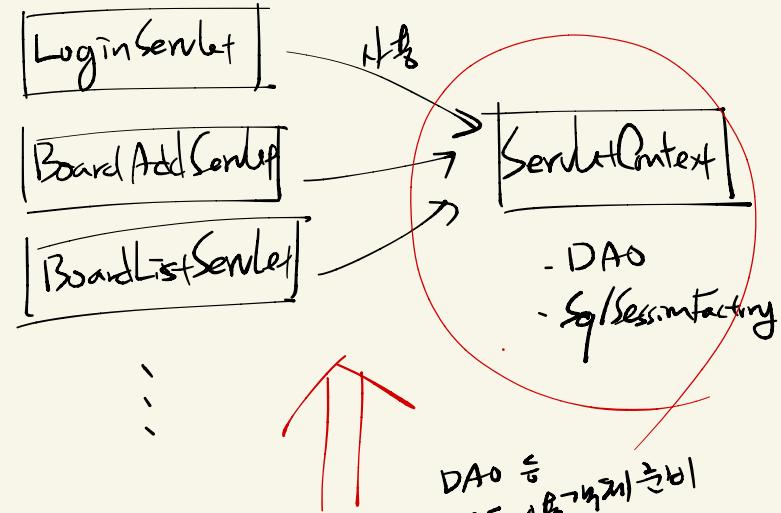
\* 63. ⇒ CTX ServletContext 보관/수

① 전통



\* 예전에  
- ex) static DAO를  
InitServlet에 주면  
그거 쓰는거.

→ ② InitServlet 종속성 주입



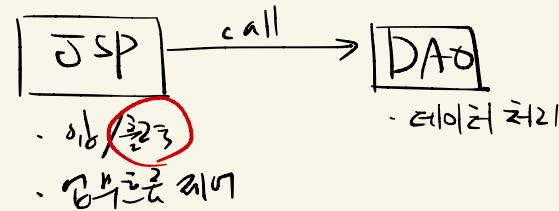
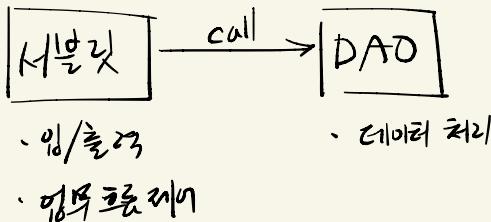
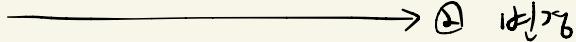
DAO 등  
공통 DAO가 되도록 한다!

ContextLoaderListener

↓  
<interface>  
ServletContextListener

## \* 64. MVC 모델

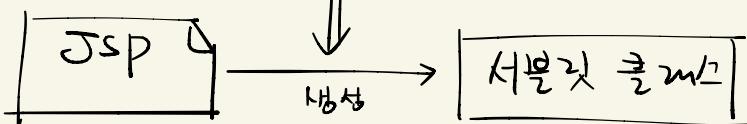
① 이전 방식



\* JSP

서블릿 풀러스  
Generator  
||

JSP Engine



- 템플릿 데이터 → 출력문
- JSP Action tag → 처리코드
- script tag element → 처리코드
- :

} ⇒ 풀제이지로 생성해야 하는  
개발 부담을 경감!

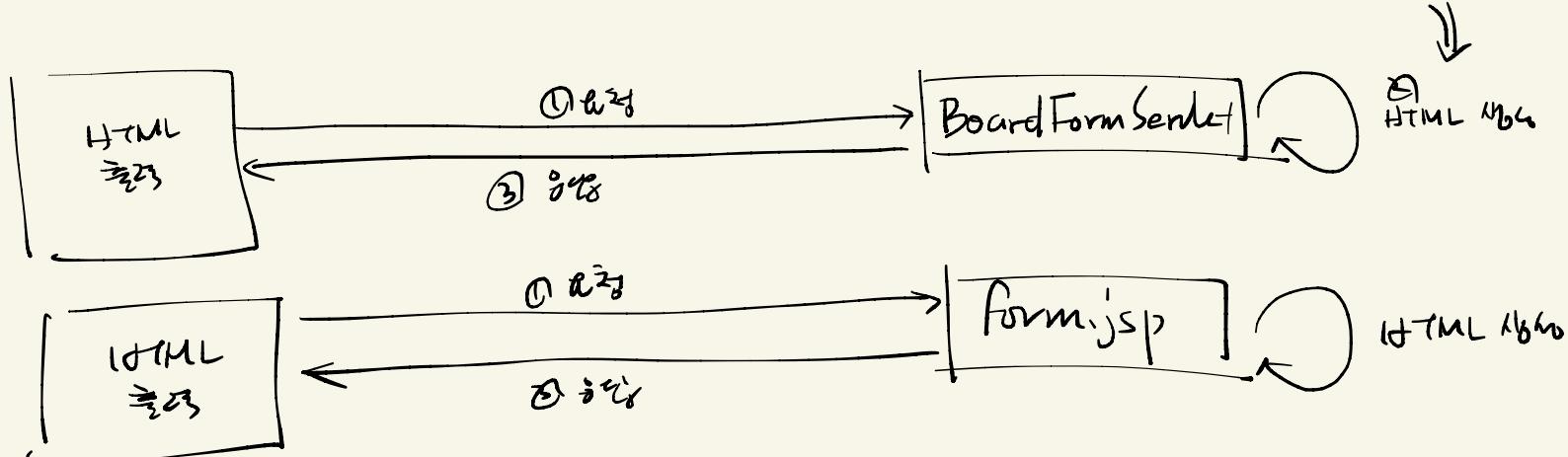
\* 이런 방식으로 차이점

- JSP 기술 사용해서 코드를 훨씬 더 간단하게

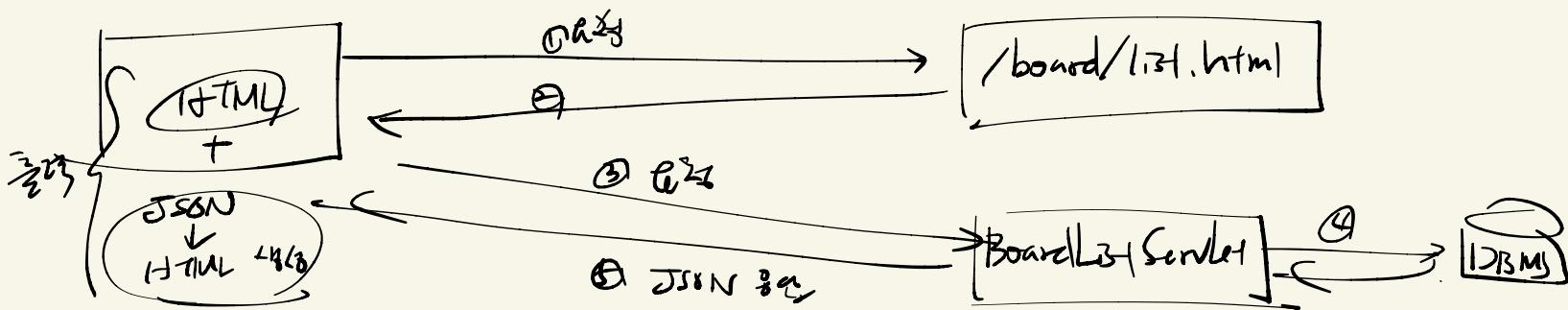
↑  
제작 시간  
줄여 놓다.

