

# 61. Spring Framework 8.0 출시!

- ① 'spring-webmvc' 확장 버전을 출시한 이후

✓ ( spring 6.x → Jakarta EE 11 & 9.x      jakarta.\*      Tomcat 10.x  
  spring 5.x → JavaEE (8.x) = Jakarta EE (8.1)      jaxws.\*      Tomcat 9.x )

- ② 스트링 애플리케이션으로 뷰를

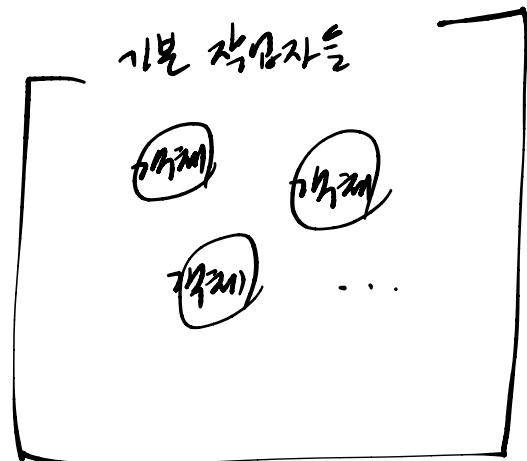
- ③    " 템플릿 JSP "

- ④    " Application 빌드 AppConfig 123 "

- ⑤    " 콘트롤러와 서비스 뷰 "

- ⑥    " IoC 컨테이너로 뷰 "

# \* Spring Framework



이런 기능을 처리할 때  
적용자가 등록되어 있으려면  
(인증)  
그 적용자를 실행 (마이그로우)하면  
처리된다.  
없으면 예외를 던져서 기능을 무시한다.

기능 추가?

그 기능을 수행할 적용자를 등록  
(적용자)



- ① 적용 적용자를 사용해서 등록
- ② (액션레이아웃) 설정을 통해  
(XML)

\* 요청 자세히 봐서 요청 헤더의 자세히 봐

<form>

```
<input name="email"/>  
<input name="password"/>  
<button>로그인</button>
```

