* 081. CRUD 기능 하나로 합치기

Create

(Retrieve
Read)

Update

Delete

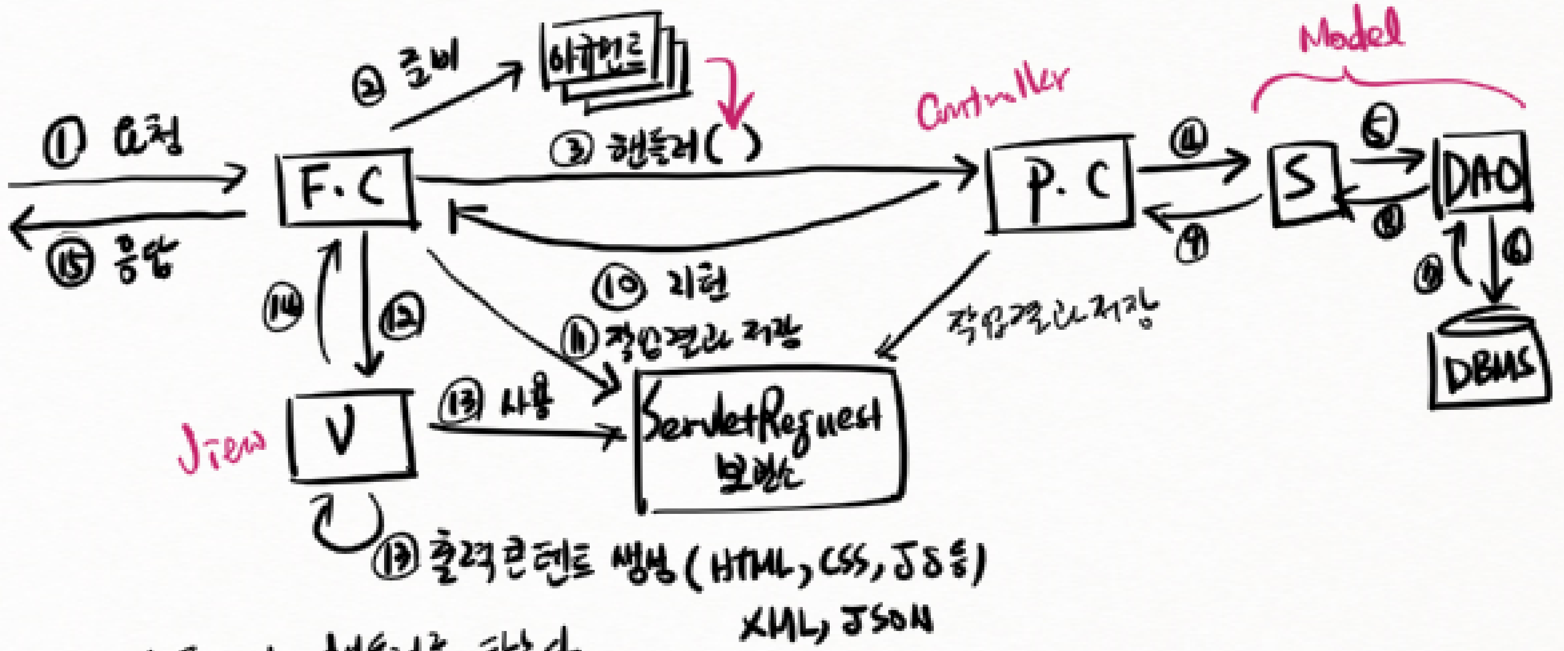
BoardFormController
BoardAddController
BoardListController
BoardDetailController
BoardUpdateController
BoardDeleteController
BoardFileDeleteController

⇒

BoardController

- form()
- add()
- list()
- detail()
- update()
- delete()
- fileDelete()