## 서버 렌더링 (rendering) 과정

서버 렌더링  
HTML 흐름



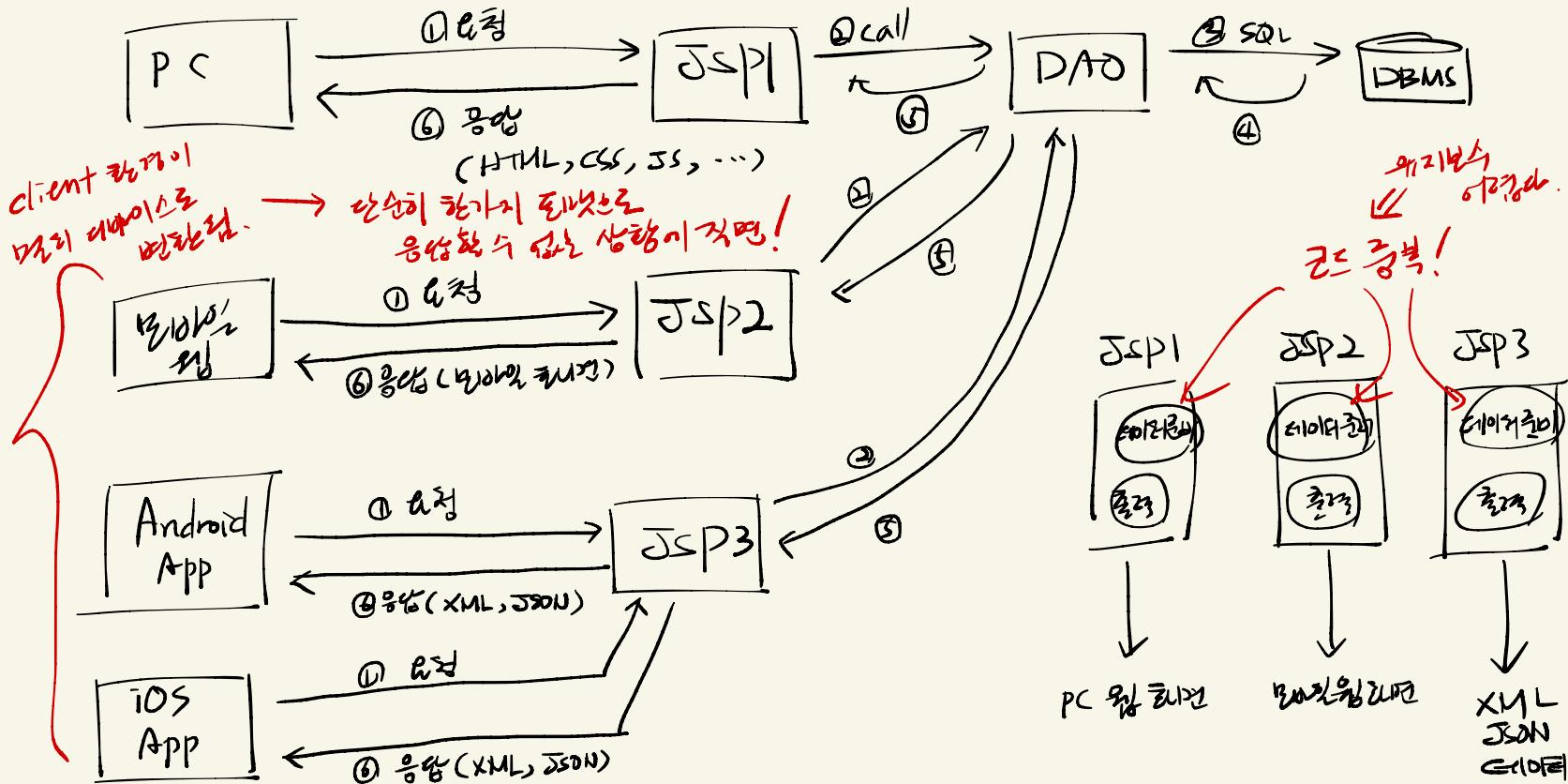
## 클라이언트 렌더링 과정



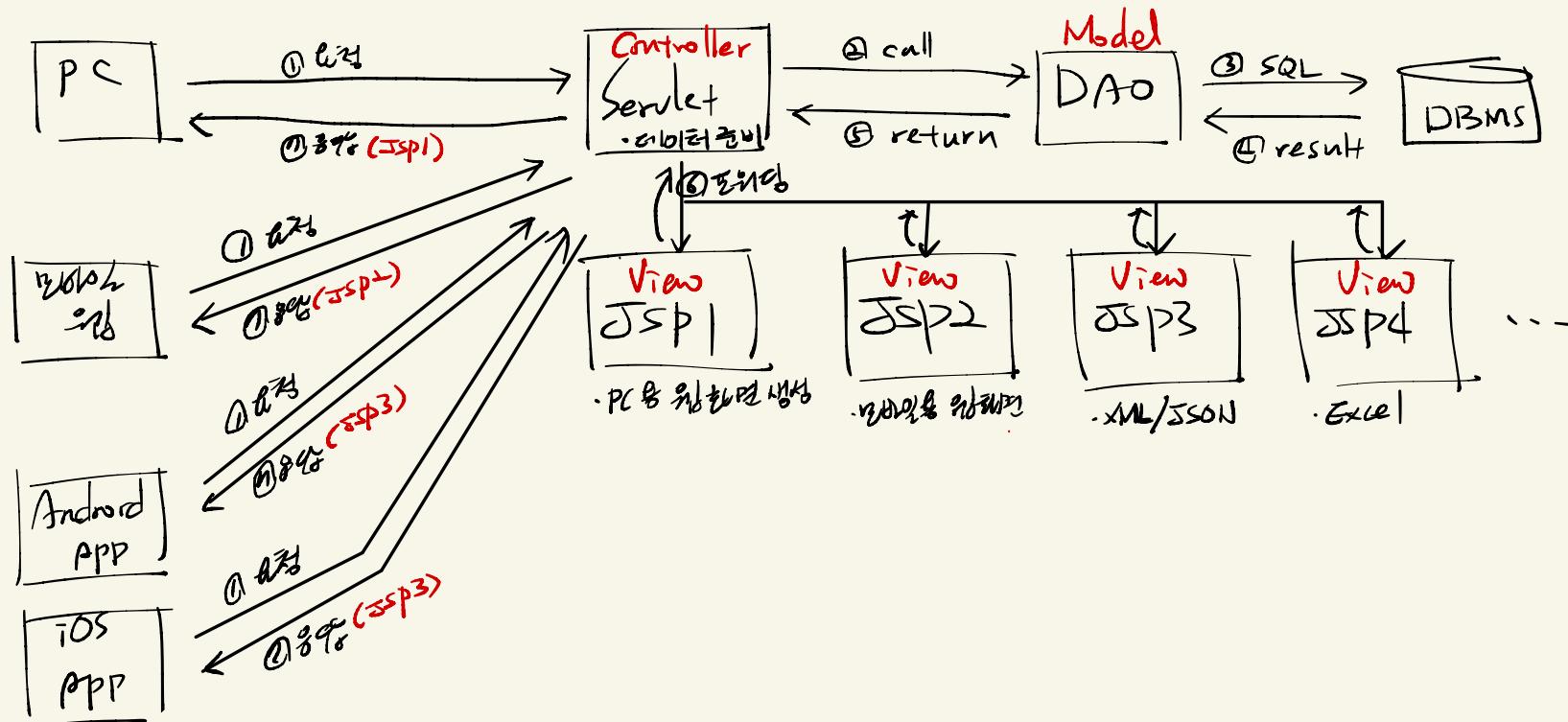
