

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 스프링 아�플리케이션으로 뷰를

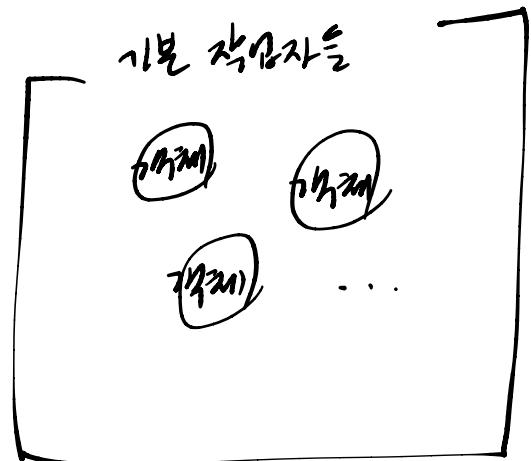
- ③ " 웹으로 뷰를

- ④ " Application 뷰로 AppConfig 123

- ⑤ " 콘트롤러로 뷰를

- ⑥ " IoC 컨테이너로 뷰를

* 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

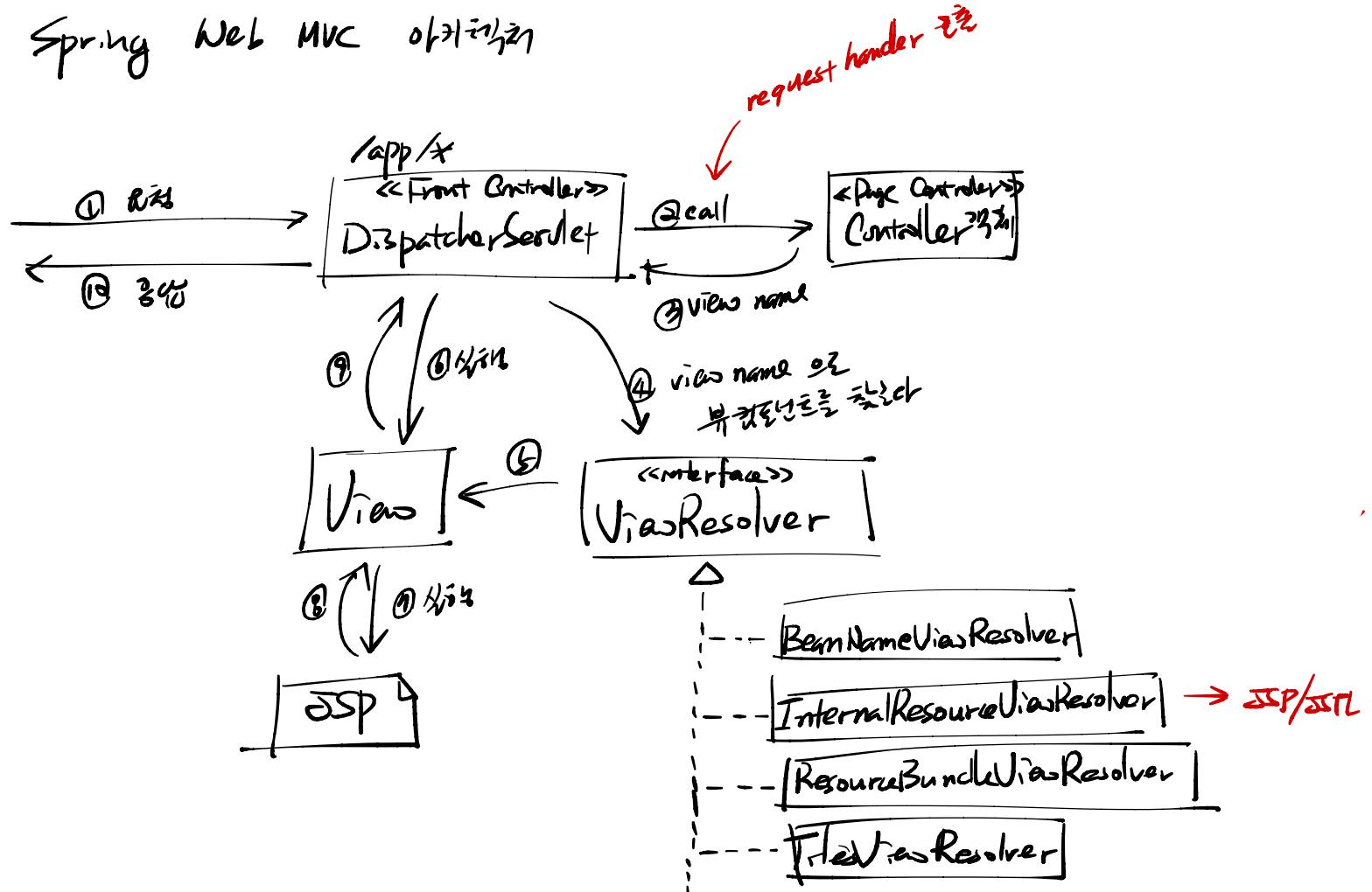
String

login(String email, String password) {

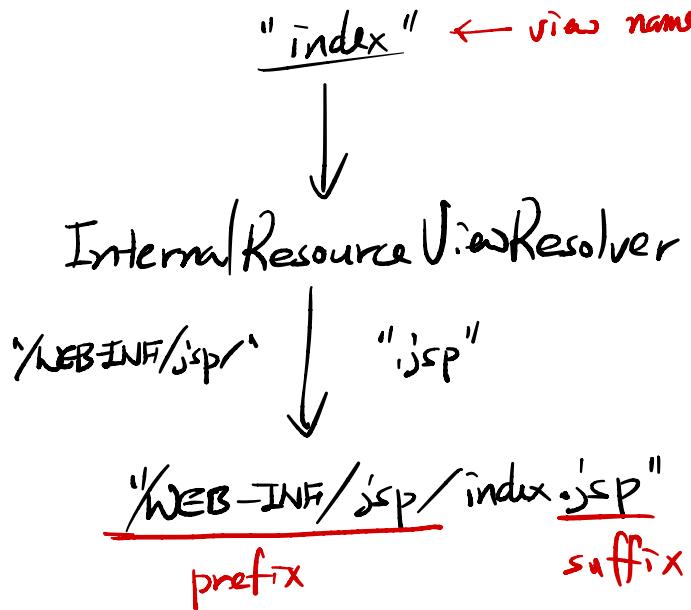
요청 헤더의 자세히 봐

}

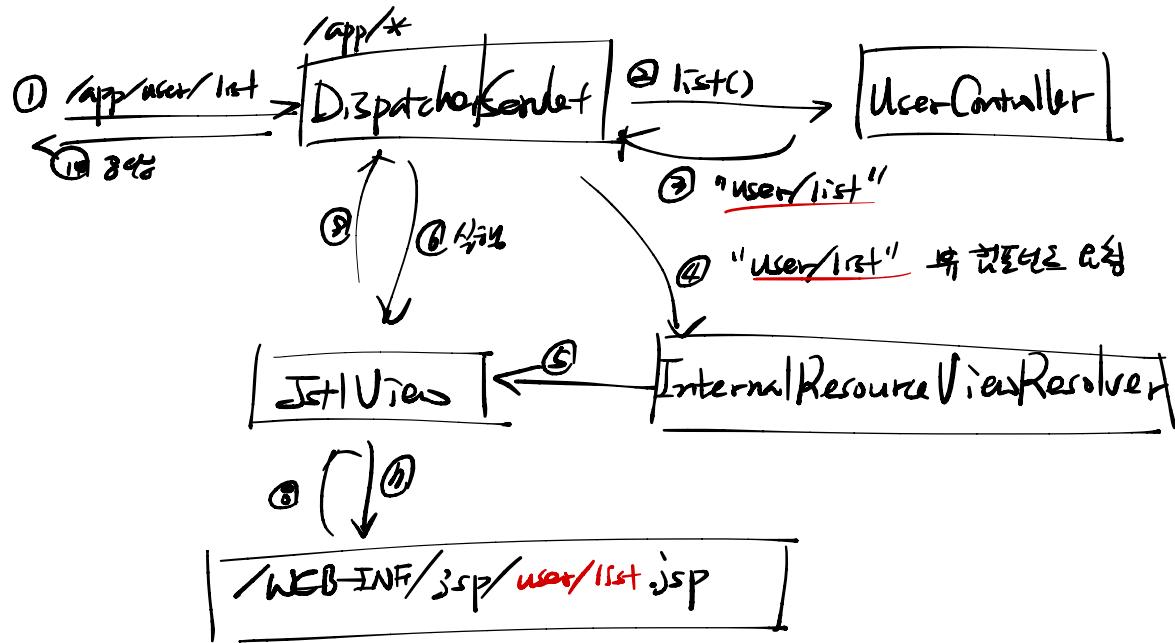