POST로

</form>

요청 헤더

요청 자세히 봐

email=aaa@test.com & password=1111

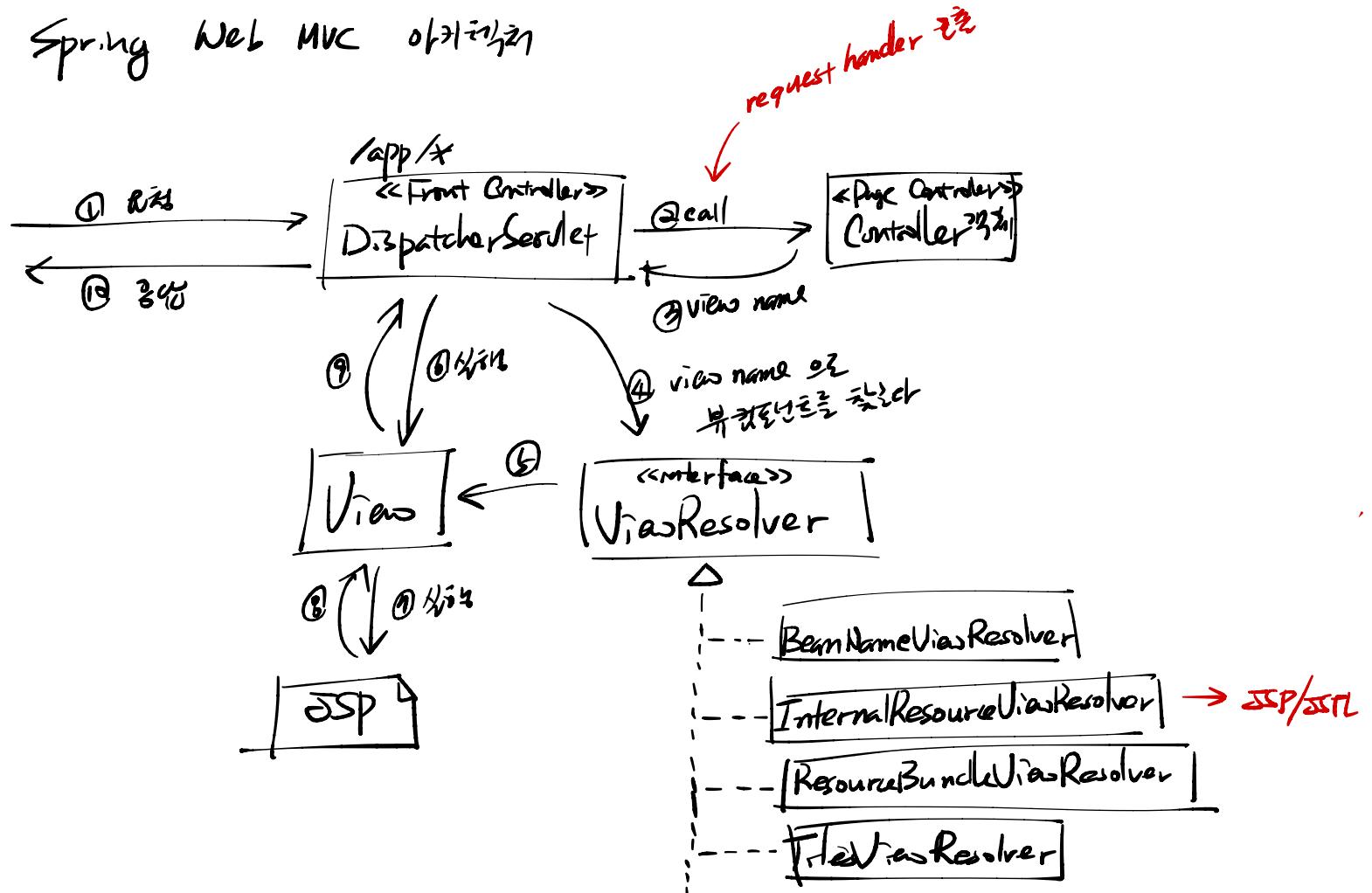
String

login(String email, String password) {

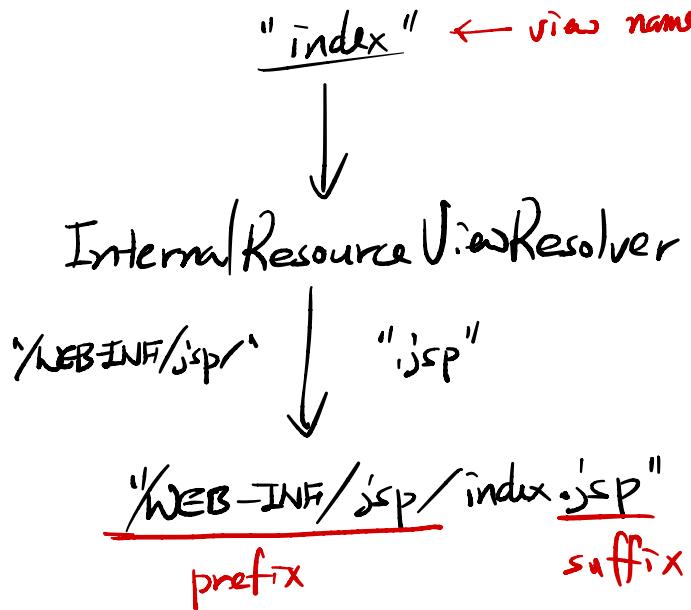
요청 헤더의 자세히 봐

}

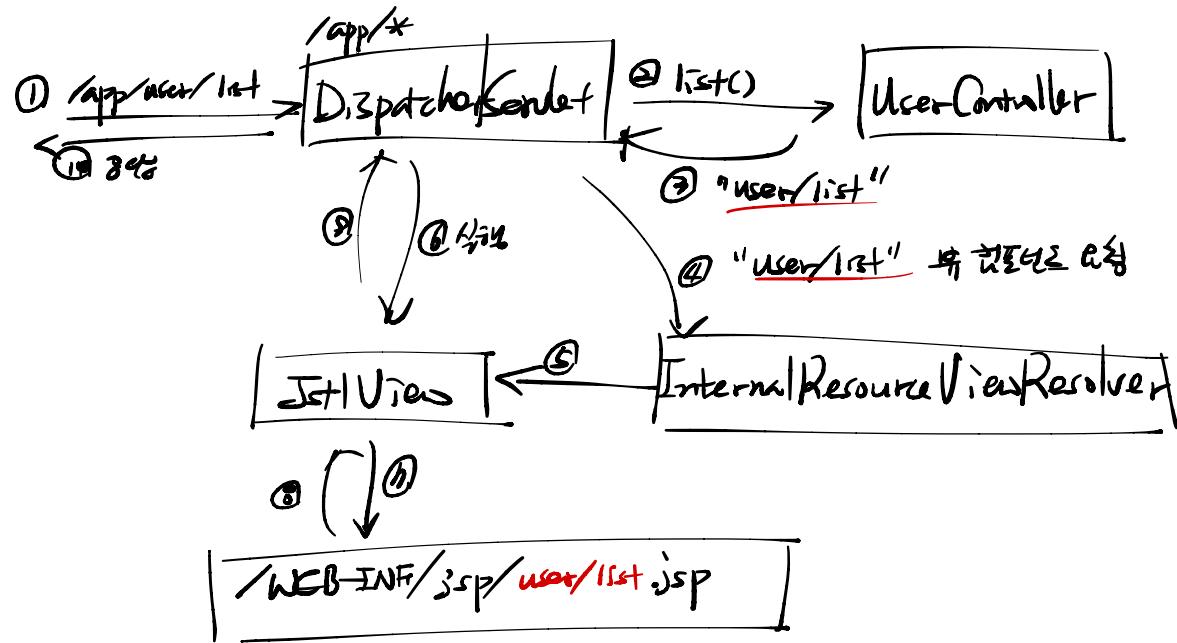
## \* Spring Web MVC 07/27/21



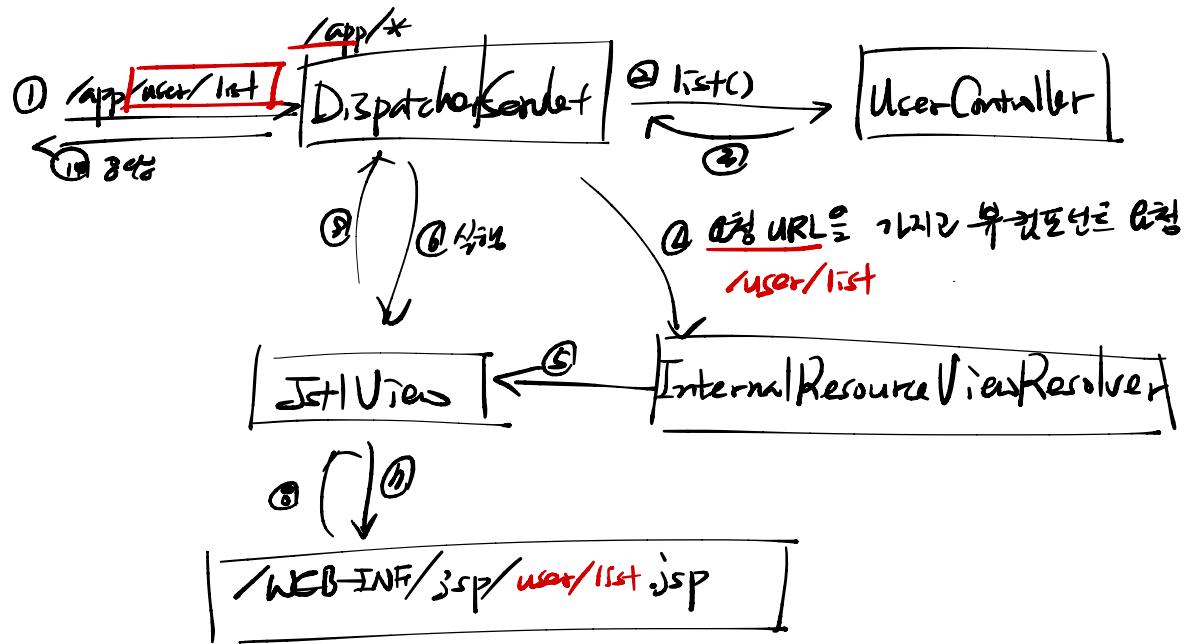
- \* Internal/Resource ViewResolver



\* view name %\*% can

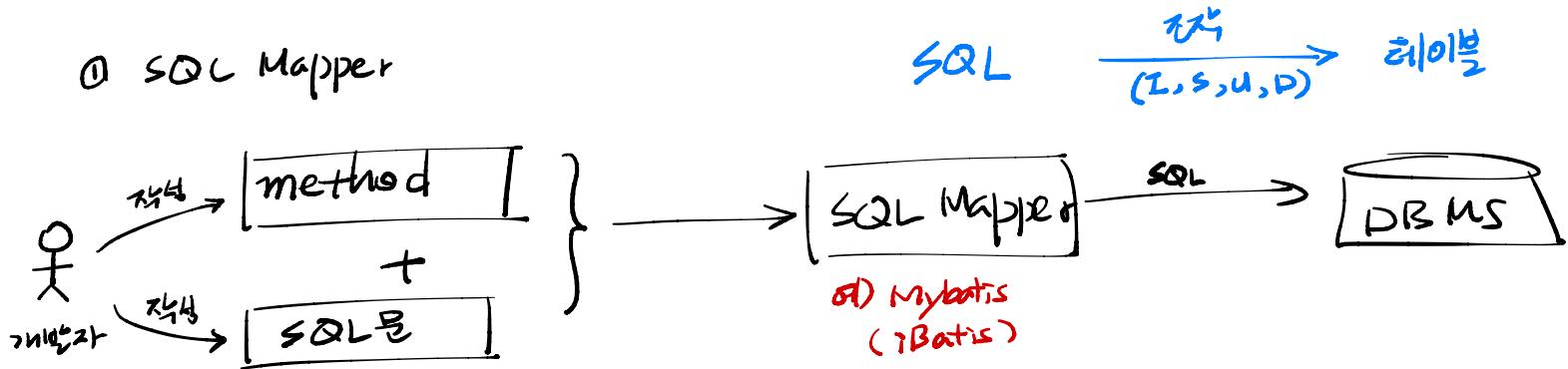


\* view name  $\stackrel{\text{보통}}{\rightarrow}$  controller

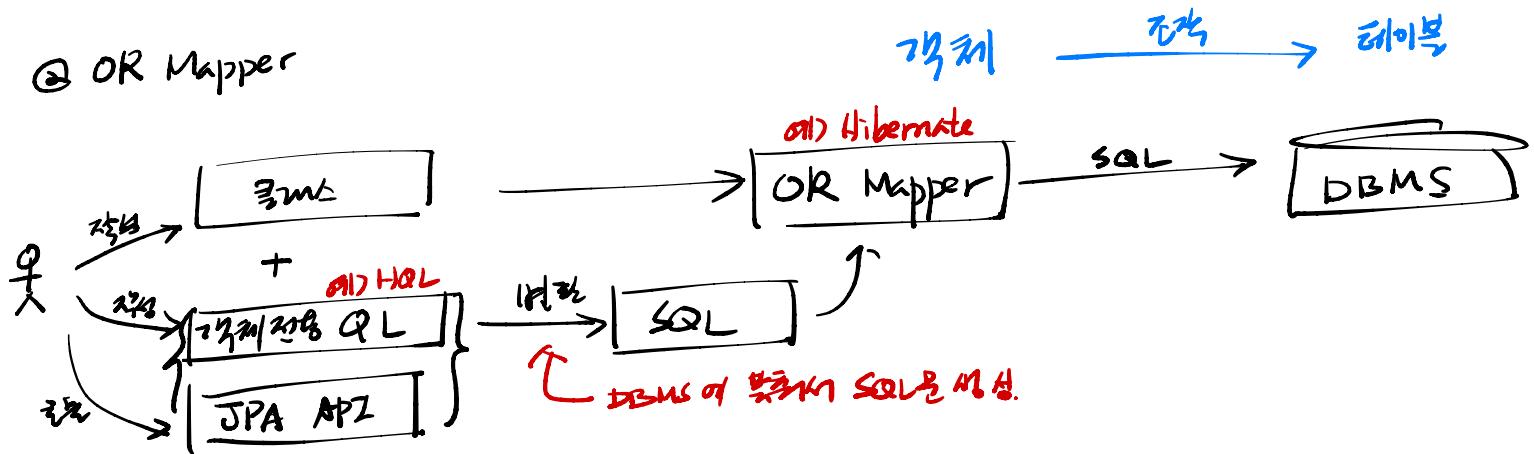


## \* SQL Mapper vs OR Mapper

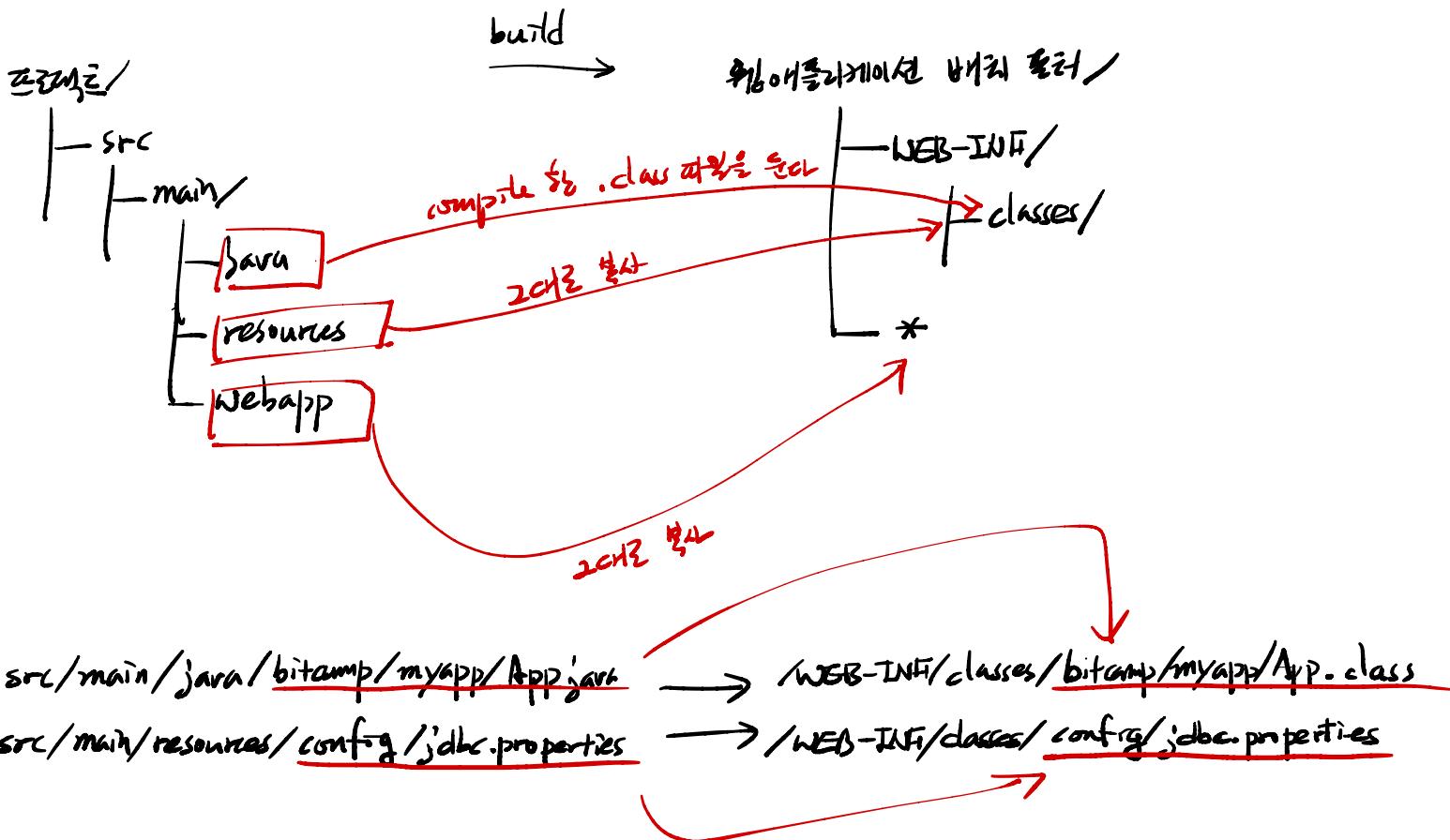
### ① SQL Mapper



### ② OR Mapper



## \* build et classpath



\* DaoFactory

↑  
생성  
DAO

SqISessionTemplate

↑  
Mybatis  
가져온  
DAO

[BoardDao]

insert()

[DaoFactory]

↳ sqISession.insert("BoardDao.insert", -);

[SqISessionTemplate]