# 65. MVC 구조 2

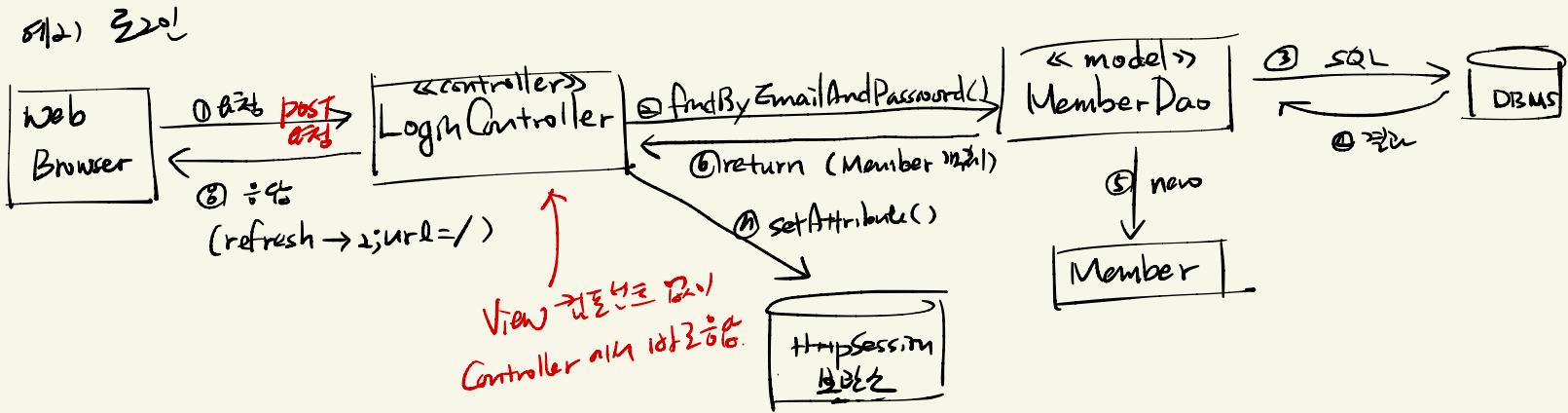
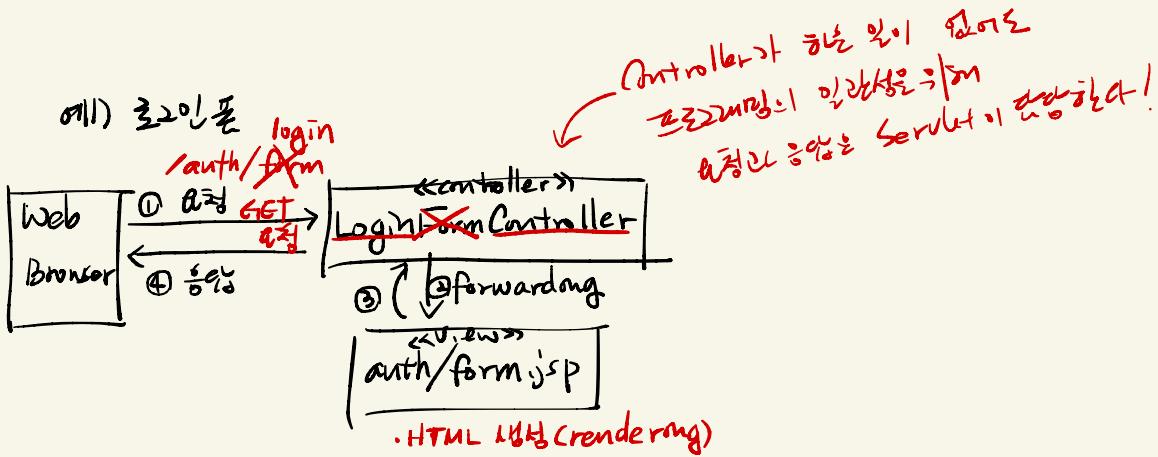
① 이전 방식