* Spring Web MVC 07/27/21



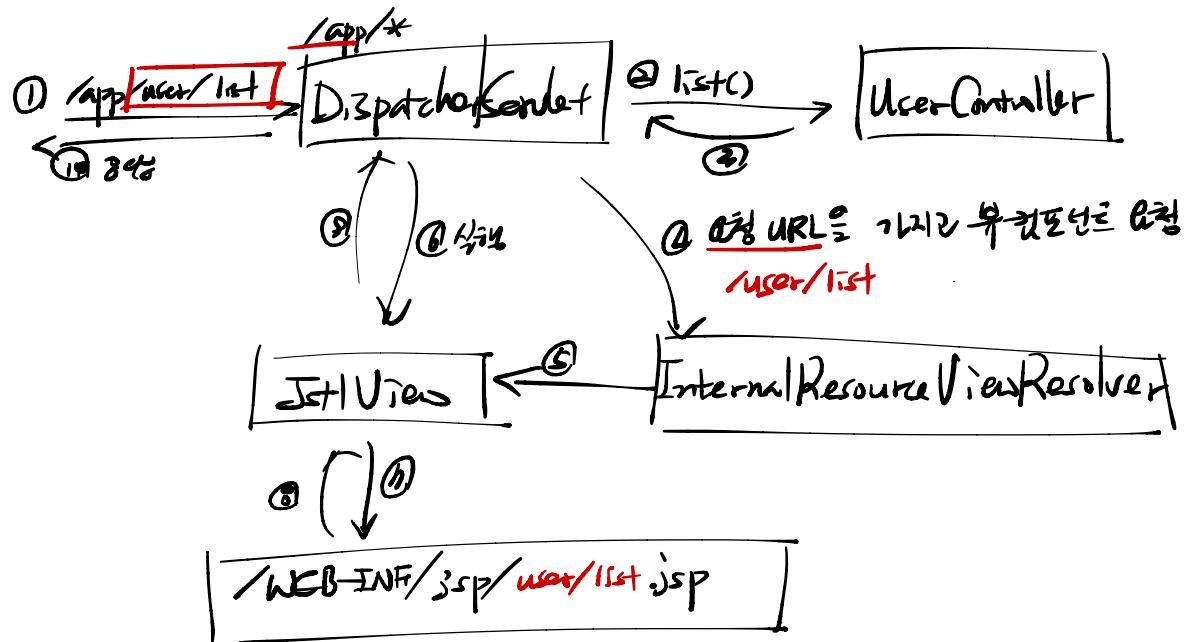
- * Internal/Resource ViewResolver



* view name %*% can

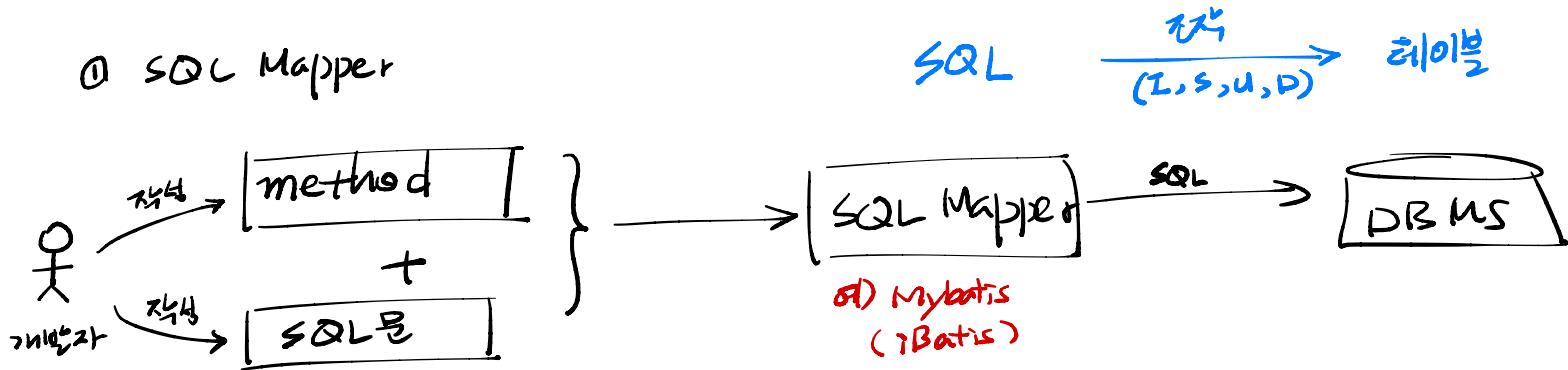


* view name $\stackrel{\text{보통}}{\rightarrow}$ controller

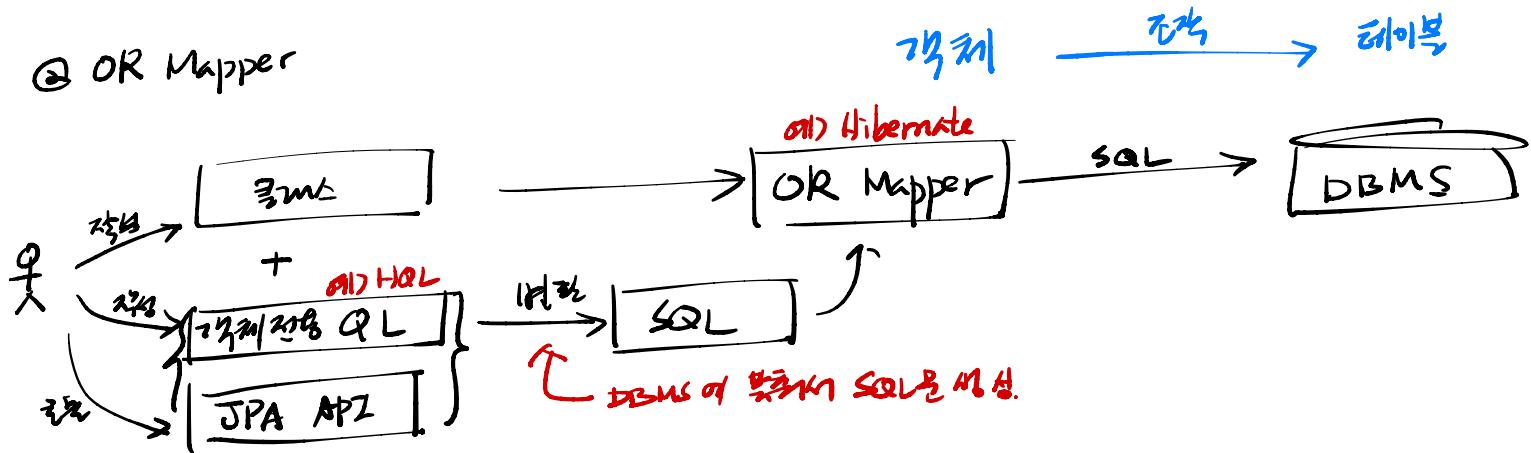


* SQL Mapper vs OR Mapper

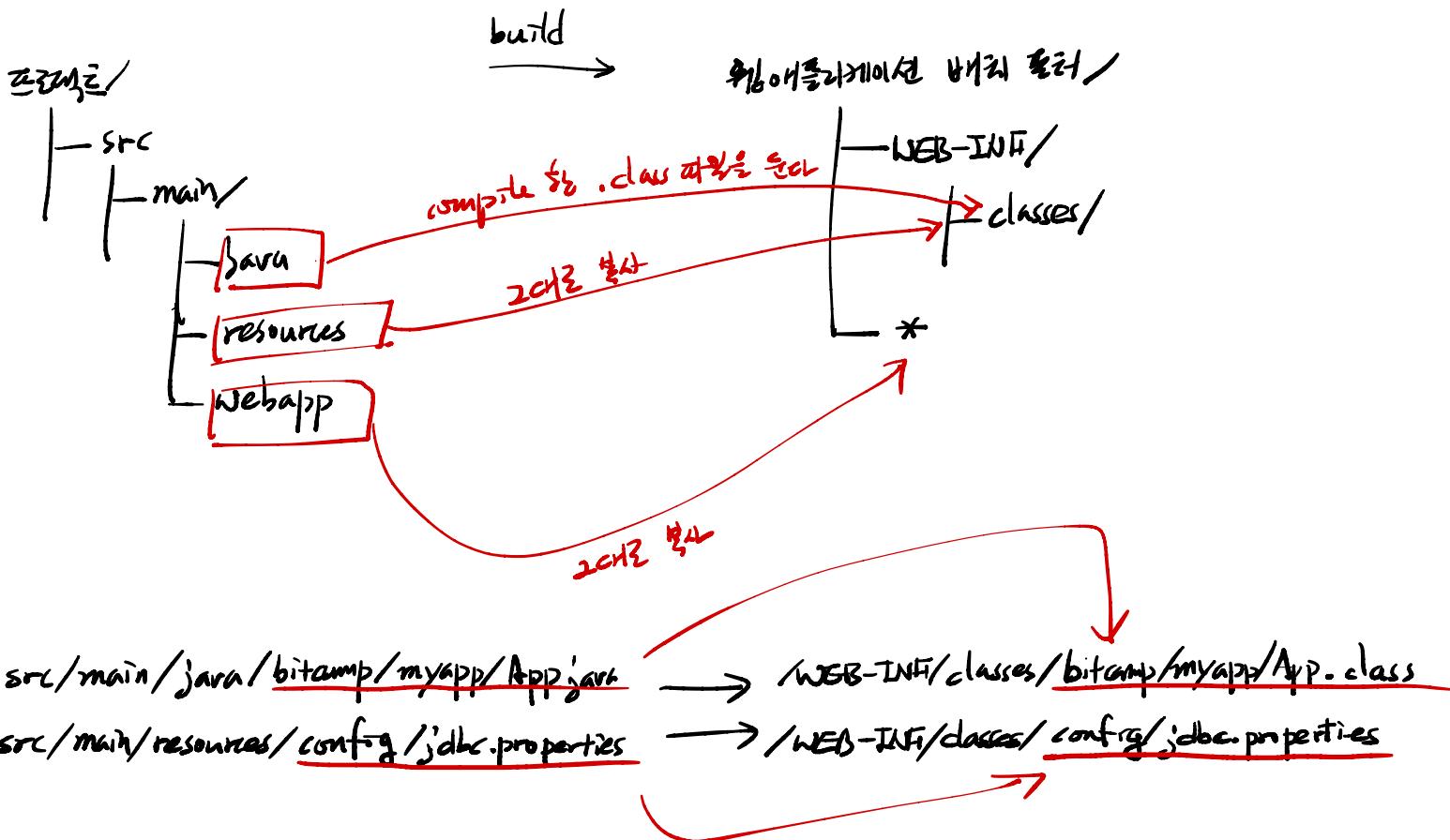
① SQL Mapper



② OR Mapper



* build et classpath

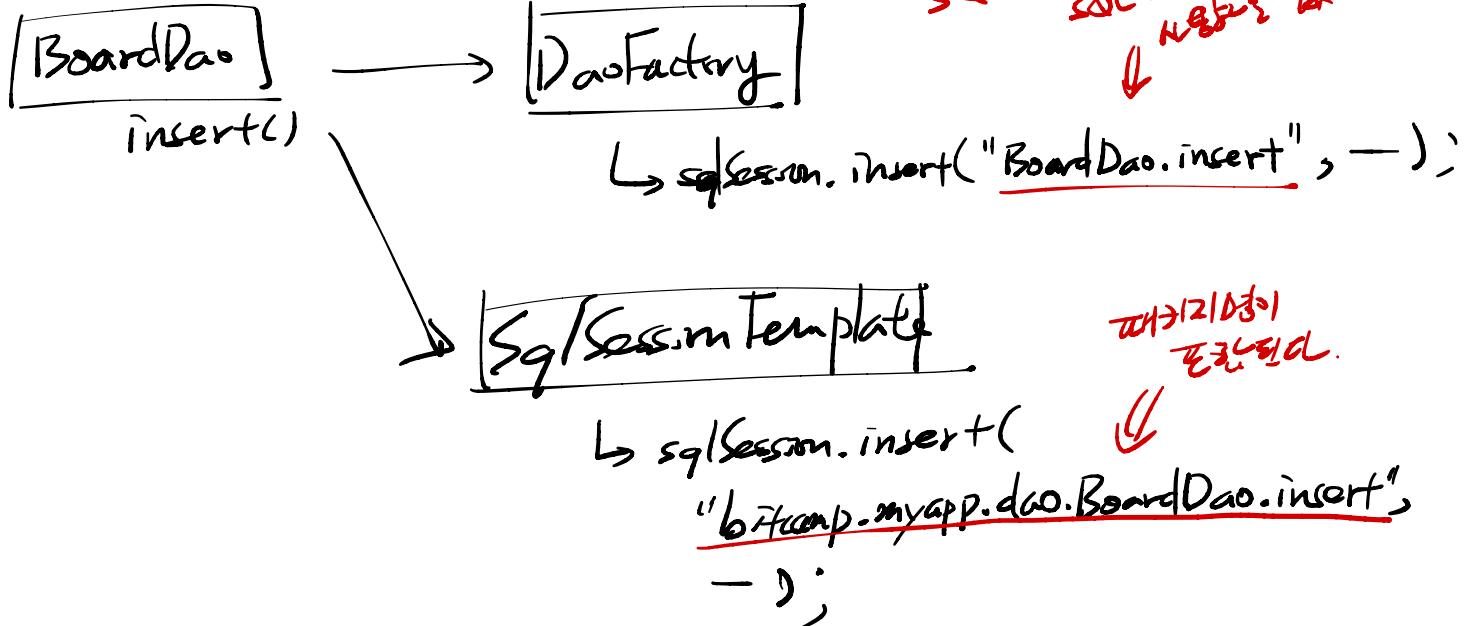