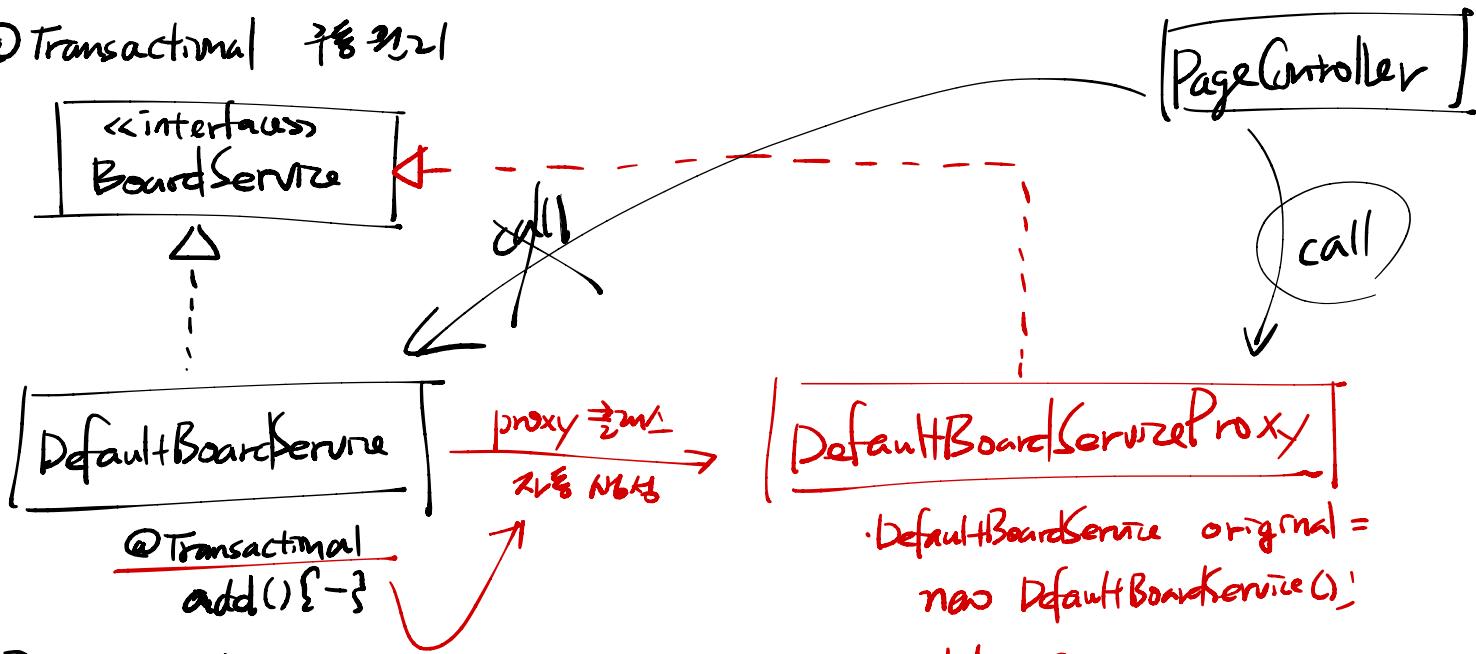
↳ sqISession.insert(

"bifrap-myapp.dao.BoardDao.insert",  
- );

SQL  
실행  
하기위한  
SQL문  
생성  
하기위한  
Template

MyBatis  
Template

\* @Transactional 78 31, 21



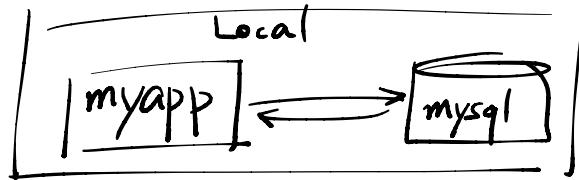
\* AOP의 특징, 예제

↳ 기존 코드를 대체하지 않고  
추가 기능을 적용하는 방법  
↳ proxy 구현은 어렵지만 ("proxy를 만들기 어렵다")

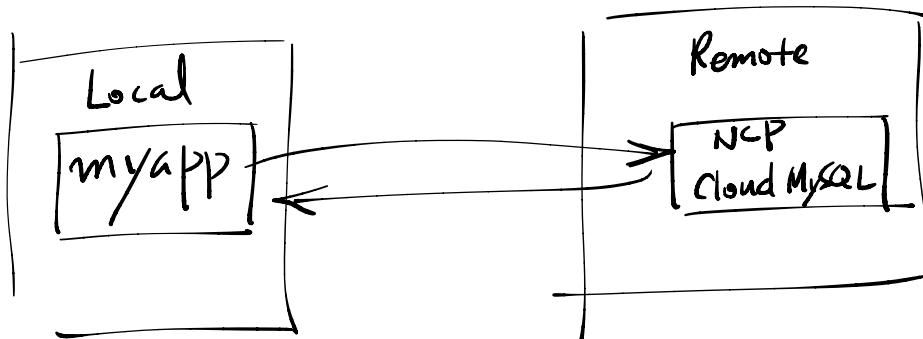
. **add () {**  
    **try {**  
        **original.add();**  
        **txManager.commit();**  
    **} catch {**  
        **txManager.rollback();**  
    **}**

## 62. Naver Cloud Platform 사용 예

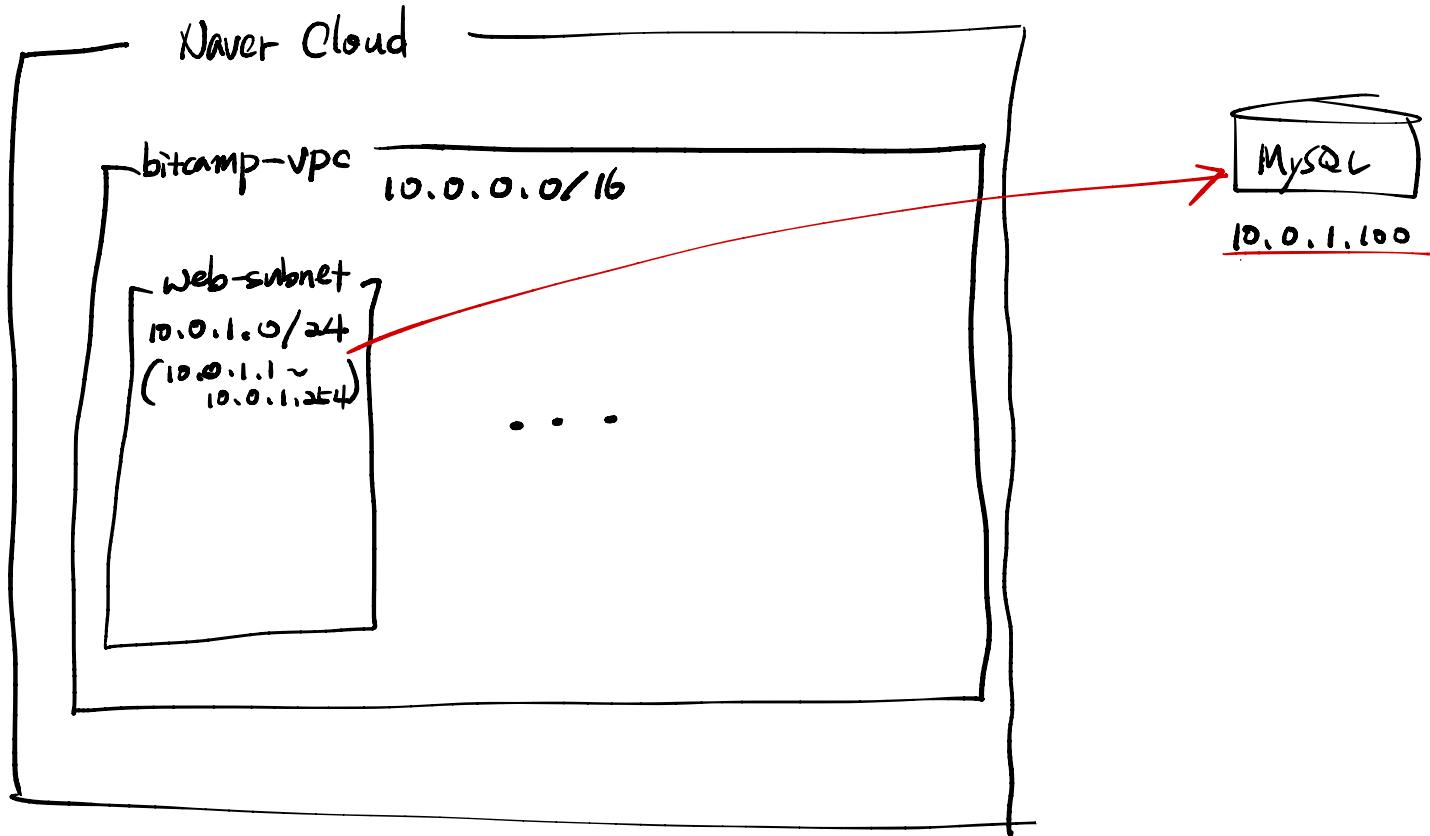
\* ORI



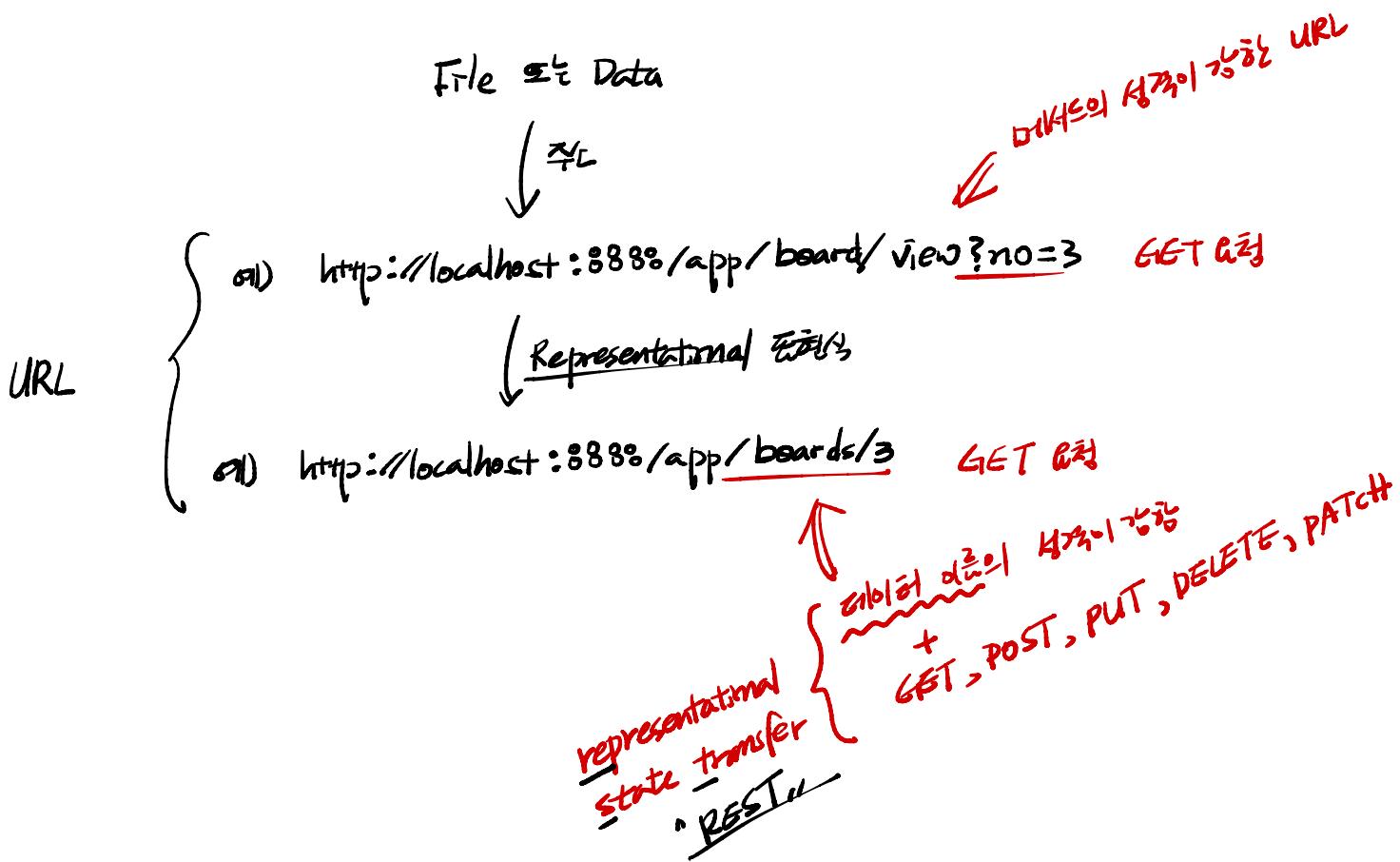
\* 대입



\* VPC (Virtual Private Cloud) - 퍼블릭 네트워크 대신 내부망을 사용하는 네트워크



## \* RESTful API



\* function → REST API

다른 프로토콜

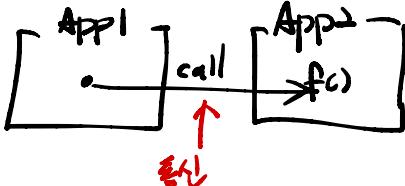
① function

(RPC)