②) 개시는 목적  
↓

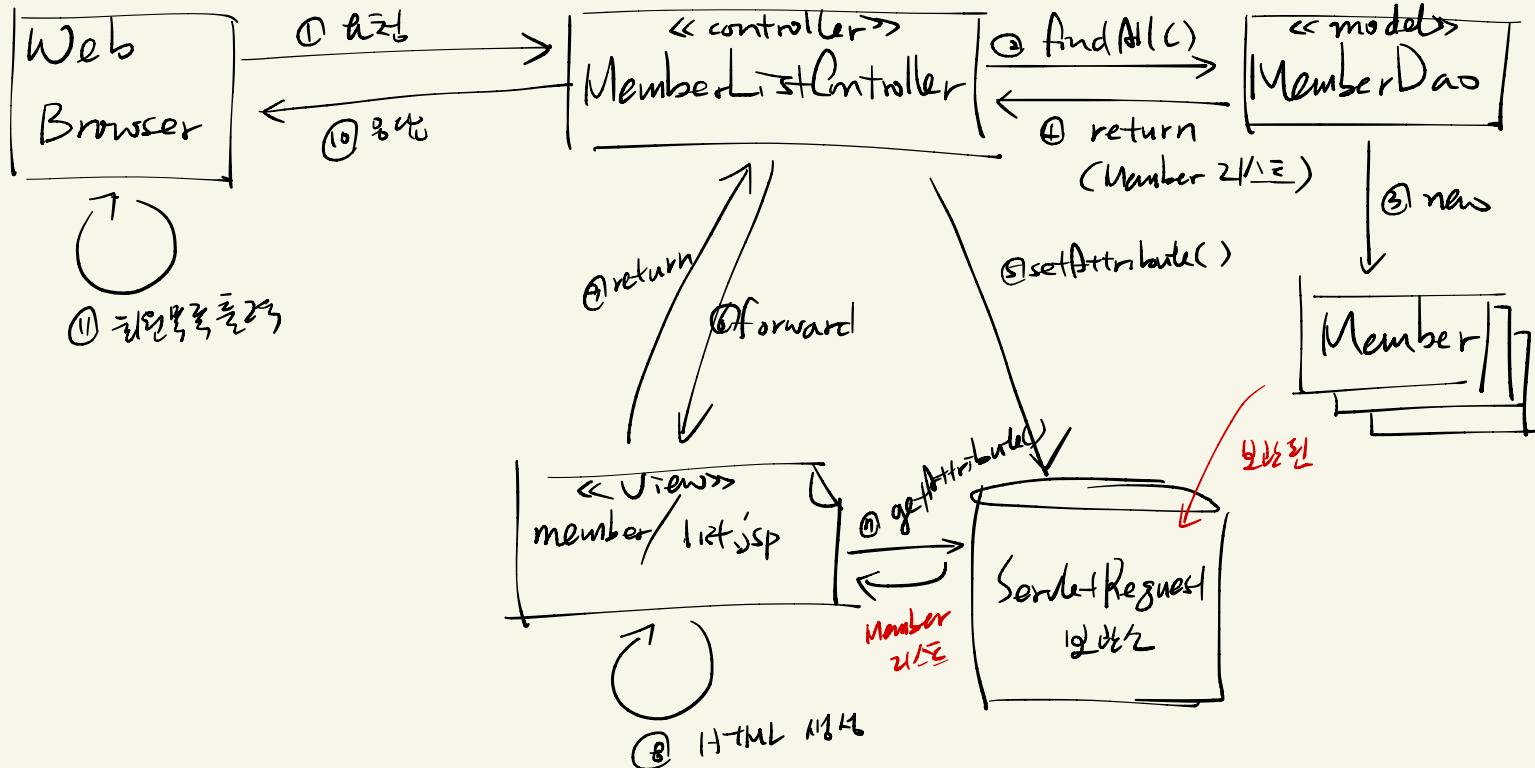


② 인터랙션 흐름



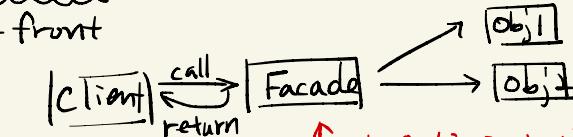
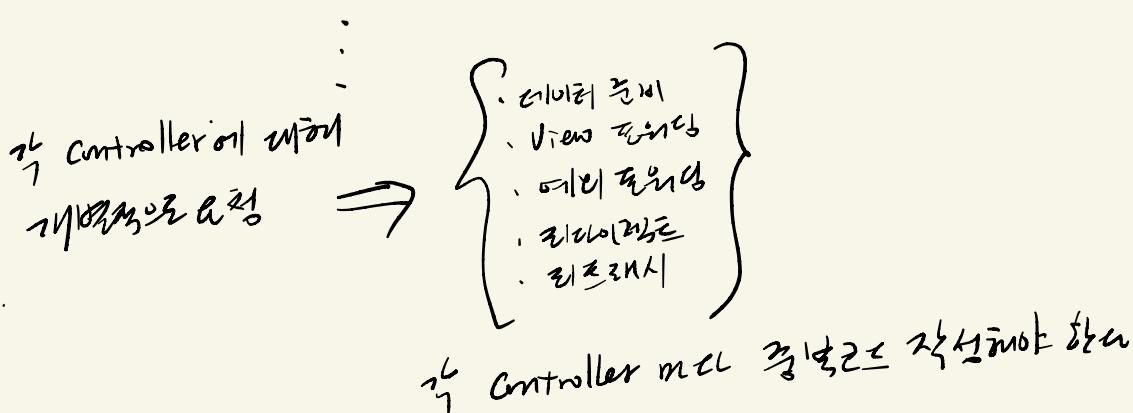
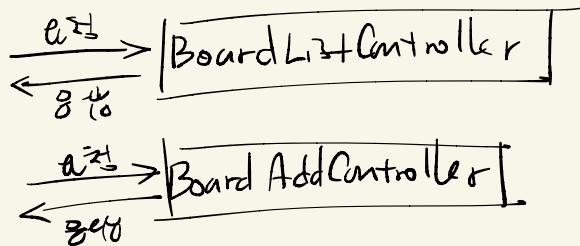


### (013) 회원 목록 조회



## 66. Front Controller 패턴과 GOF 패턴

① 이전 108시



Facade  $\approx$   
= front

작업을 수행하기 위해  
여러 객체를 어떤 순서로  
어떻게 사용해야 하는지에  
관심이 있다 = "encapsulation"  
(내포화)