* DaoFactory 4

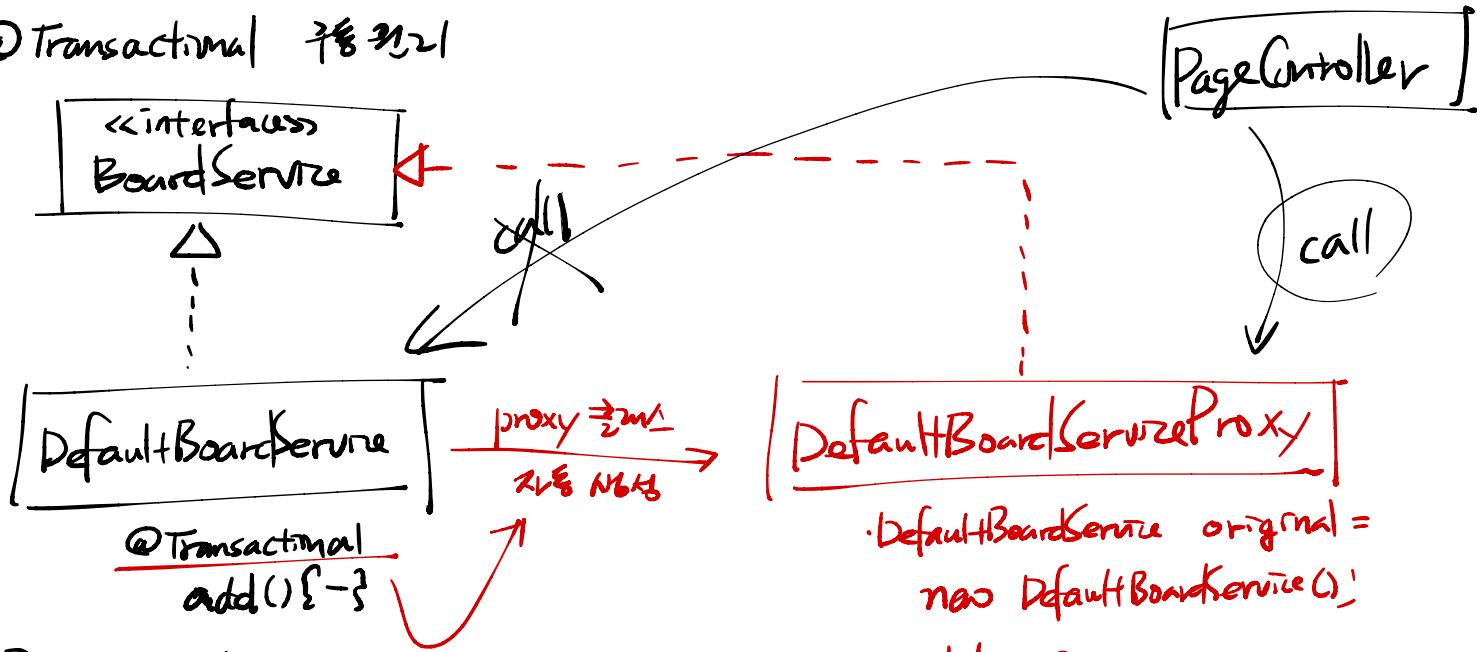
↑ graduate
DAO 04/24

SqISessionTemplate

Myobatrachus嘉陵江
Dorsal 鳃孔



* @Transactional 78 31, 21



* AOP의 특징, 예제

↳ 기존 코드를 대체하지 않고
추가 기능을 적용하는 방법
↳ proxy 패턴을 활용하는 방법 ("proxy 를 써라")

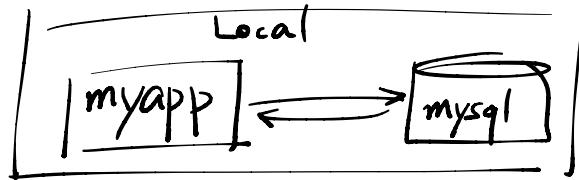
특징
단점

```

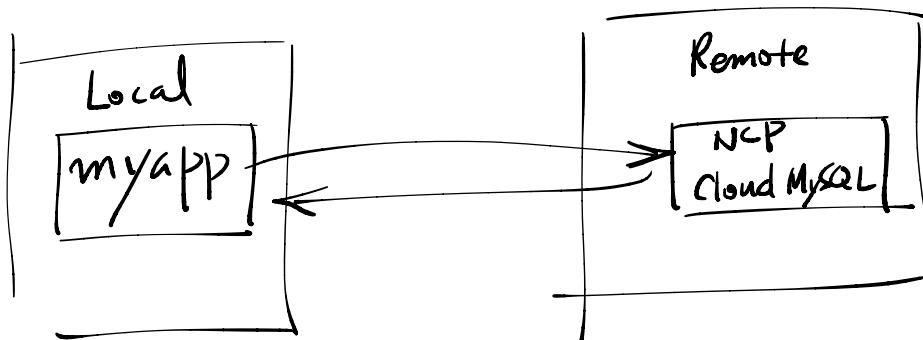
    try {
        original.add();
        txManager.commit();
    } catch {
        txManager.rollback();
    }
  