② Remote Procedure Call

(RMI)

③ Remote Method Invocation



C 등 전통적인 프로그래밍

기업의 업무 처리



App의 기능을 분산 (분산 컴퓨팅)



한 App이 하던 일을

여러 App으로 퍼뜨려 실행하는

App 간의 일종의 협동체



다른 다른 App의  
function을 호출할 수 있는  
기술이 등장하게 되었다.



C++ 등 대형화된 프로그래밍



ODP 프로그래밍의 특징이 빠듯  
RPC를 개선

\* function → REST API

③ RMI → ④ CORBA

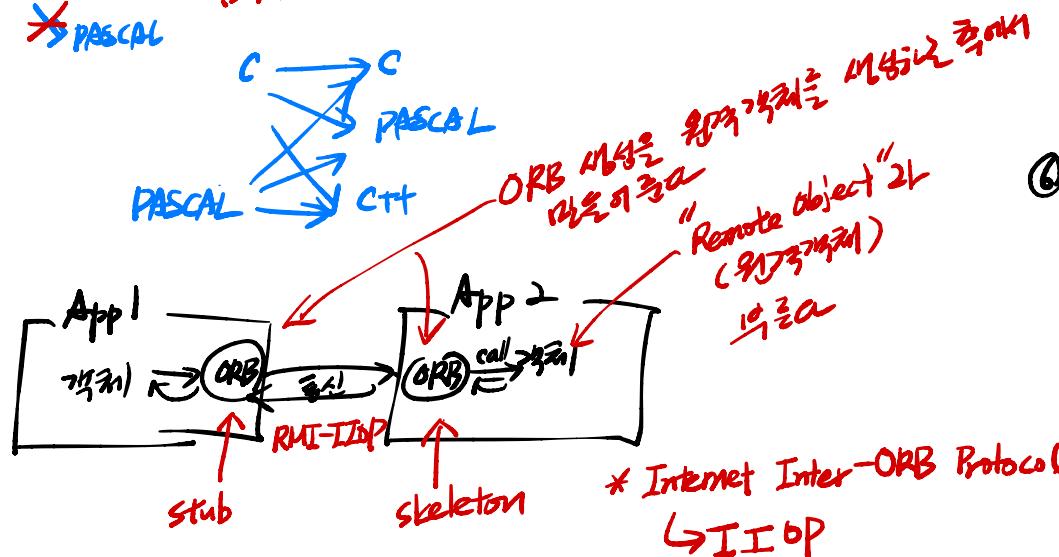
같은 인터페이스  
같은 App끼리  
같은 환경에서  
만나 가능

common  
Object Request Broker  
Architecture

자신의 인터페이스를  
다른 App끼리  
같은 환경에서 만날 수 있는  
방법

C → C  
X PASCAL

C → C  
X PASCAL  
X C++



⑤ Web-service

✓ 웹으로 API HTTP 프로토콜을 활용  
✓ ORB를 통과하지 않고 단순화  
서버에서 직접

단순화된 웹으로 개발자의  
작업 부담을 줄여주는  
기술입니다!

⑥ Enterprise JavaBeans (EJB)

Java에서 제공하는 EJB 기술

단점!

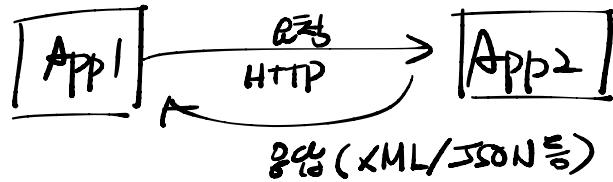
2000년 초반에는 PC에서  
제작된 데이터베이스를 써서 EJB로 개발  
하면서 성능이 크게 떨어지거나 오류가 많았습니다.

\* function → REST API

(Web-Service)  
EJB

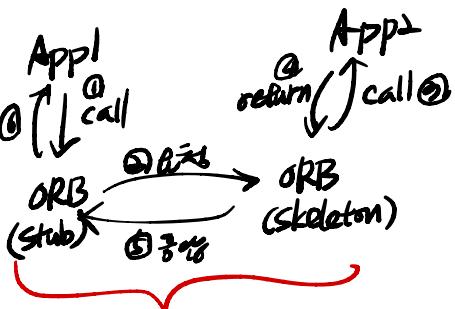
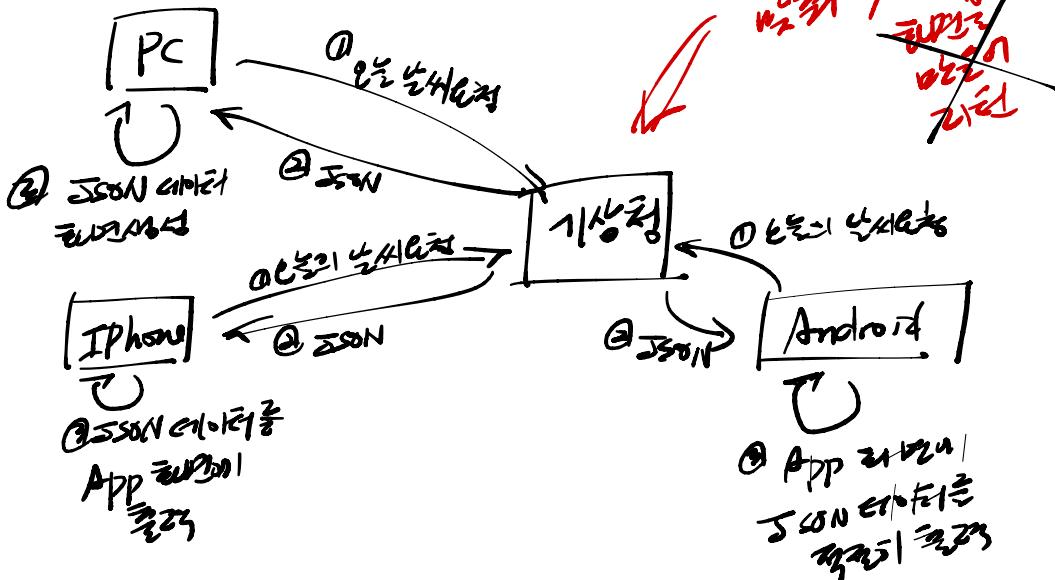
→ REST API

인터넷으로  
ORB를 통한  
서비스 제공



Response (XML/JSON 등)

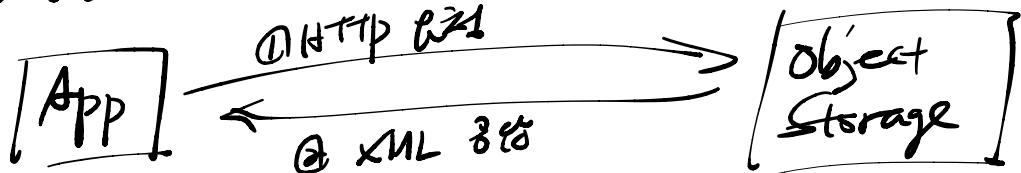
인터넷으로  
ORB를 통한  
서비스 제공



ORB를 통한  
서비스 제공

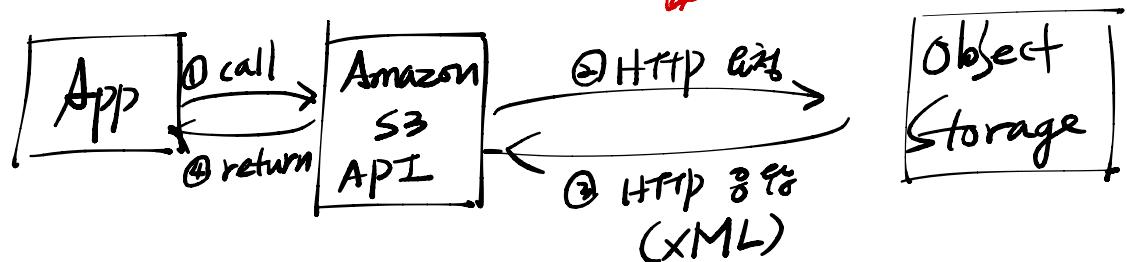
\* Object Storage REST API API  
↳ HTTP 퀼/값, 키!