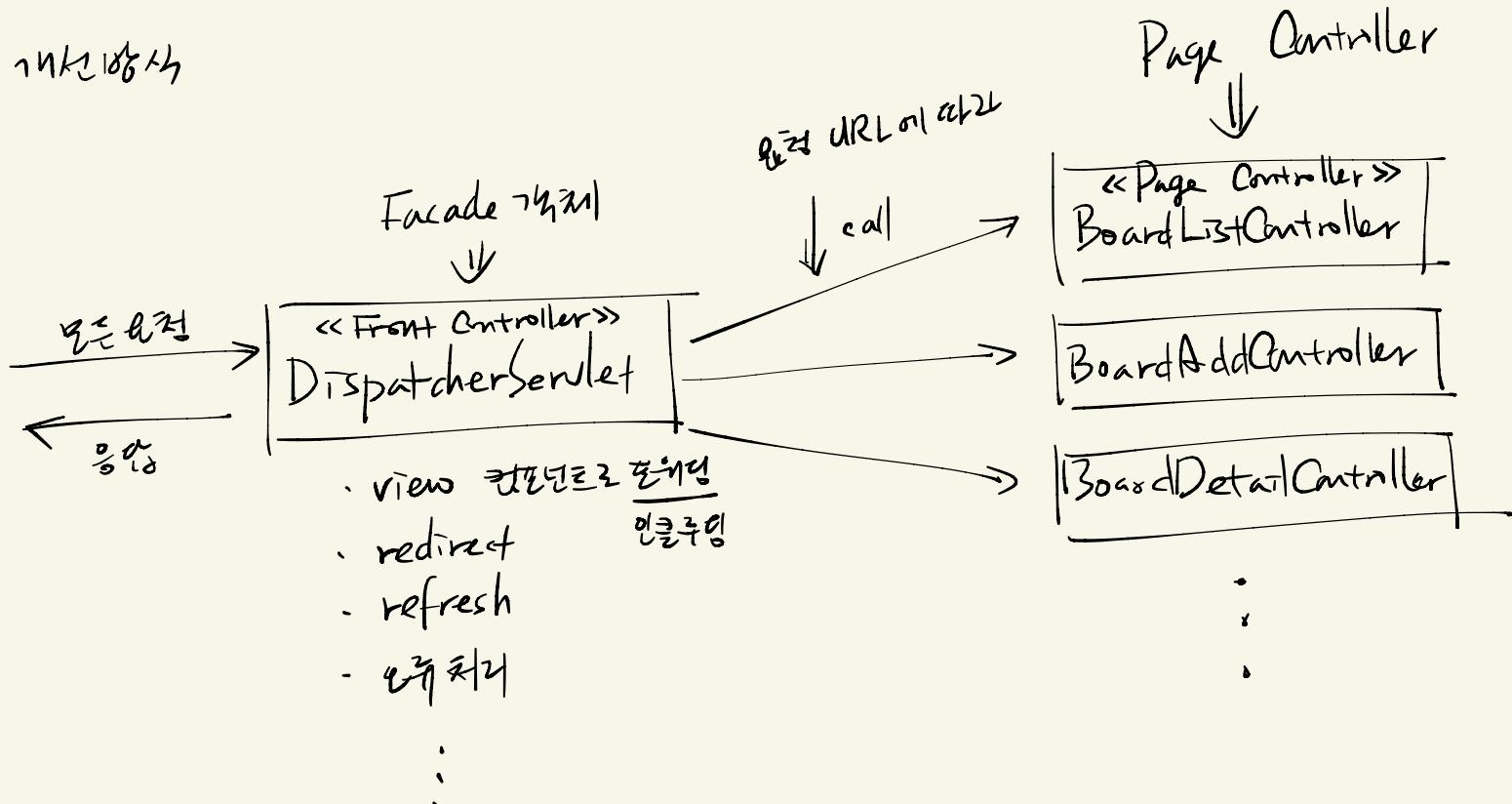


호출자 입장에서는  
복잡한 로직(실행흐름)을  
알 필요가 없기 때문이다

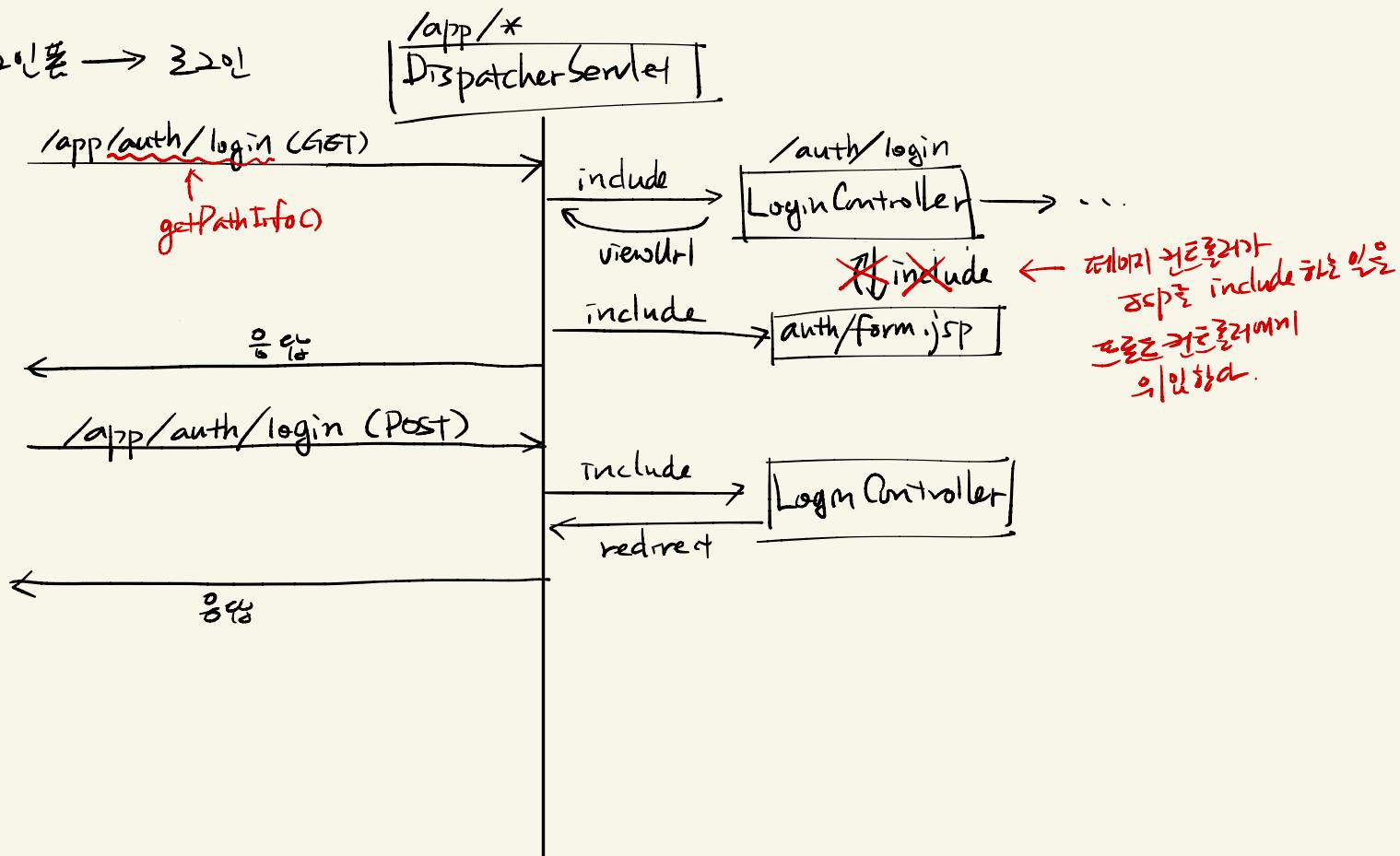


만약 작업의 흐름이 변경  
되었을 때 편집하기  
방법을 까치지 않도록

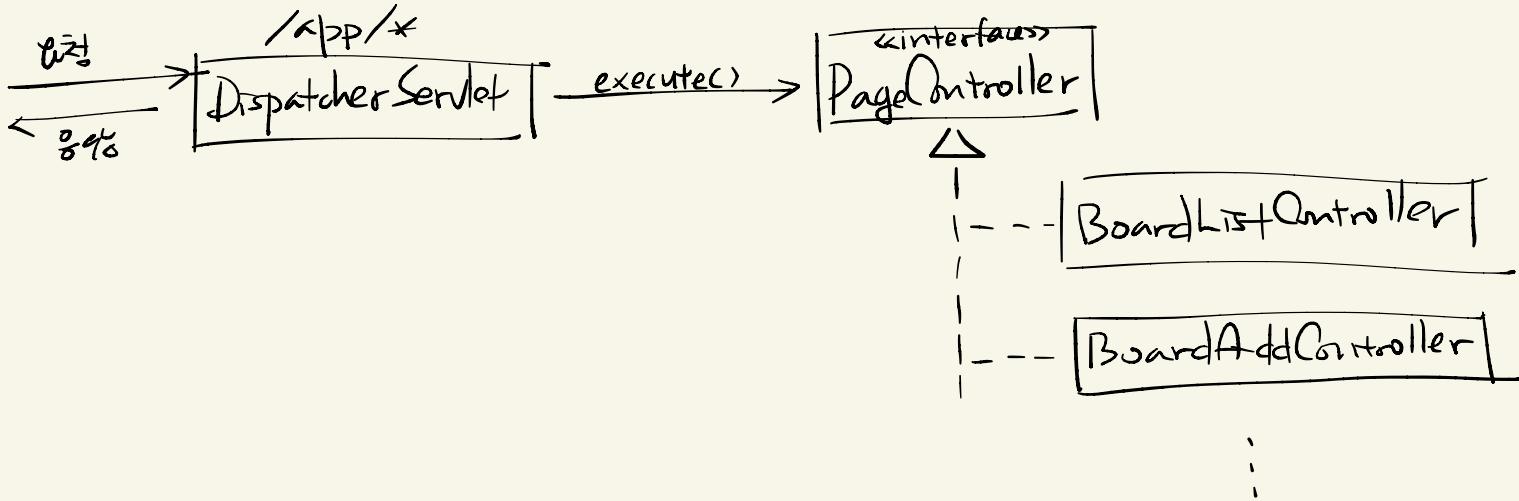
② 인터페이스



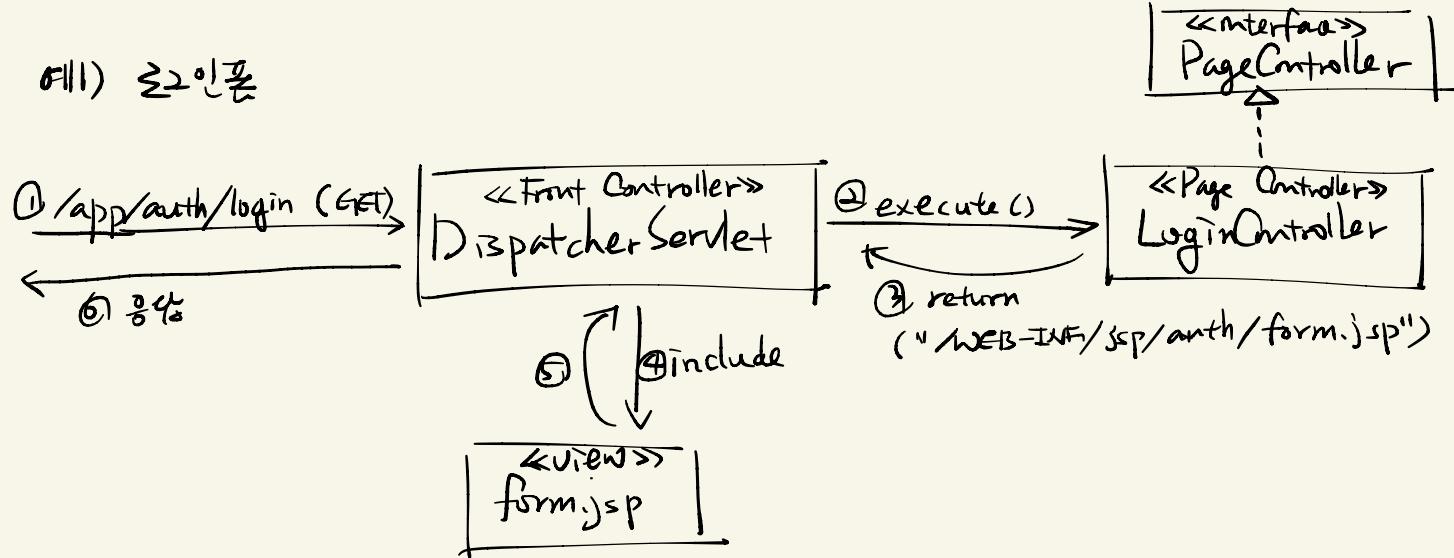
0911) 3201 裏 → 3201