* 082. Spring WebMVC Framework 사용법



① @RequestMapping 사용법

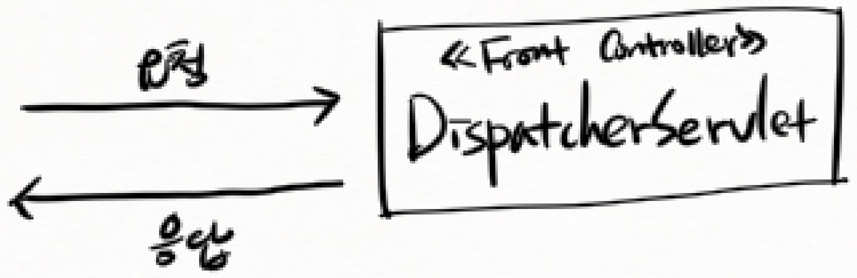
② 스프링의 CharacterEncodingFilter 사용법

③ 필터를 자바 코드로 배치하는 방법

④ "요청 핸들러"의 파라미터 다루는 방법

request handler : 요청이 들어왔을 때 호출되는 메서드
예) add(), list(), detail() 등

1) 요청 URL을 처리할 페이지 컨트롤러의 핸들러를 찾는다
2) 핸들러의 파라미터를 알아낸다
3) 핸들러에게 넘겨줄 파라미터 값을 준비한다
4) 준비한 파라미터 값을 가지고 핸들러를 호출한다
5) 핸들러가 리턴한 값을 분석한다
6) ServletRequest 보관소에 담아야 할 값은 담고
뷰 컴포넌트 URL은 해당 뷰 컴포넌트를 실행
7) 클라이언트에 응답



* 멀티파트 요청 데이터를 도메인 객체로 받기

multipart/form-data ⟶ 도메인 객체
예) Board

멀티파트 해결사

"MultipartResolver"

별도 설정이 필요합니다.

* 요청 핸들러의 리턴 타입

① void handle() {—}

↖ JSP 주소가 없다. 요청 URL을 JSP 주소로 사용한다

② String handle() {—}

↖ JSP URL

③ Model handle() {-}

↖ ServletRequest 보관소에 담을 객체들을 리턴
JSP 주소가 포함되어 있지 않다.
→ 요청 URL을 JSP 주소로 사용한다

④ Map handle() {—}

↖ ③번과 같은 방식으로 처리한다

⑤ ModelAndView handle() {—}

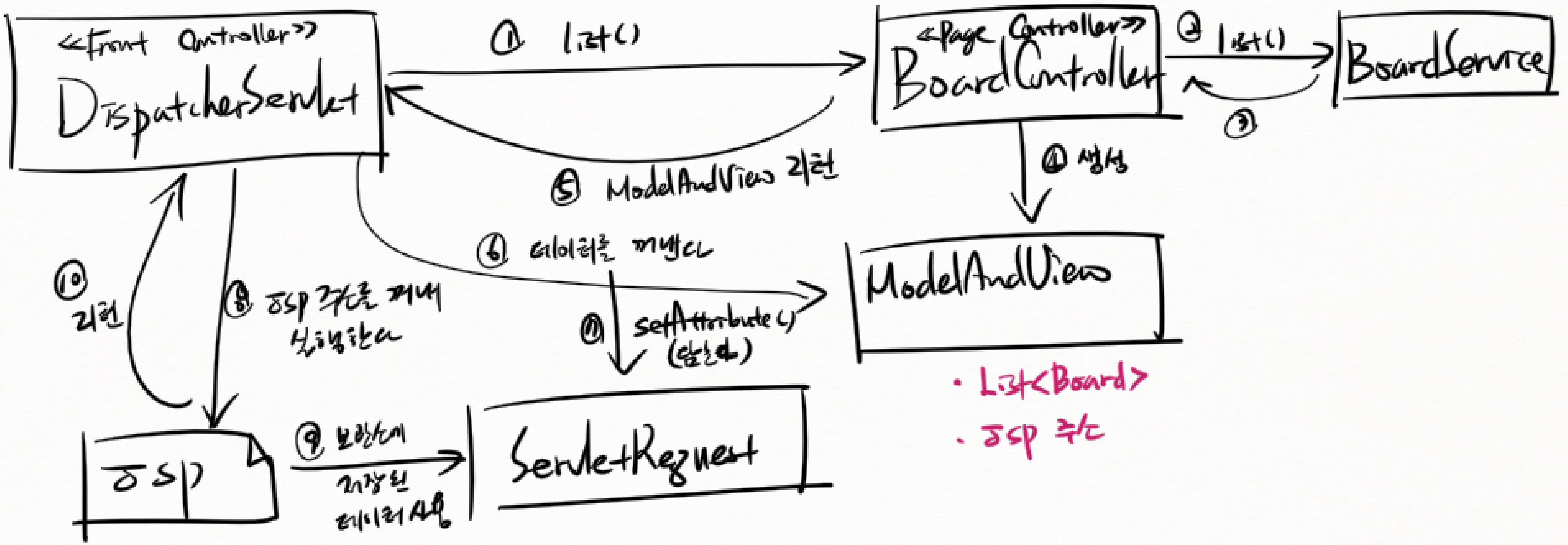
↖ JSP 주소 + ServletRequest 보관소에 저장할 객체들

* Front Controller (DispatcherServlet)는 요청 핸들러의 리턴타입에 따라 적절하게 처리한다.