① 직접 접근

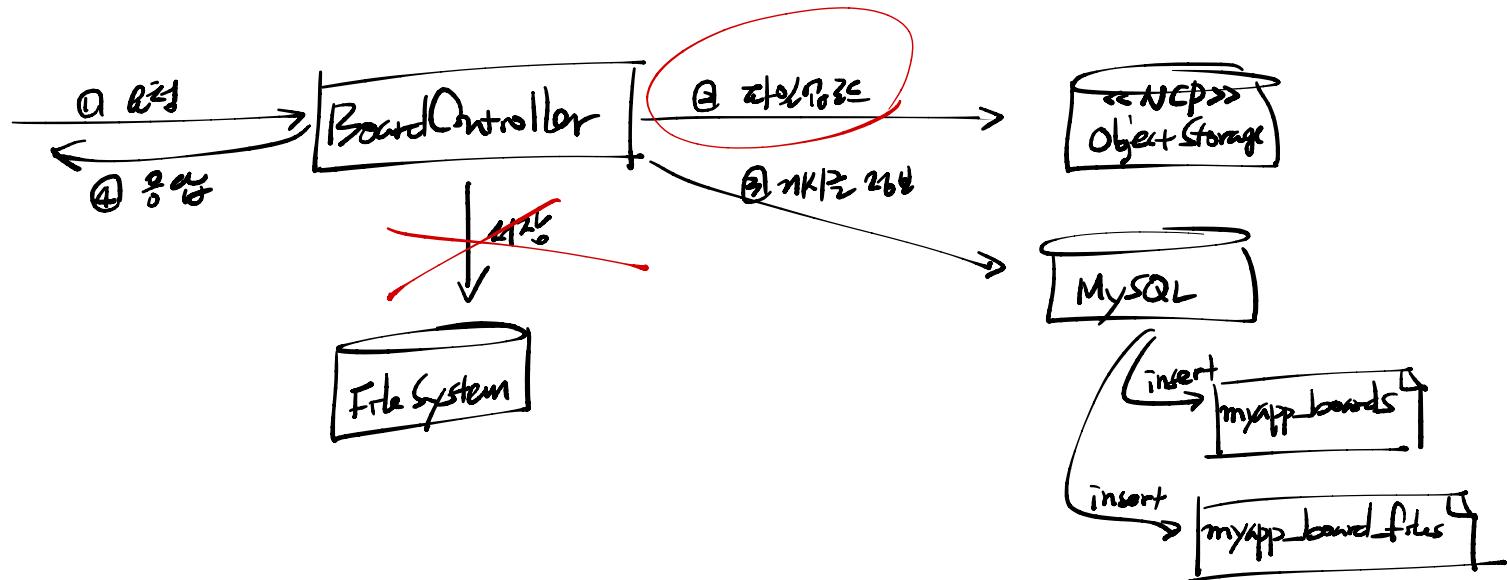


HTTP 퀼/값은 직접 접근하기 편리하다  
HTTP 응답을 직접 접근하기 편리하다

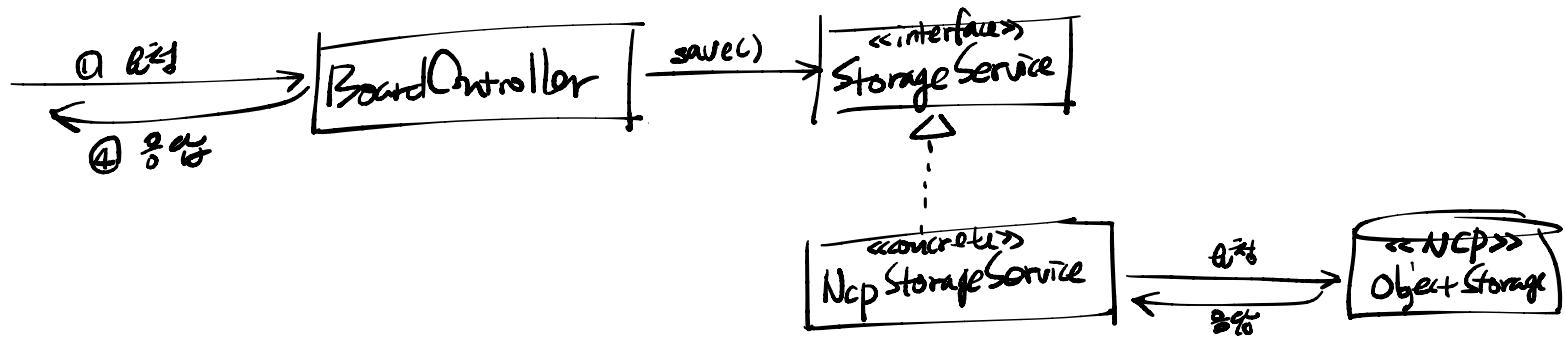
② 중간 거치



\* 내부로 첨부파일은 NCP의 Object Storage에 저장

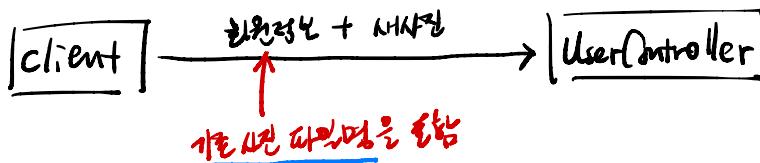


\* 게시글 첨부파일은 NCP의 Object Storage에 저장 → NCP 연동로직을 서비스 객체로 분리



## \* 사용고객과 보안

① 보안에 악용은 예 : 회원정보 변경



A라는 사람이 로그인 한 후에  
마술의 정보를 바꿔서 사용자인증을  
다른 사람의 것으로 치환할 수 있다.  
↳ 다른 사람의 사용자인증을 할 수 있다.



이제, 변경, 삭제 시

기존 정보를 사용할 때는

항상 새롭게 프로그램 사용하자!  
기존 기반코드를 클라이언트가 만지 말자! → 꼭 새롭게 프로그램 사용하자!

- i) 기존 사용자인증 → 클라이언트가 보낸 세션키로  
세션을 훔쳐 사용하기
- ii) NH 네이버 인증 → 세션을 훔쳐
- iii) 카카오 인증 → DB 변경

## \* Transaction Propagation

<u>Caller</u>	Transaction	
<u>Propagation</u>	X	O (Tx1)
(default) REQUIRED	Tx1	Tx1
REQUIRES_NEW	Tx1	Tx2
MANDATORY	예외	Tx1
SUPPORTS	선택적 투명성 none	Tx1
NOT_SUPPORTED	불행	현재 트랜잭션 외부의 상황을 적용
NEVER	실행	예외

\* propagation  $\rightarrow$  ~~to~~ or:

① REQUIRED

UserController.delete() X  
↓ call  
Tx1 { DefaultUserService.delete()

Tx1 { UserController.delete()  
↓ call  
DefaultUserService.delete()

② REQUIRES-NEW

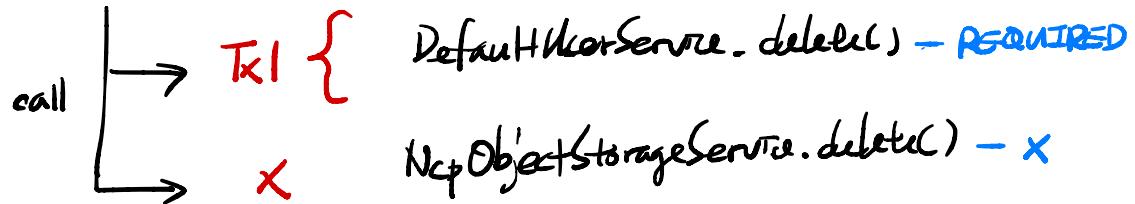
UserController.delete() X  
↓ call  
Tx1 { DefaultUserService.delete()

Tx1 { UserController.delete()  
↓ call  
Tx2 { DefaultUserService.delete()