## 67. 서버이지 컨트롤러를 POJO로 전환 (Plain Old Java Object)

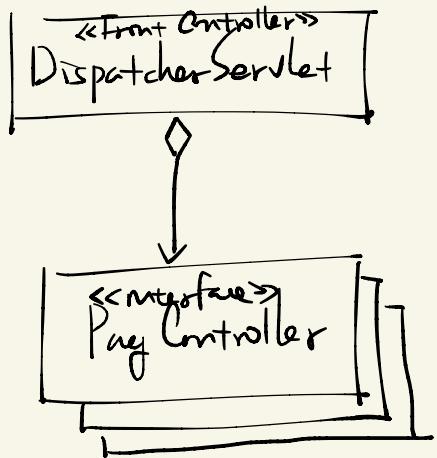


011) 登录逻辑

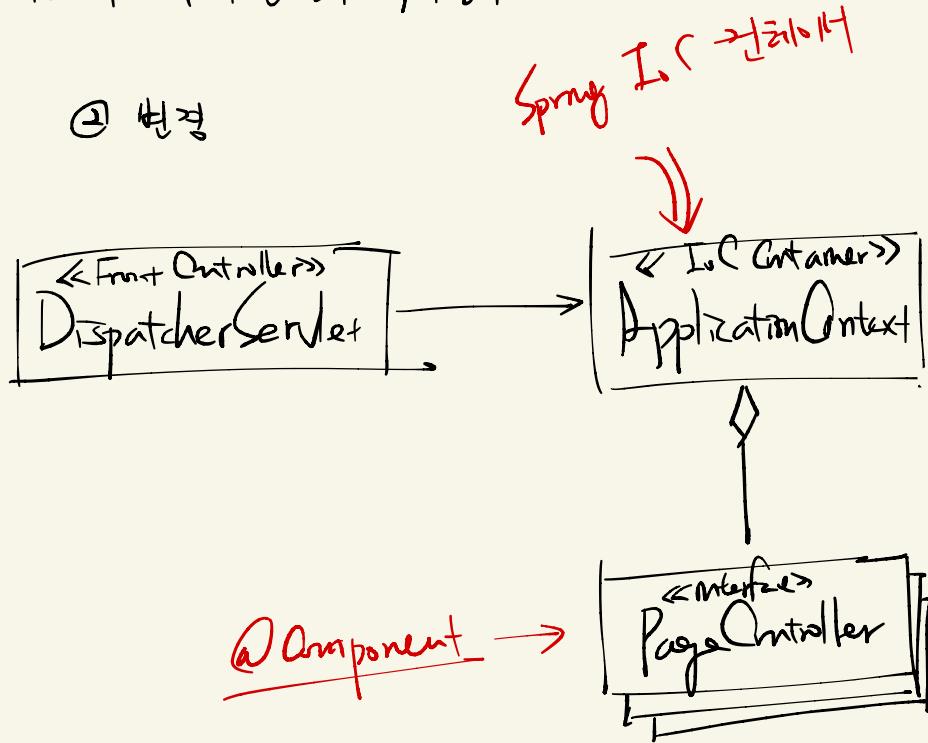


## 68. Spring IoC → 컨테이너로 이용해서 데코디→인트로드 가기하기

① 이전 방식

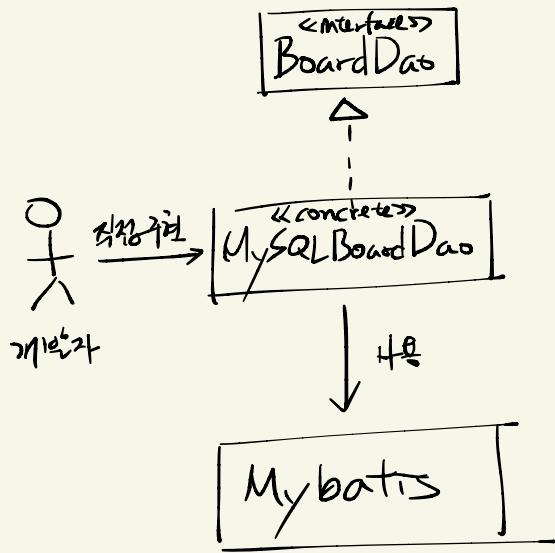


② 변경

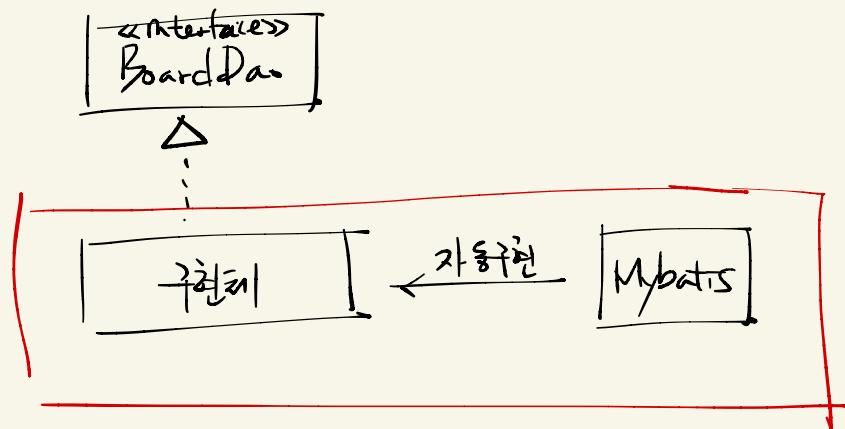


## 69. Mybatis + Spring IoC 커스터마이징

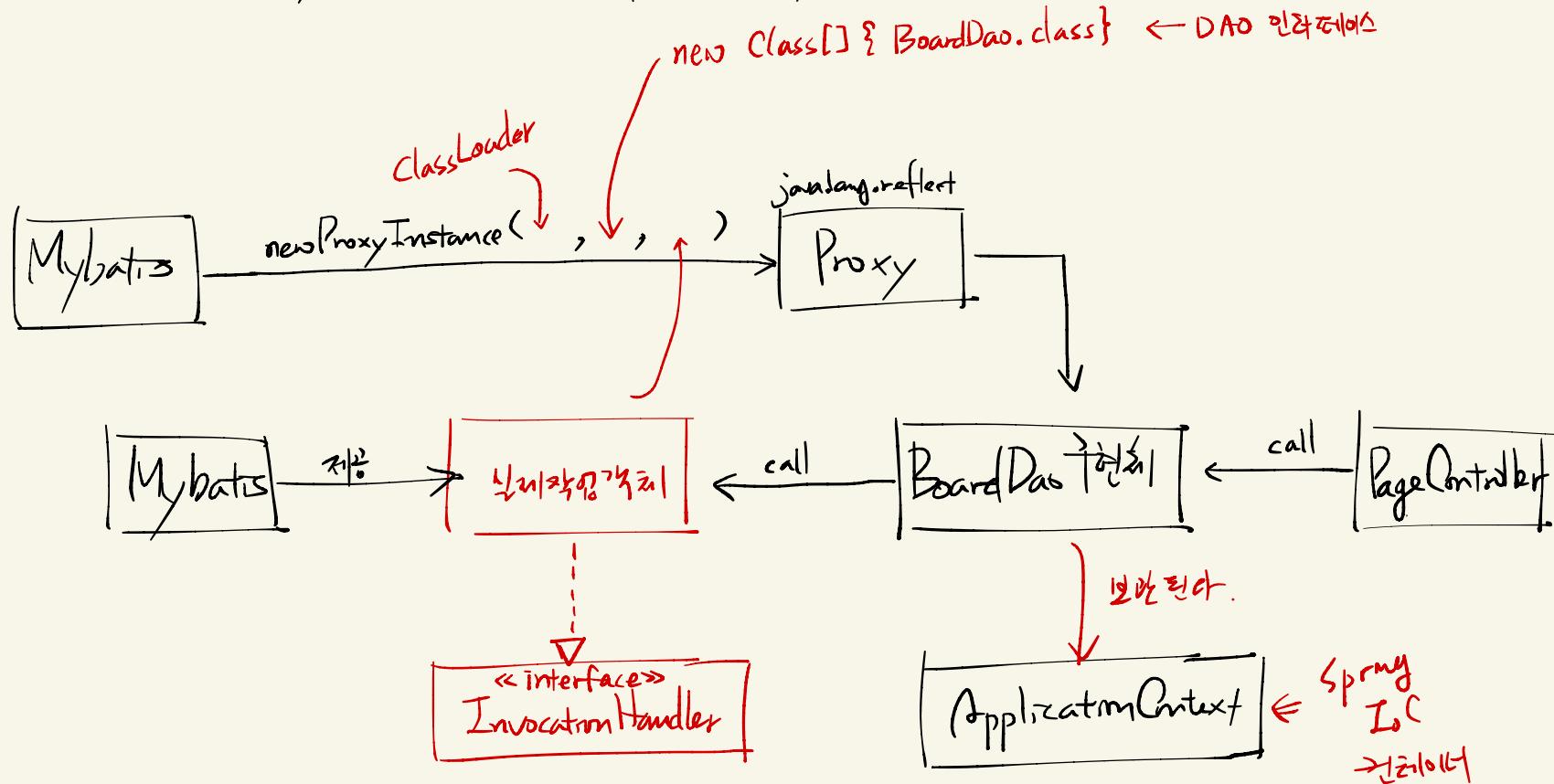