```

62. Naver Cloud Platform 사용 예

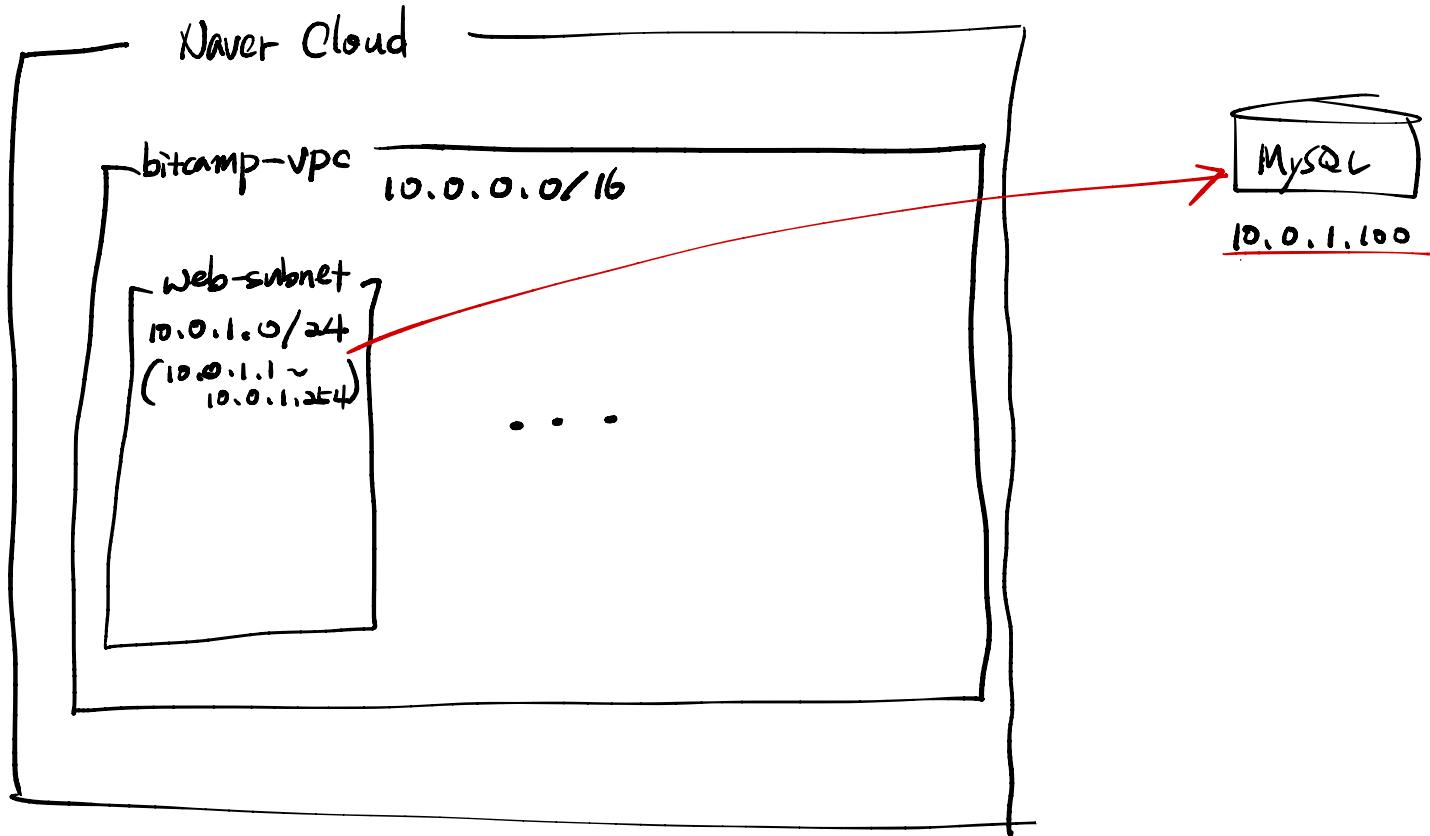
* ORI



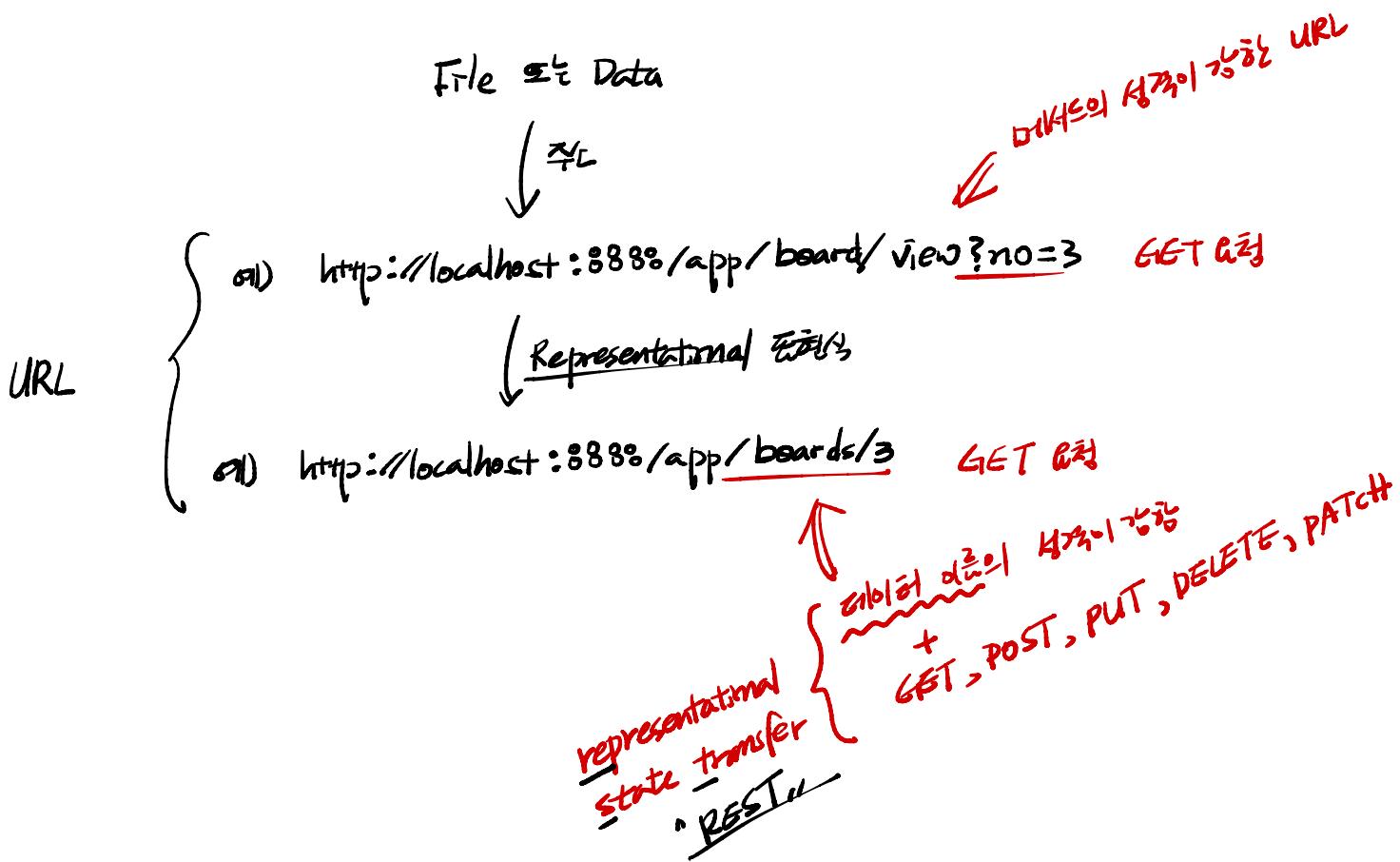
* 대입



* VPC (Virtual Private Cloud) - 퍼블릭 네트워크 대신 퍼비전 네트워크를 사용하는 자체적인 네트워크 환경



* RESTful API



* function → REST API

다른 프로토콜

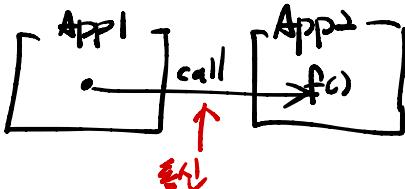
① function

(RPC)

② Remote Procedure Call

(RMI)

③ Remote Method Invocation



C 등 프로그래밍 언어

기법의 암호화



App의 행동을 분석 (분석 컴퓨터)



한 App이 하는 일을

여러 App으로 번역해 보내는

App 한자 행동이 생략된다



다른 다른 App의
function을 수행할 수 있는
기술이 등장하게 된다.



C++ 등 프로그래밍 언어



ODP 프로그래밍의 특징이 빌드

RPC를 개선

* function → REST API

③ RMI → ④ CORBA

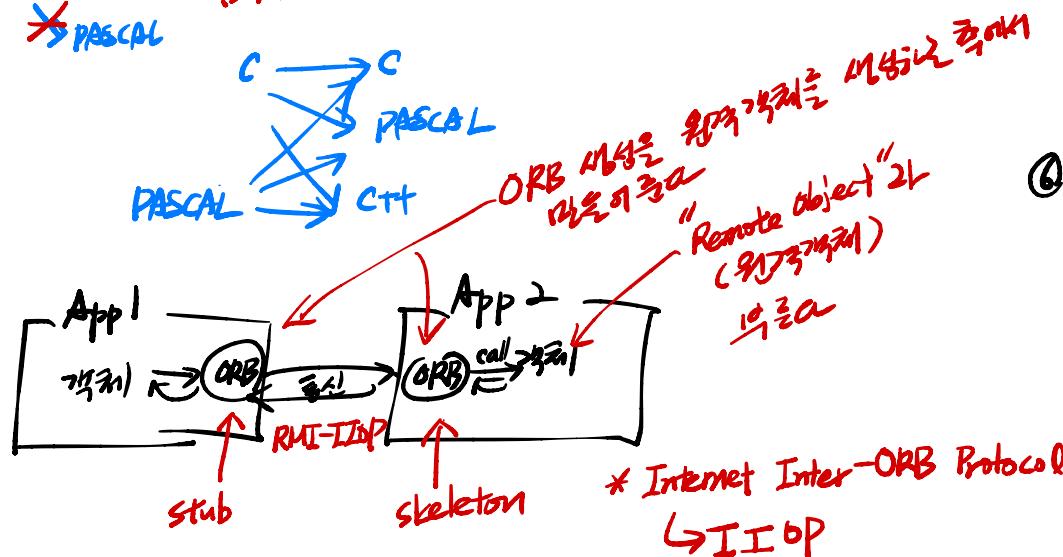
같은 인터페이스
같은 App끼리
같은 환경에서
만나 가능

common
Object Request Broker
Architecture

자신의 인터페이스를
다른 App끼리
같은 환경에서 만날 수 있는
방법을 찾았던 거지

C → C
X PASCAL

C → C
X PASCAL
X C++



⑤ Web-service

✓ 파일과 HTTP 프로토콜을 활용
✓ ORB를 통과해 인터넷에 맞춰
설정되었음

↓
인터넷 환경에 맞도록 개발된
방법을 찾았지!