\* 키워드 어노테이션 정리

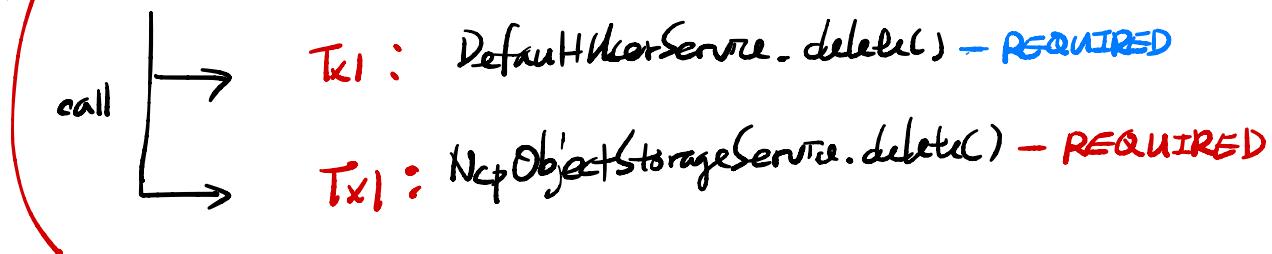
① 허용

X UserController.delete() X

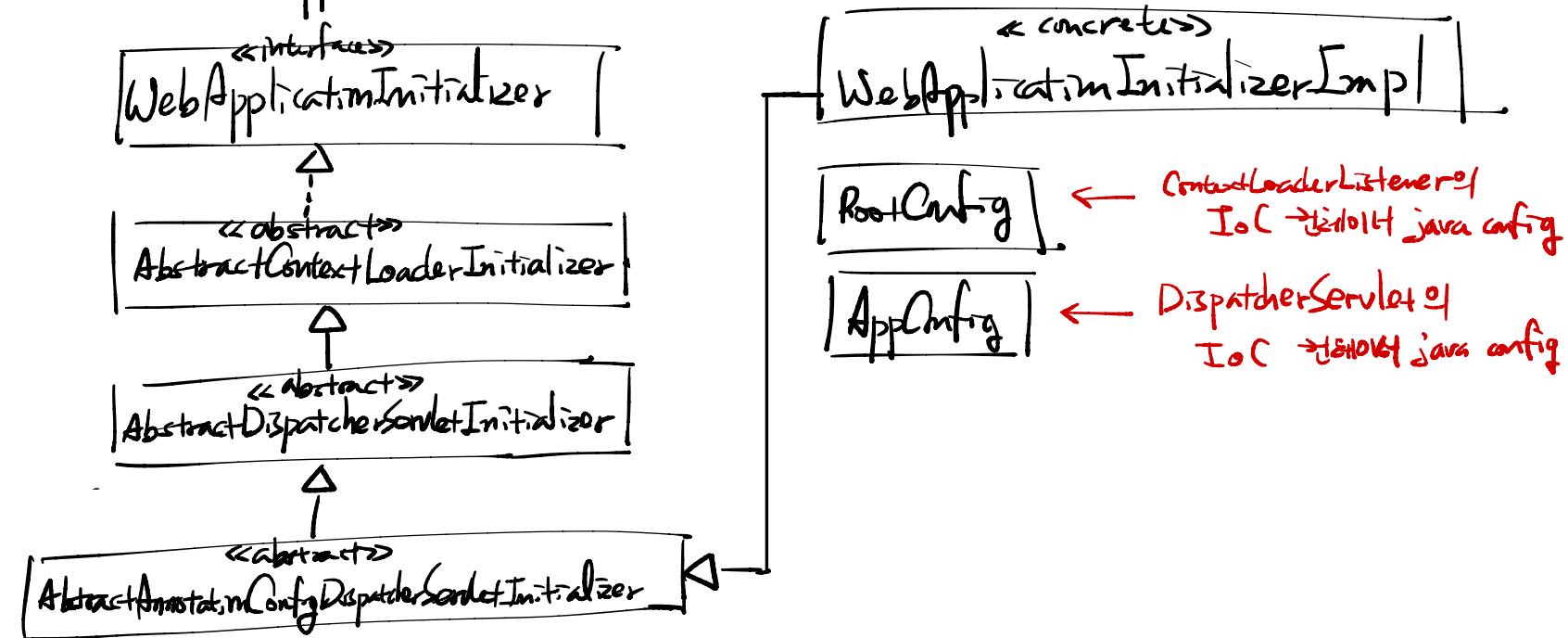


② 허용

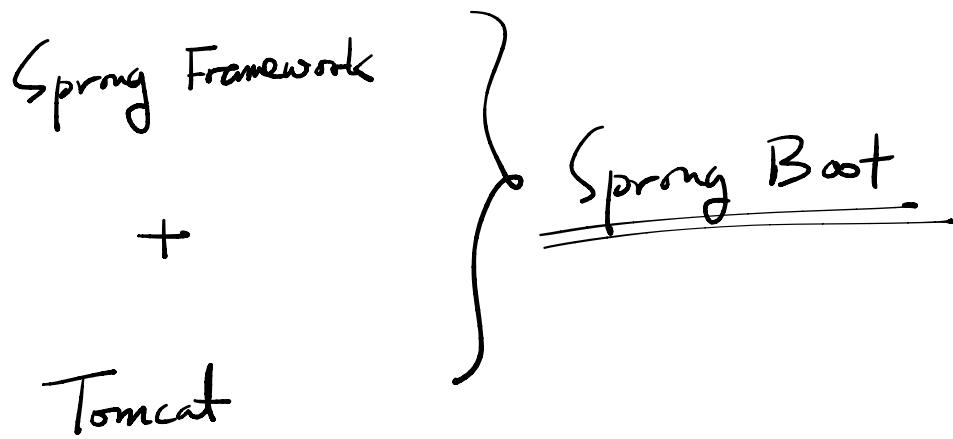
Tx1 UserController.delete() - REQUIRED



## 63. WebApplicationInitializer



## 65. Spring Boot mpls



66.  $\text{SELECT}_{[012]} \text{ FROM}_{[1]}$

$$\text{SELECT}_{[012]} \text{ FROM}_{[1]} \text{ WHERE} = \frac{(\text{012}(\text{011012})^4 \times 215)}{= \text{0121010101010101}}$$

select  
from  
where  
order by

limit 3개, 215

limit 0, 4

limit 4, 4

limit 8, 4

limit 12, 4

limit 16, 4

- |    |       |                      |
|----|-------|----------------------|
| 0  | _____ | ) 1 $\text{2150121}$ |
| 1  | _____ |                      |
| 2  | _____ |                      |
| 3  | _____ |                      |
| 4  | _____ | ) 2 $\text{2150121}$ |
| 5  | _____ |                      |
| 6  | _____ |                      |
| 7  | _____ |                      |
| 8  | _____ | ) 3 $\text{2150121}$ |
| 9  | _____ |                      |
| 10 | _____ |                      |
| 11 | _____ |                      |
| 12 | _____ | ) 4 $\text{2150121}$ |
| 13 | _____ |                      |
| 14 | _____ |                      |
| 15 | _____ | ) 5 $\text{2150121}$ |
| 16 | _____ |                      |
| 17 | _____ |                      |

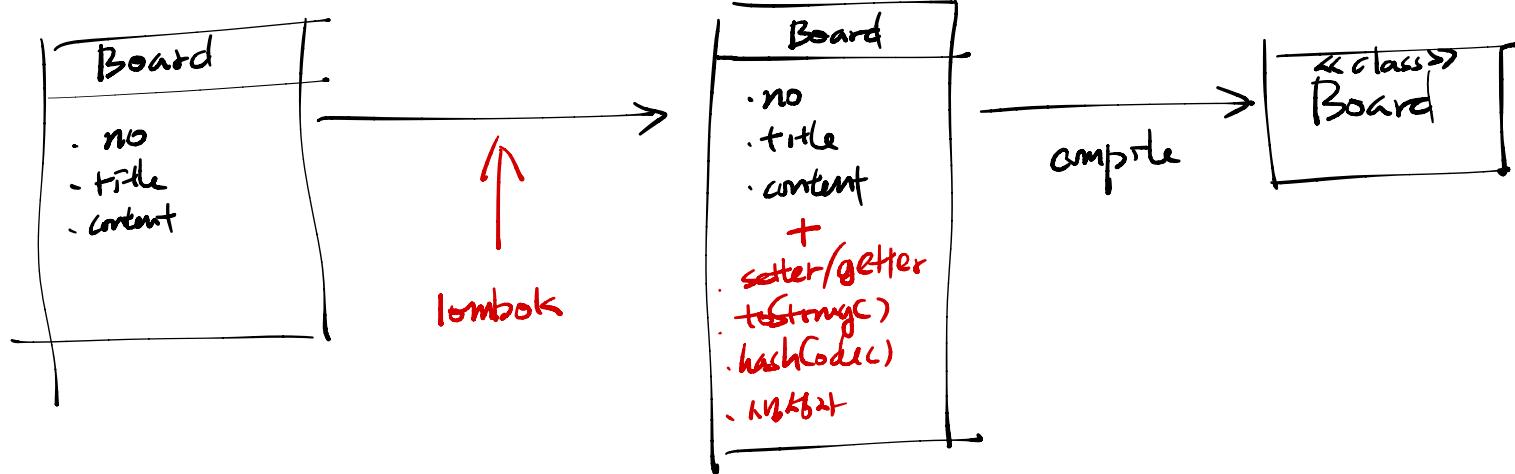
## 67. Lombok 2010-2021年版

↳ Domain 实体 には 何が付く → setter/getter, toString(), hashCode(), equals()  
何が付かない なに?

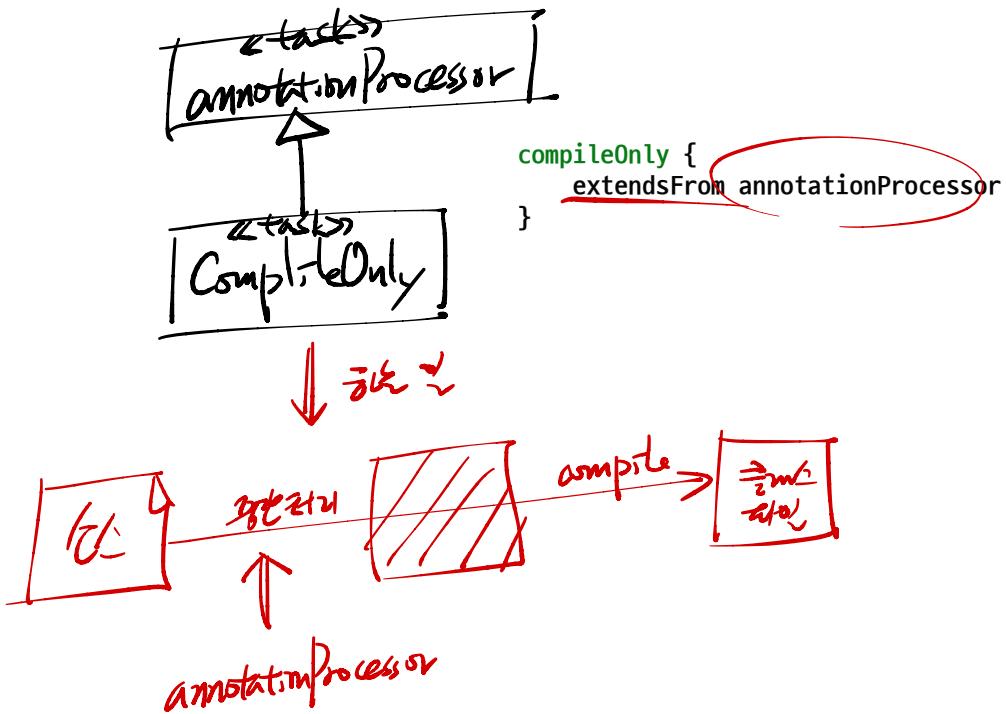
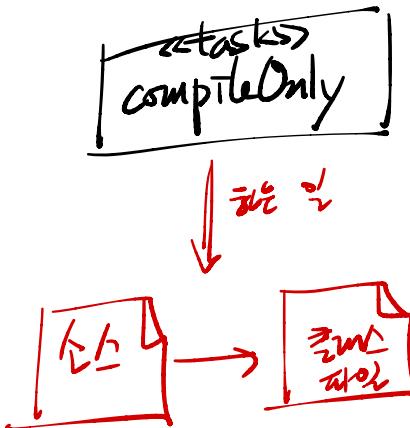
Gradle