① 디자인 패턴



② 변경

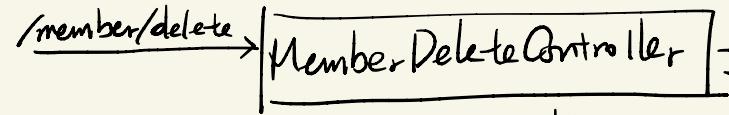


\* Mybatis → DAO 구현체로 자동 생성해줄 원리

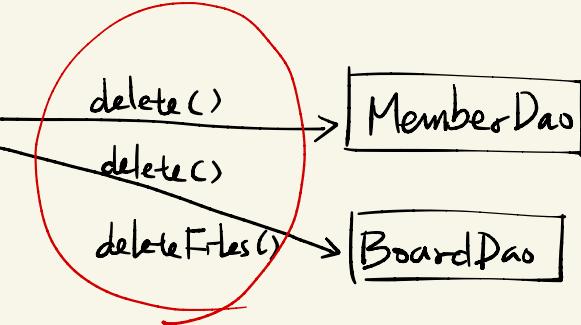


## 10. 서비스 커스텀으로 토이

### ① 이전 방식



- 요청 데이터를 가공
  - a) String → int  
String → Date  
String → Member
  - MemberDao → 파일로 저장
- 응답 데이터를 가공
  - a) DB 객체 → HTML  
" → JSON  
" → XML  
" → Excel