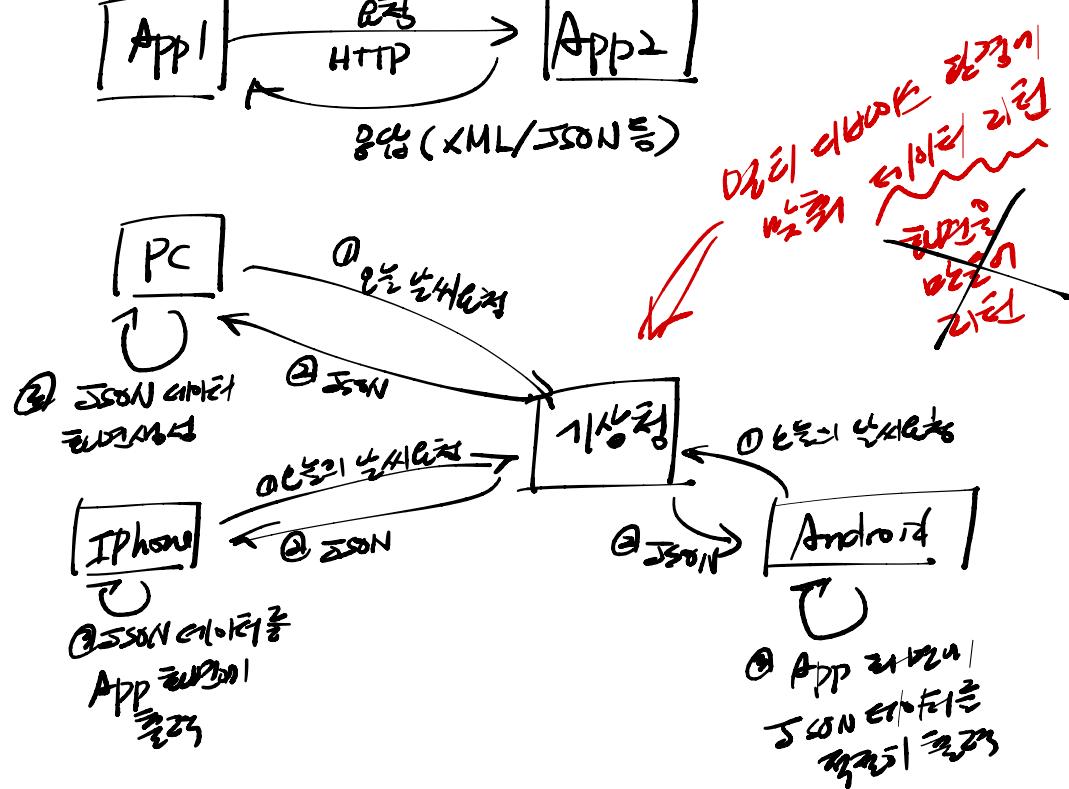
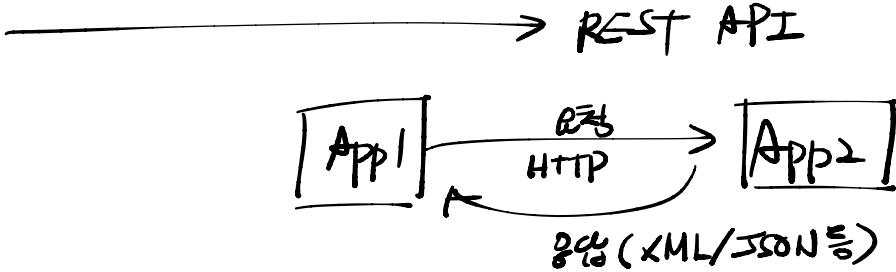
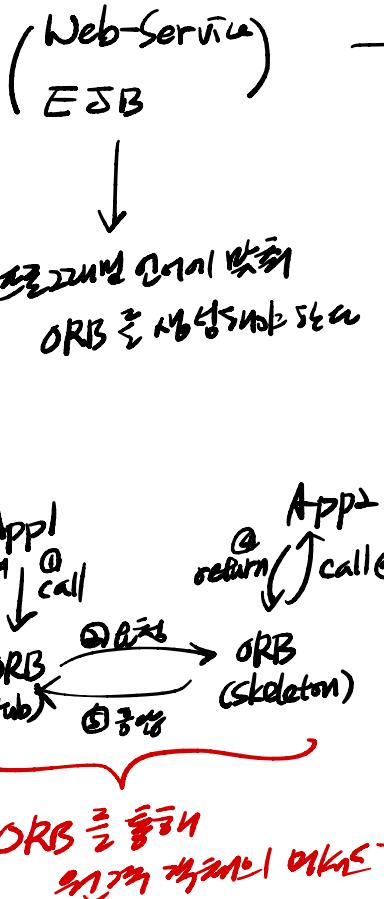
⑥ Enterprise JavaBeans (EJB)

자바언어 기반 EJB

단점!

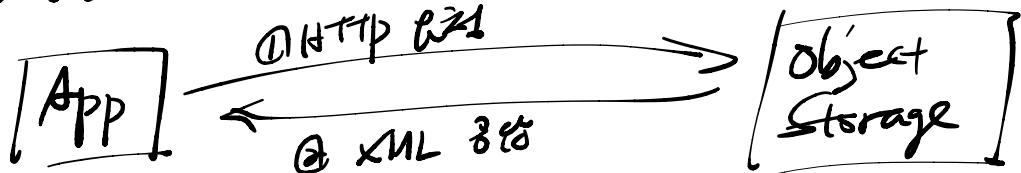
2000년 초반에는 PC를
대상으로 디버깅을 위한 IDE가
제작되었지만 초기에는 사용되지 않았다.

* function → REST API



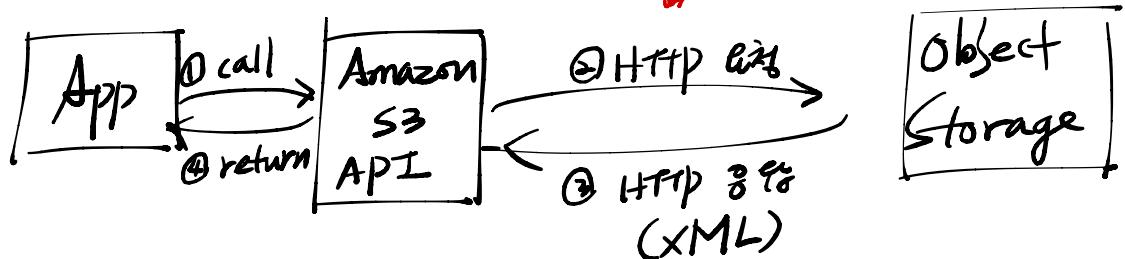
* Object Storage REST API API
↳ HTTP 퀼/값, 키!