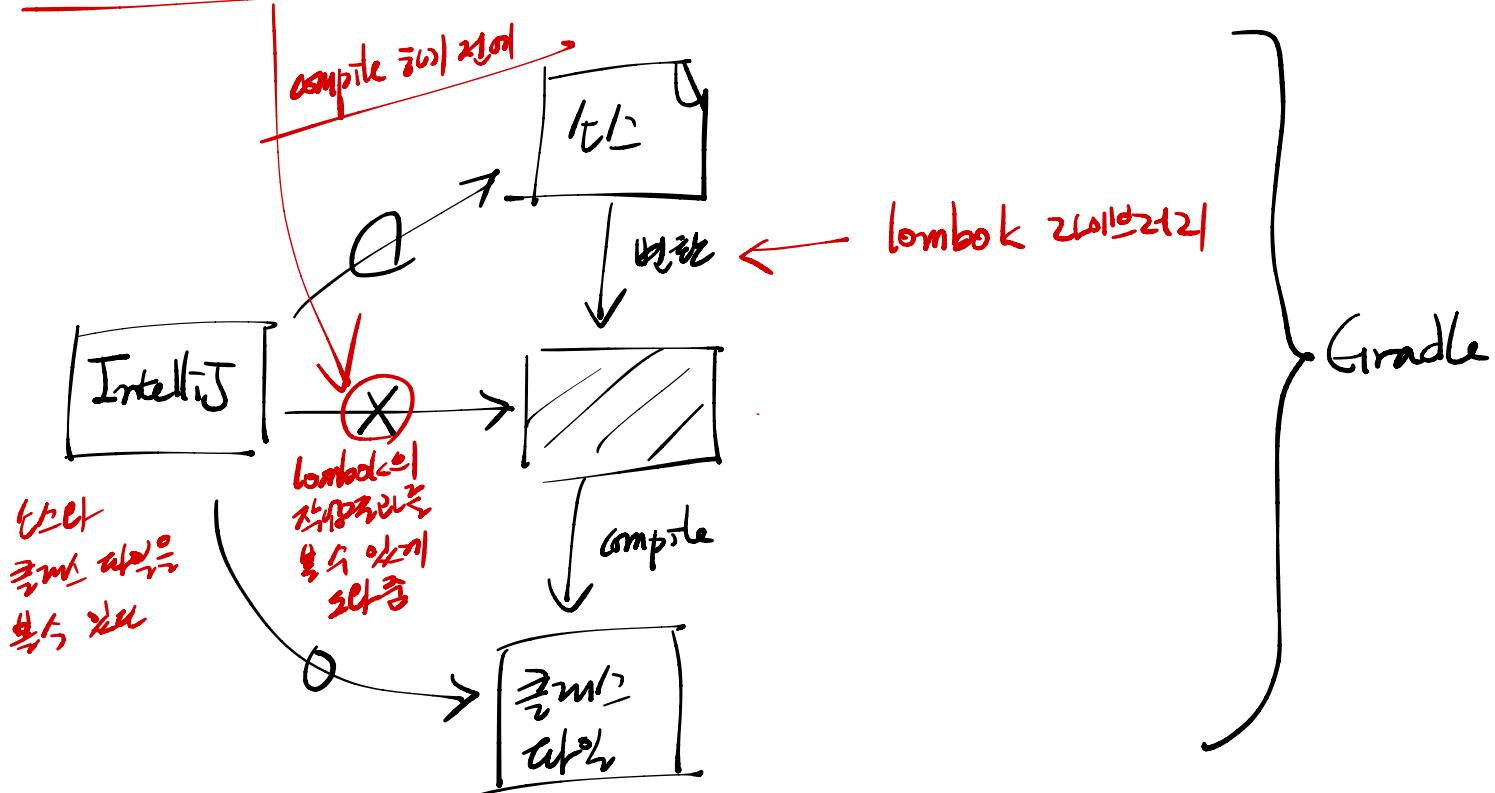
Build



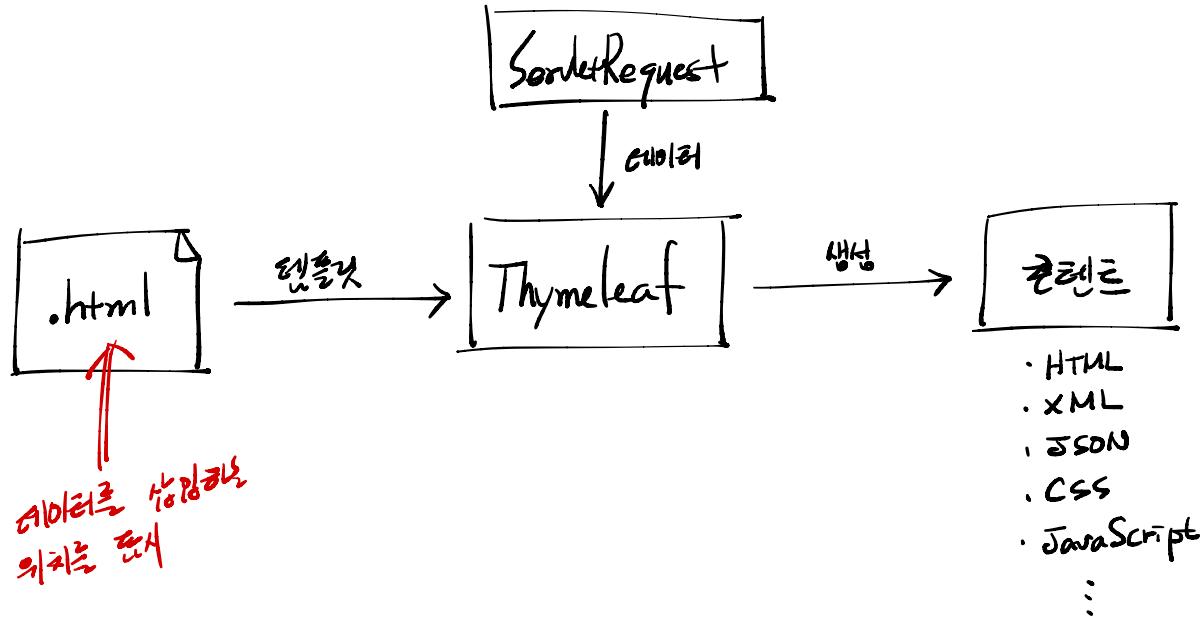
## \* Gradle et Task



## \* lombok IntelliJ 풀이방법의 차이



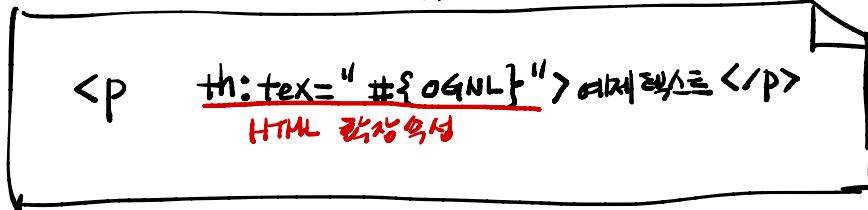
\* Thymeleaf 템플릿 인자



## \* Thymeleaf 템플릿과 HTML

.html

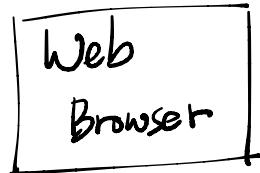
템플릿 파일



JSP → \${변수명}

Mybatis → #{} выражение

Thymeleaf → \${변수명}



HTML 편집과 편집을 허용



디자이너

디자이너는 Thymeleaf가  
수동화한 코드에 맞게 편집  
하고 편집을 확인할 수 있다

### Thymeleaf의 특징

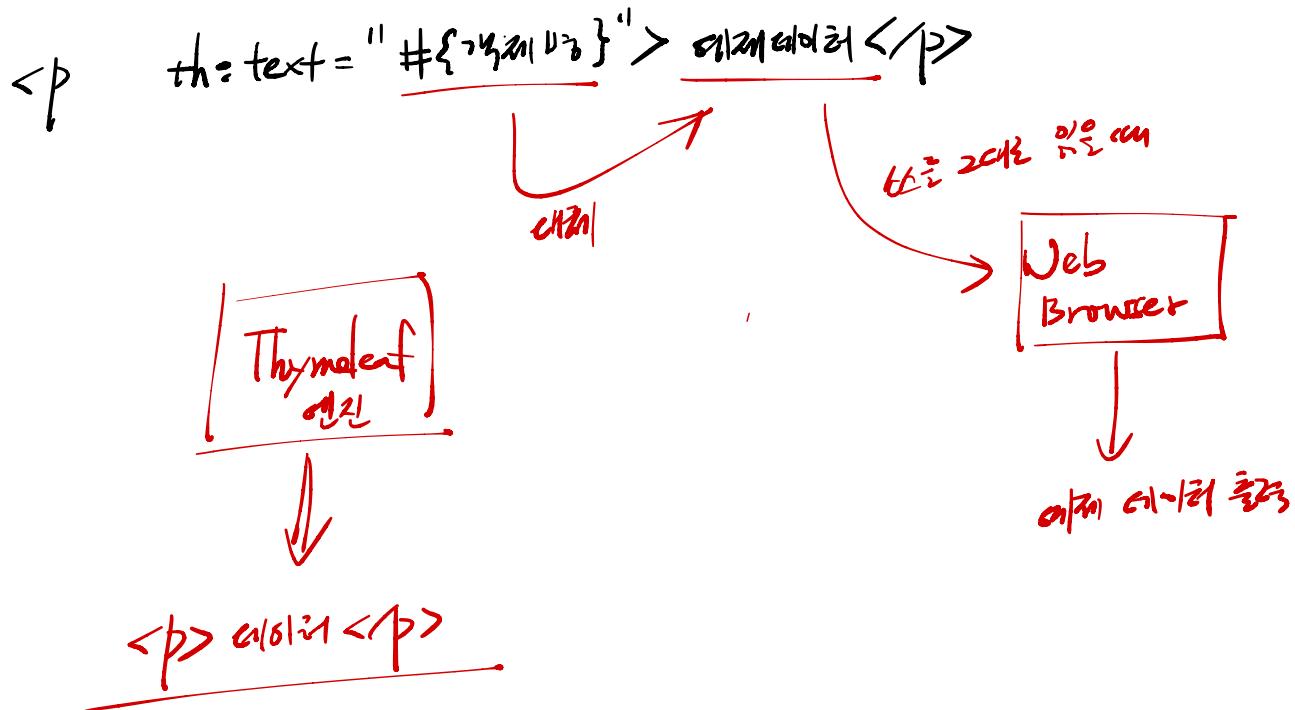
- ① JSP와 다르게 템플릿이 HTML 파일이다.

↳ 템플릿 언어의 사용성이  
스스로로 웹브라우저에서  
제작되는 대목으로 활용된다.

- ② Thymeleaf 파일이 HTML 편집을 허용하기 때문이다

스스로로 편집을 대화형으로 허용된다.

\* Thymeleaf 템플릿 HTML



## \* HTML의 속성과 属性

<table> th: text = "국민당" <td> </td>

↑  
국민당  
국민당



<td> </td>

## \* Thymeleaf Namespace

namespace ( прост)

```
<html xmlns:th="http://www.thymeleaf.org">
```

Thymeleaf 엔진이 Thymeleaf 엔진이  
th 네임스페이스로 시작하는 것은 사용한다

th 네임스페이스로 시작하는 것은 사용한다

```
<p th:text="${welcome}">환영합니다!</p>
```

Thymeleaf 엔진은  
HTML 템플릿

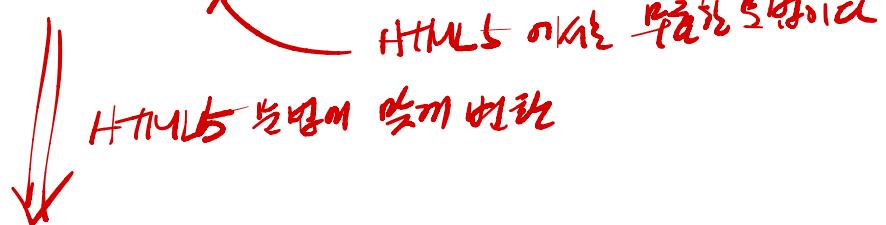
th:\* 대체 속성을 말해  
Thymeleaf는 대체 속성으로 인식하여  
처리된다

※ XML namespace 처리는  
HTML 문서에 넣는다

th: 를 기본으로 인식한다  
그것은 가능할 생각하지 마!

## \* HTML5 et Thymeleaf

```
<p th:text="${welcome}"> 안녕 </p>
```



```
<p data-th-text="${welcome}"> 안녕 </p>
```

HTML5 속성이  
여러개인 경우

\* HTML5의 템플릿 속성

data-th-text = " \${id}"

"data-th-text"

## \* th-text vs th-utext

welcome = <u>안녕</u> 친구님.

data-th-text

escape 문자를 차단한다

HTML문은 링크처럼 사용하는  
HTML 키워드에 대응되는 문자를  
보관하여 HTML 키워드가 영향을  
끼치지 않게 하는 기능이다.

&lt; b &gt; 안녕<u> / b &gt; 친구님.

↑

HTML Entity

(HTML 특수기)

↳ HTML keyword와 종종나는 문자를  
표현하기 위한 특수기로 사용

} HTML 키워드가 아닌  
단순 문자로 취급하기  
위함.

data-th-utext

unescape

escape 문자를  
차단하지  
않는다

HTML 키워드가  
포함된다.

< b > 안녕<u> / b &gt; 친구님.

단일 텍스트로  
HTML 특수기로  
변경된다.