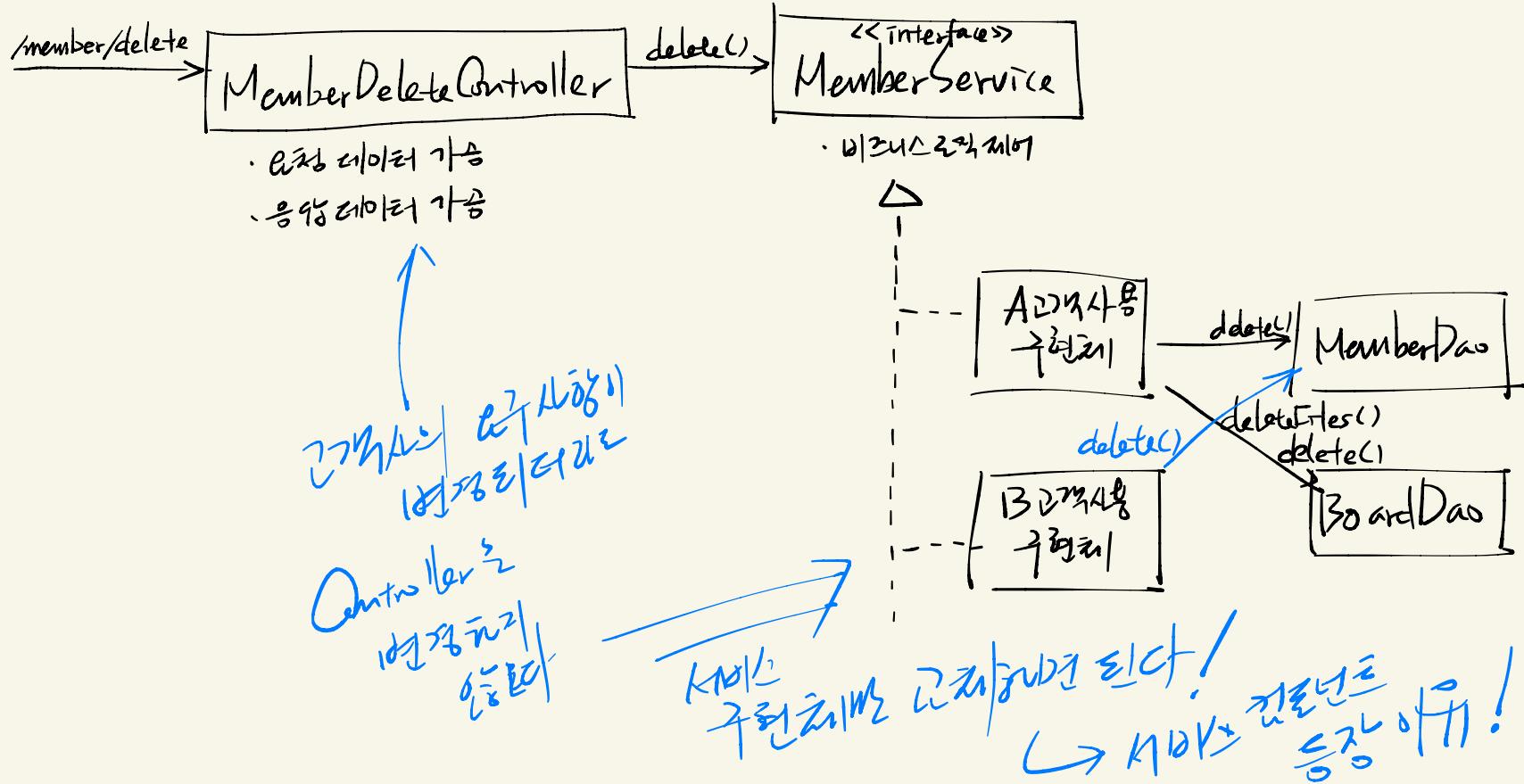
트랜잭션으로 묶여 다룬다.  
회원 삭제시 관련된 데이터도 함께 처리된다 } DAO는  
회원 삭제 }

고객사 팀이 컨트롤러를  
작성해야 한다  
↓  
유저 팀은  
작성해야 한다.

A 팀 → 회원과 관련된 모든  
데이터 쌍을 원하는  
B 팀 → 회원 데이터만  
원하는 경우.  
→ 문제점! DAO는 모두 통일

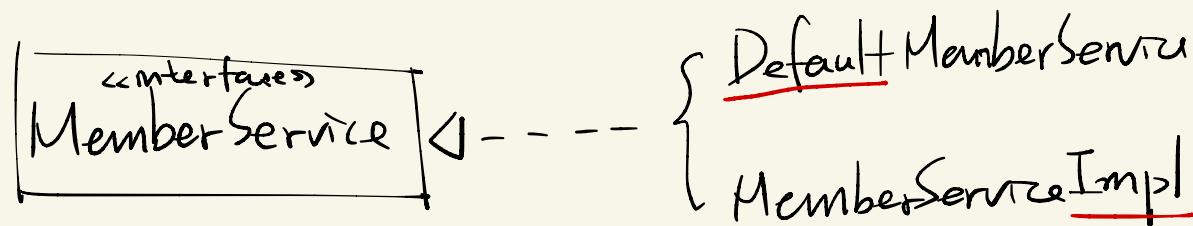
문제?  
- 고객사 팀이 컨트롤러를  
작성할 수 있다

## ② 디자인

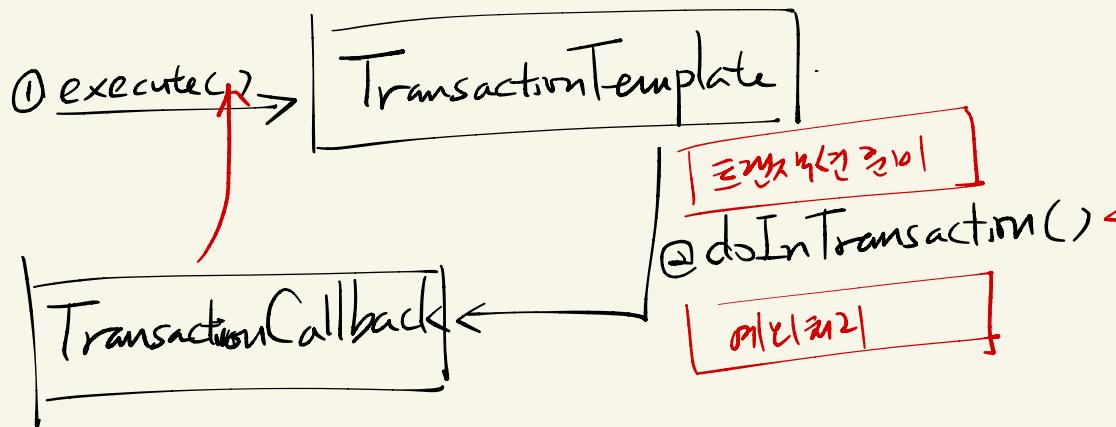
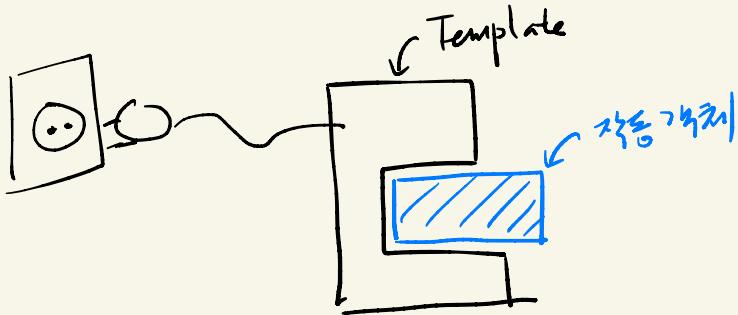


\* 서비스 → 컨포넌트 예

클래스도 서비스 구현체이거나 예



# 11. TransactionTemplate များကို အသေးစိတ်



ကြောင်းပါ။ များကို အသေးစိတ်  
လောက် ပေး ပေး ပေး ပေး ပေး  
များကို အသေးစိတ်

စိတ်ချိန်  
Transaction များကို  
ကြောင်းပါ။ များကို  
အသေးစိတ်

13. Proxy چیزی یعنیکه ینجا کسی چیزی را