① 직접 접근

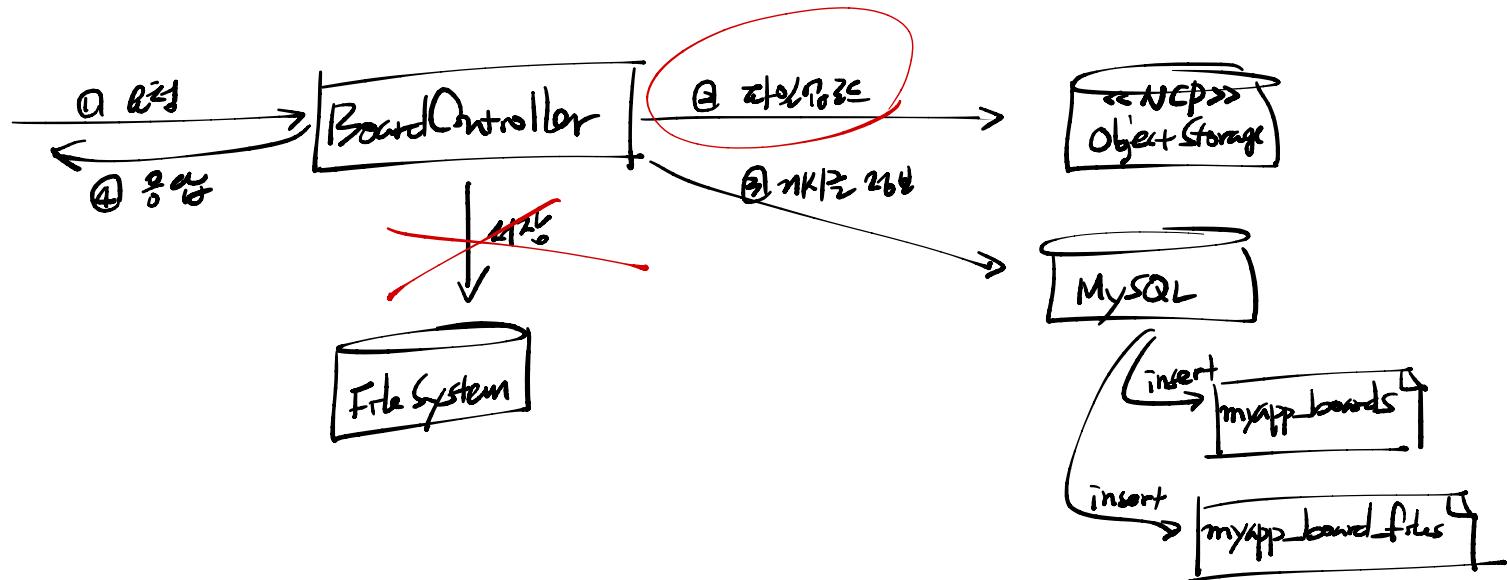


HTTP 퀼/값은 직접 접근하기 편리하다
HTTP 응답을 직접 접근하기 편리하다

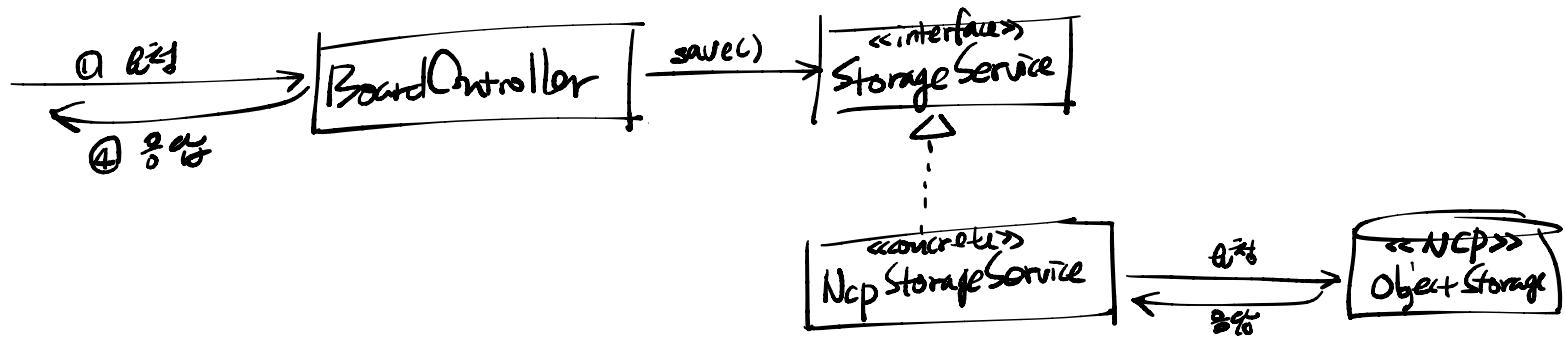
② 중간 관리



* 내부로 첨부파일은 NCP의 Object Storage에 저장

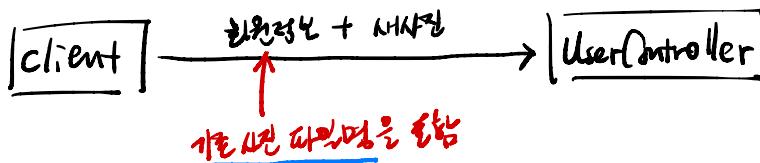


* 게시글 첨부파일은 NCP의 Object Storage에 저장 → NCP 연동로직을 서비스 객체로 분리



* 사용고객과 보안

① 보안에 악용은 예 : 회원정보 변경



- i) 기존 사용자계정 → 클라이언트가 보낸 세션키로
회원을 조작해 사용자 계정
- ii) NH 계정 정보 → NH 계정 조작
- iii) 정보변경 → DB 변경

A 회사는 사람이 로그인 한 후에
마음의 정보를 바꿀 때 사용자계정을
다른 사람의 것으로 치환할 수 있다?

↳ 다른 사람의 계정을 사용할 수 있다.



이제, 변경, 삭제 시

기존 정보를 사용할 때는

항상 새비밀번호를 사용하라!

기존 계정정보를 클라이언트가 조작해
할 수 있다! → 즉, 이제, 변경, 삭제할 때
새비밀번호를 입력해야 한다!

* 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

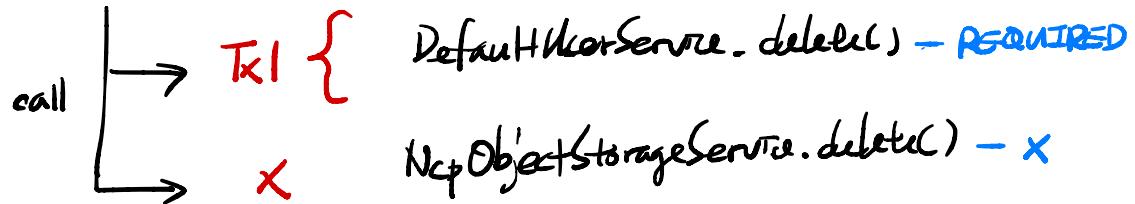
UserController.delete() X
↓ call
Tx1 { DefaultUserService.delete()