## \* Link

`data-th-href = "@{url}"`

`<a href = "@{url}">`

Actual URL: http://localhost:8888/app/board/list

`@{http://localhost:8888/app/board/view}` → `href = "http://localhost:8888/app/board/list"`  
자바 경로

`@{~board/view}` → `href = "/board/view"`  
위치 root path 가는

`@{/board/view}` → `href = "/app/board/view"`  
context path 가는

`@{board/view}` → `href = "board/view"`  
같은 위치로 가는

server root path ⇒ /  
context path ⇒ /app  
(위치와 동일한 경로)

## 69. Handler & Method ArgumentResolver چی가 뭔가

request handler

파라미터 ← 사용자에게 주는 값들

String add(  
    Board board,  
    MultipartFile[] files,  
    HttpSession session) {

====

}

어디로?

request handler of  
attribution 헤더  
method

1/2-3  
정보수집  
attribution 헤더

}     {  
    SendRequest  
    SendResponse  
    HttpSession  
    @RequestParam

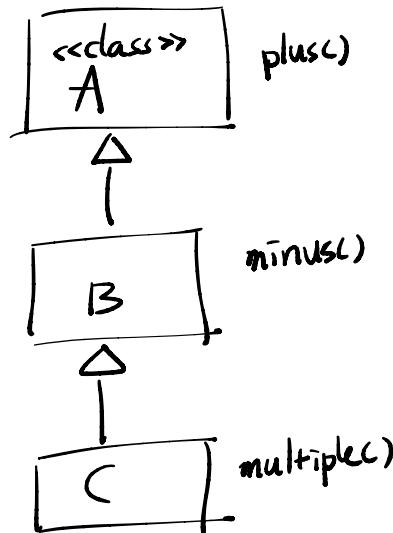
:

\* 320! 사용자의 접근을 허용하는 HandlerMethodArgumentResolver 데코더

```
public String add(  
    Board board,  
    MultipartFile[] files,  
    @LoginUser User loginUser  
) throws Exception {  
}  
...  
}
```

- ① LoginUser 어노테이션 탐지  
② LoginUserArgumentResolver 을 통해 정의

\* Class . isAssignableFrom()



A.class.isAssignableFrom(B.class)

A type의 객체가 B type의 인스턴스를 할당할 수 있나?

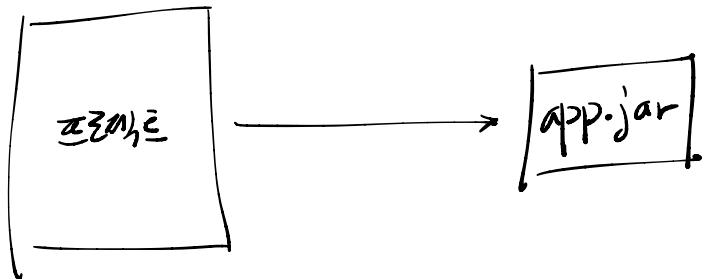
A obj = new B();  
obj . plus();                                    A + &

B obj ~~=~~ new A();

obj . minus();  
obj . plus();

## 10. Jenkins et Docker이 결합된 CI/CD 활용하기

- ① Dockerfile 작성 및 실행 확인

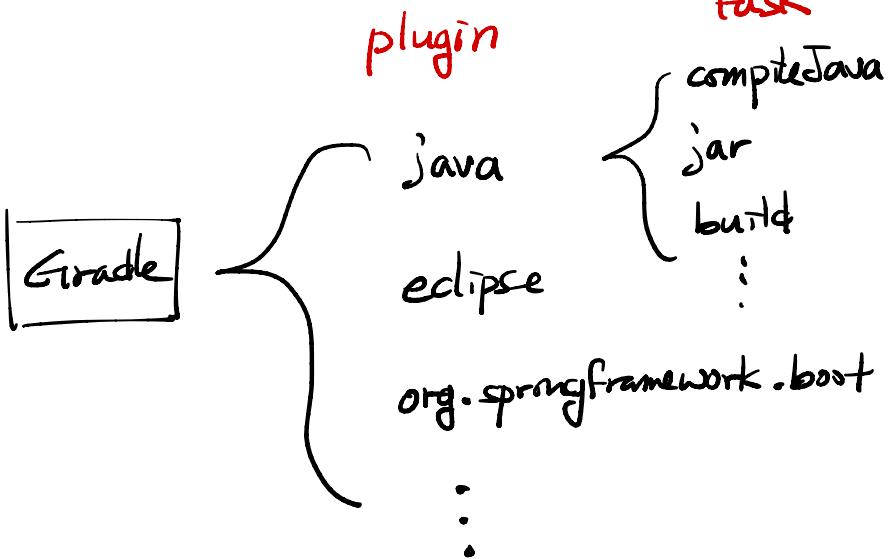


\$ gradle build --console=plain

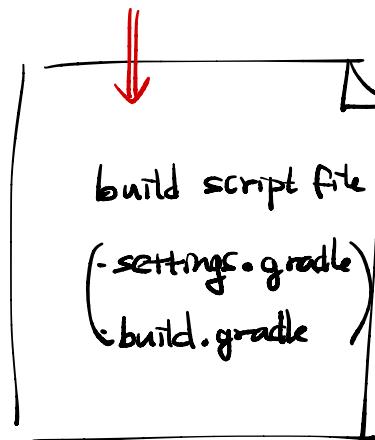
\$ java -jar .../myapp.jar

task ~~MyTask~~  
↓

\* Gradle - plugin - task - build script file



(프로젝트  
구조) on chit 보정



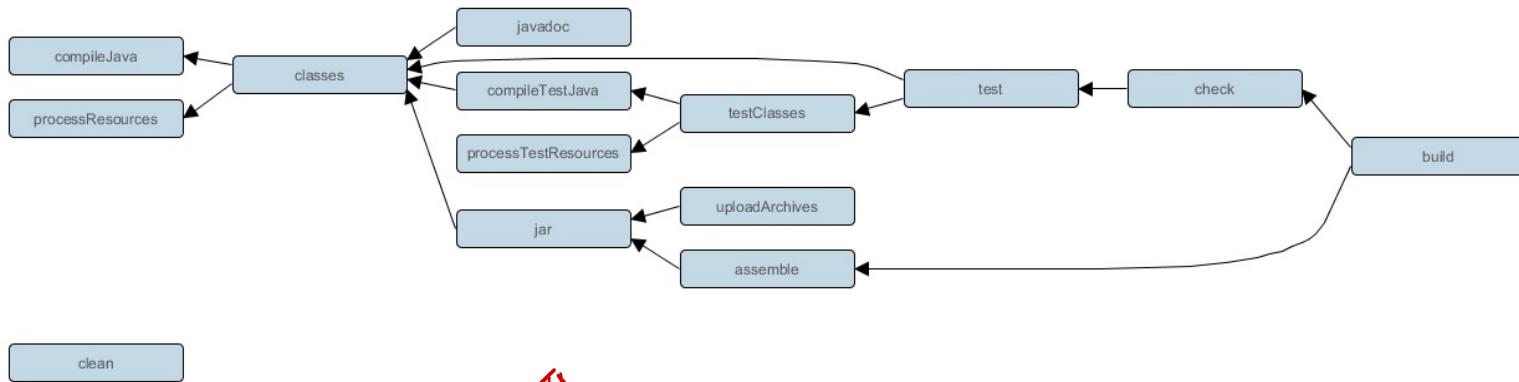
java {} - }

bookJar {} - }

:

\* Gradle 키워드

\$ gradle build  
build task의 종속성도 포함된 디펜던시 표시



'java' gradle 툴과 같이 태스크와 종속성이

70. Jenkins et Docker을 이용한 배포 자동화

② 스프링부트 설정 파일을 원형과 맞는 모드로 읽기

src/main/resources

application-dev.properties

application-prod.properties



Spring config

(dev  
prod)

\$ java -Dspring.profiles.active=dev jar myapp.jar  
JVM argument

\$ java jar myapp.jar --spring.profiles.active=dev  
Program Argument

\* Program argument & JVM argument

#java  $\Rightarrow$  m<sub>1</sub>D<sub>1</sub>

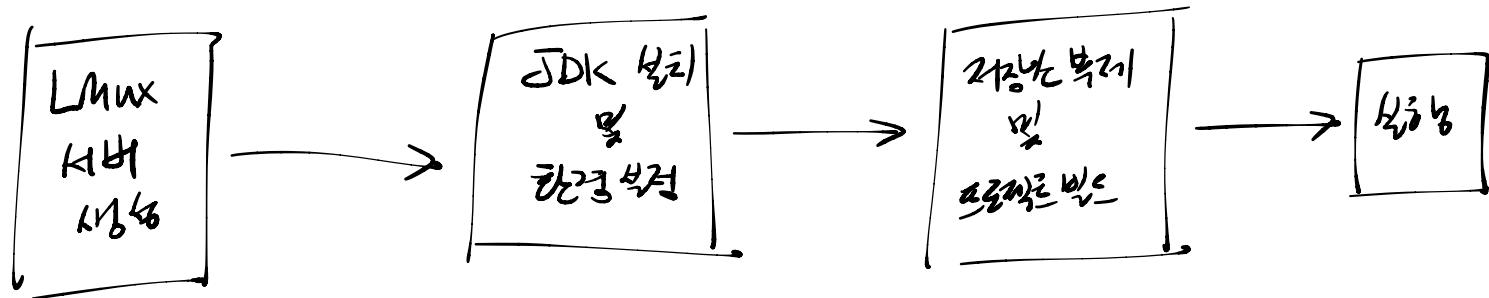
Java (space)  
Java  
m<sub>1</sub> D<sub>1</sub>  
Program Argument

#java -Dm<sub>1</sub>=D<sub>1</sub> -Dm<sub>2</sub>=D<sub>2</sub>  $\Rightarrow$  m<sub>1</sub>D<sub>1</sub>

JVM Options

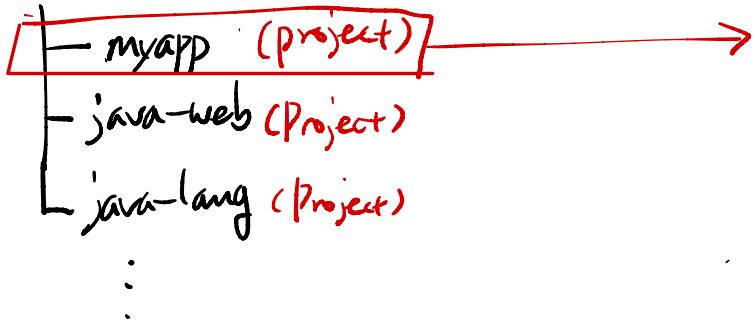
## 10. Jenkins et Docker을 이용한 배치 활용

- ③ 클라우드에서 Jenkins 서버 구축의 애플리케이션 배치 파일 생성



\* 언제 Repo 할지

bitcamp-mystudy (Repository)



언제 Repository 할지

- java-lang은 끝까지 기다려
- java-web은 빨리 다루기 끝내.

pull or push 는

Repo. 단위로

작동된다

↳ 파일/파일夹을 전