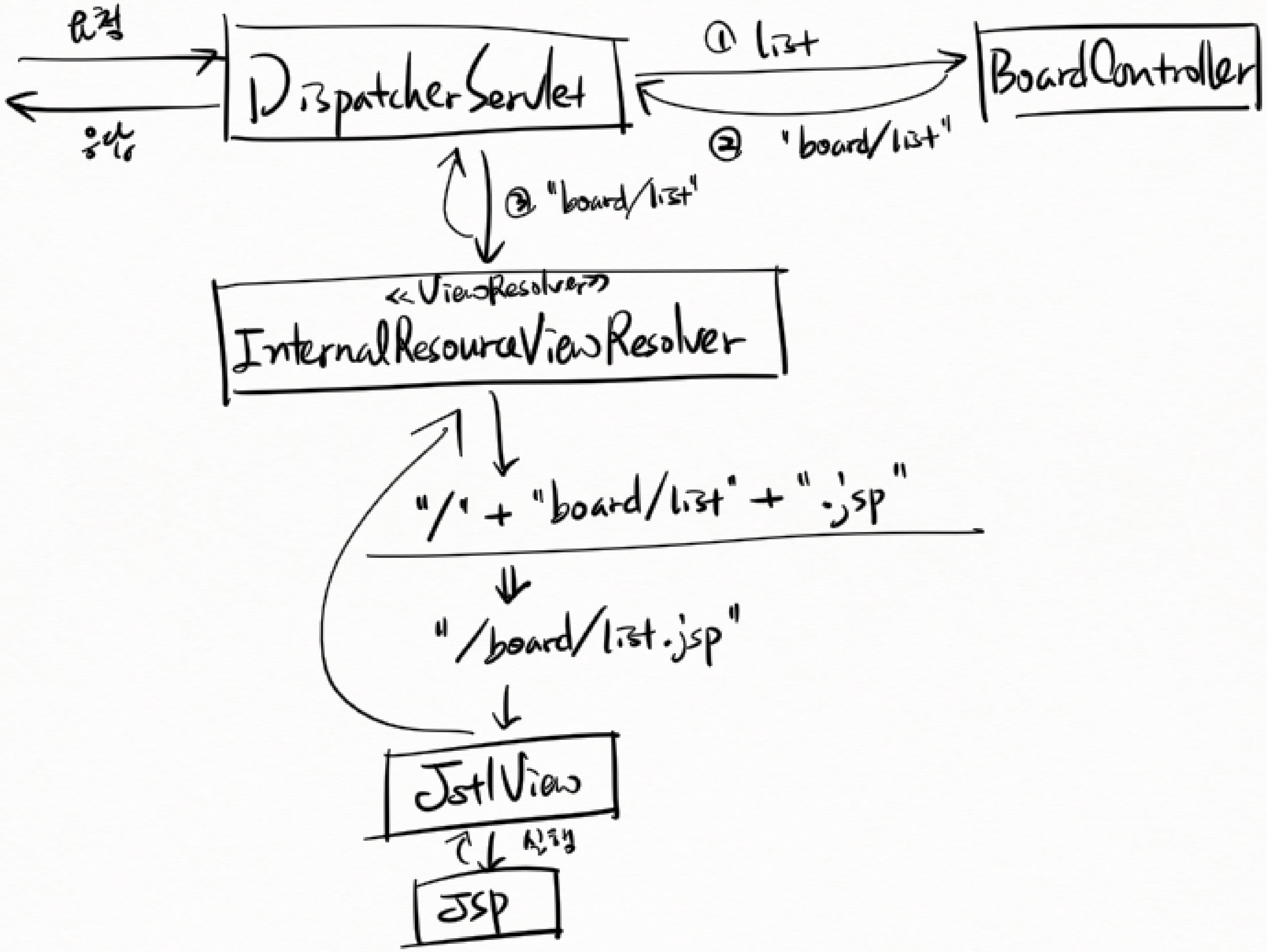
⑥ View handle() {—}

↖ JSP 주소를 View 객체에 담아 리턴

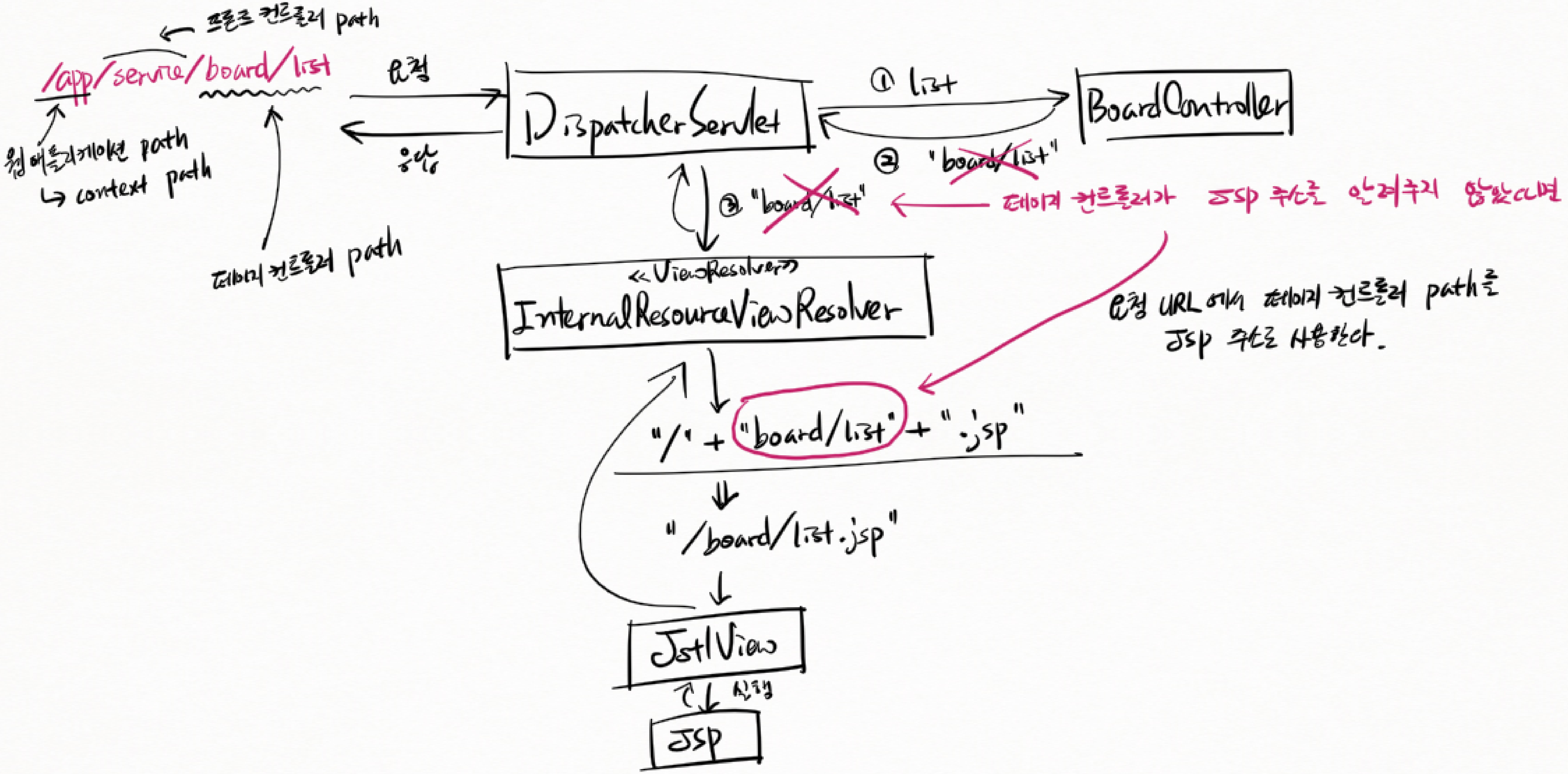
* ModelAndView 와 DispatcherServlet



* ViewResolver



* ViewResolver


프론트 컨트롤러 path
/app/service/board/list
웹 애플리케이션 path
↳ context path
페이지 컨트롤러 path
요청
응답
DispatcherServlet
① list
BoardController
② ~~"board/list"~~
③ ~~"board/list"~~
페이지 컨트롤러가 JSP 주소를 알려주지 않았다면
<<ViewResolver>>
InternalResourceViewResolver
요청 URL에서 페이지 컨트롤러 path를
JSP 주소로 사용한다.
"/" + "board/list" + ".jsp"
"/board/list.jsp"
JstlView
실행
JSP