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

* 키워드 어노테이션 정리

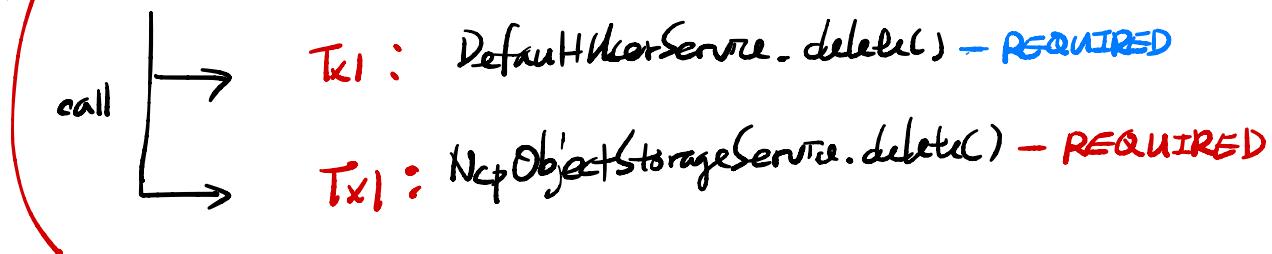
① 허용

X UserController.delete() X

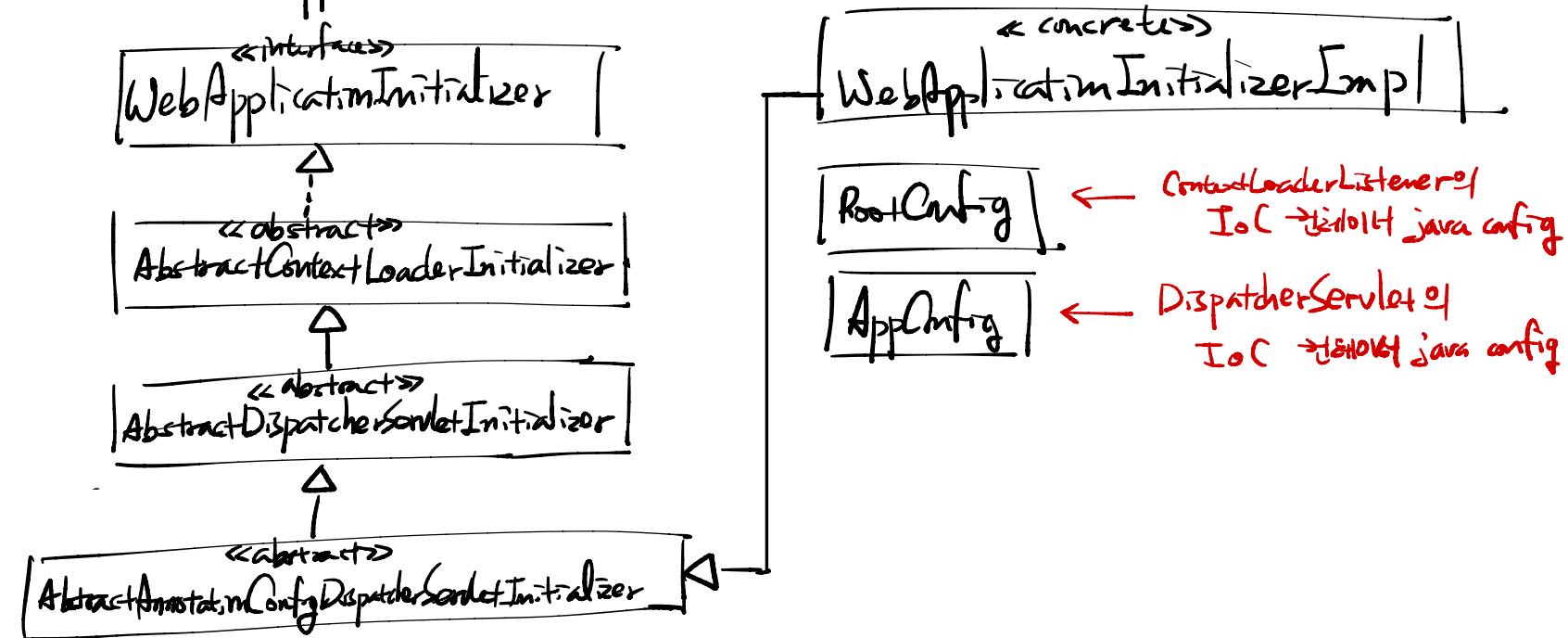


② 허용

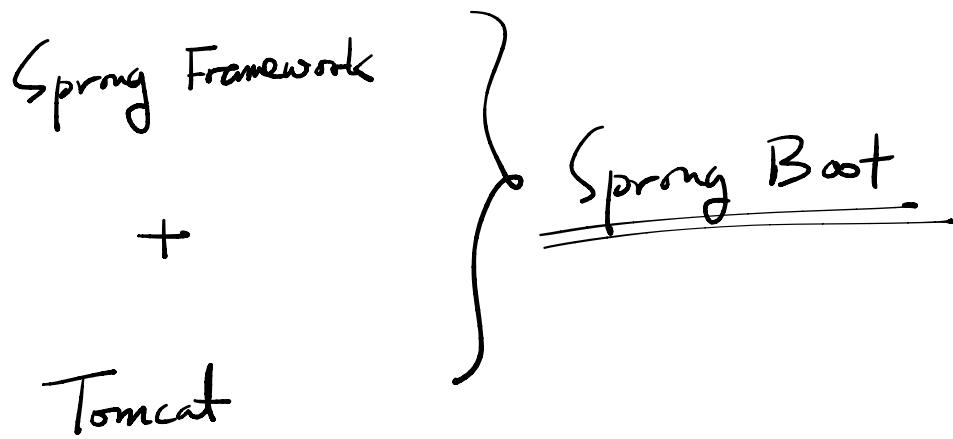
Tx1 UserController.delete() - REQUIRED



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 215012 |
| 1 | _____ | |
| 2 | _____ | |
| 3 | _____ | |
| 4 | _____ |) 2 215012 |
| 5 | _____ | |
| 6 | _____ | |
| 7 | _____ | |
| 8 | _____ |) 3 215012 |
| 9 | _____ | |
| 10 | _____ | |
| 11 | _____ | |
| 12 | _____ |) 4 215012 |
| 13 | _____ | |
| 14 | _____ | |
| 15 | _____ |) 5 215012 |
| 16 | _____ | |
| 17 | _____ | |

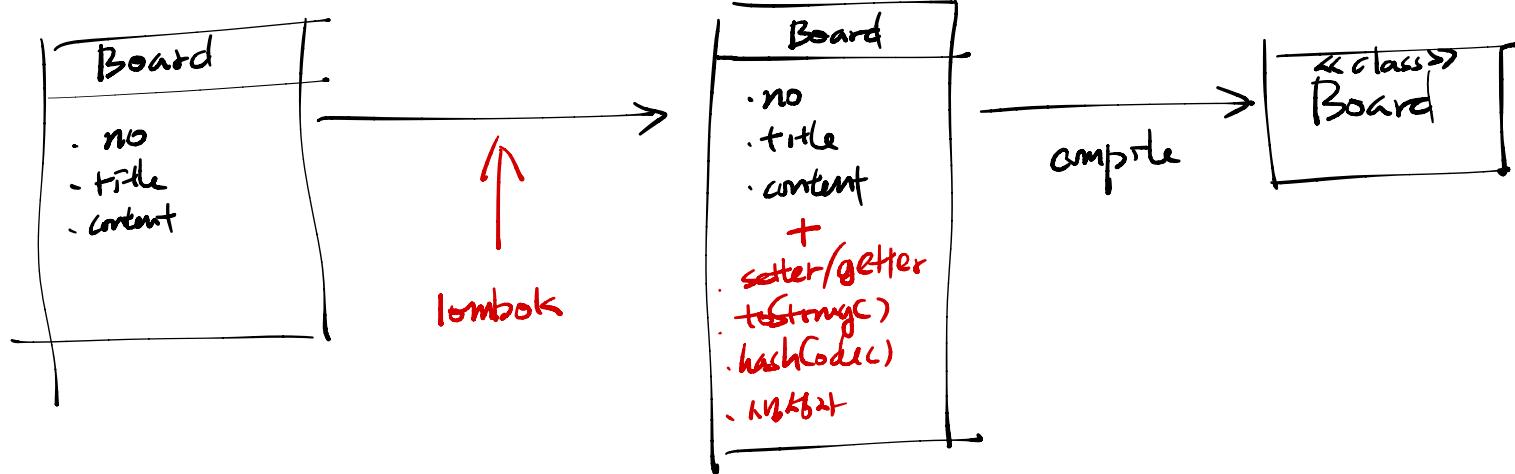
67. Lombok 2010-2021年版

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

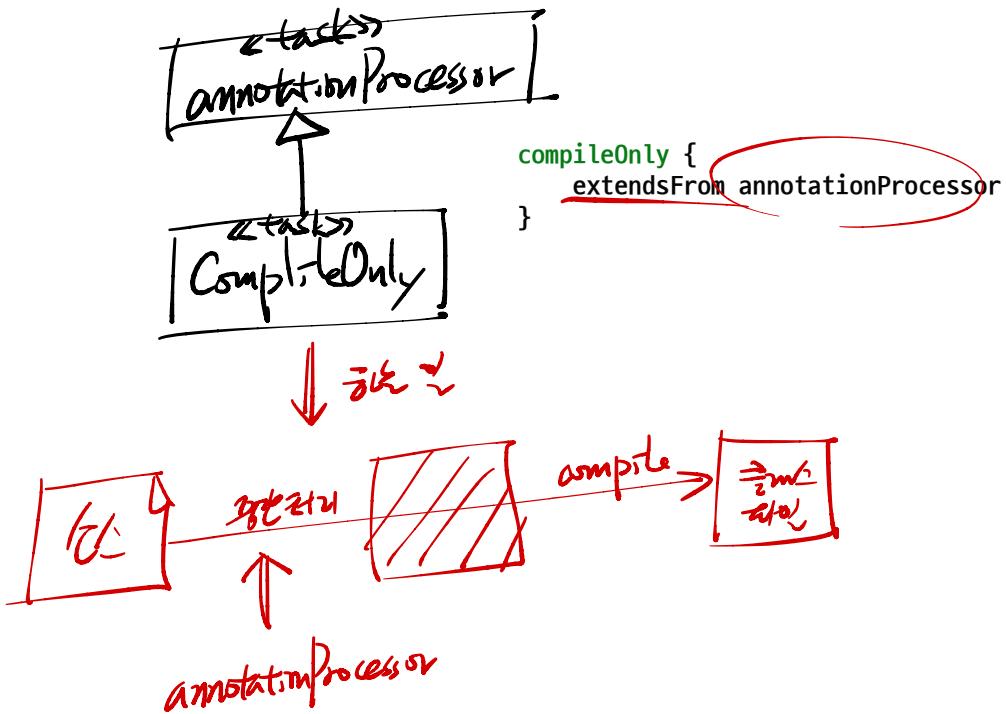
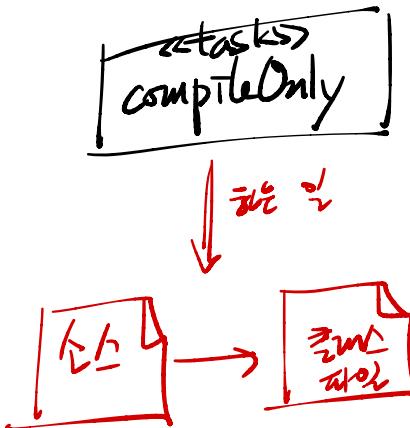
Gradle



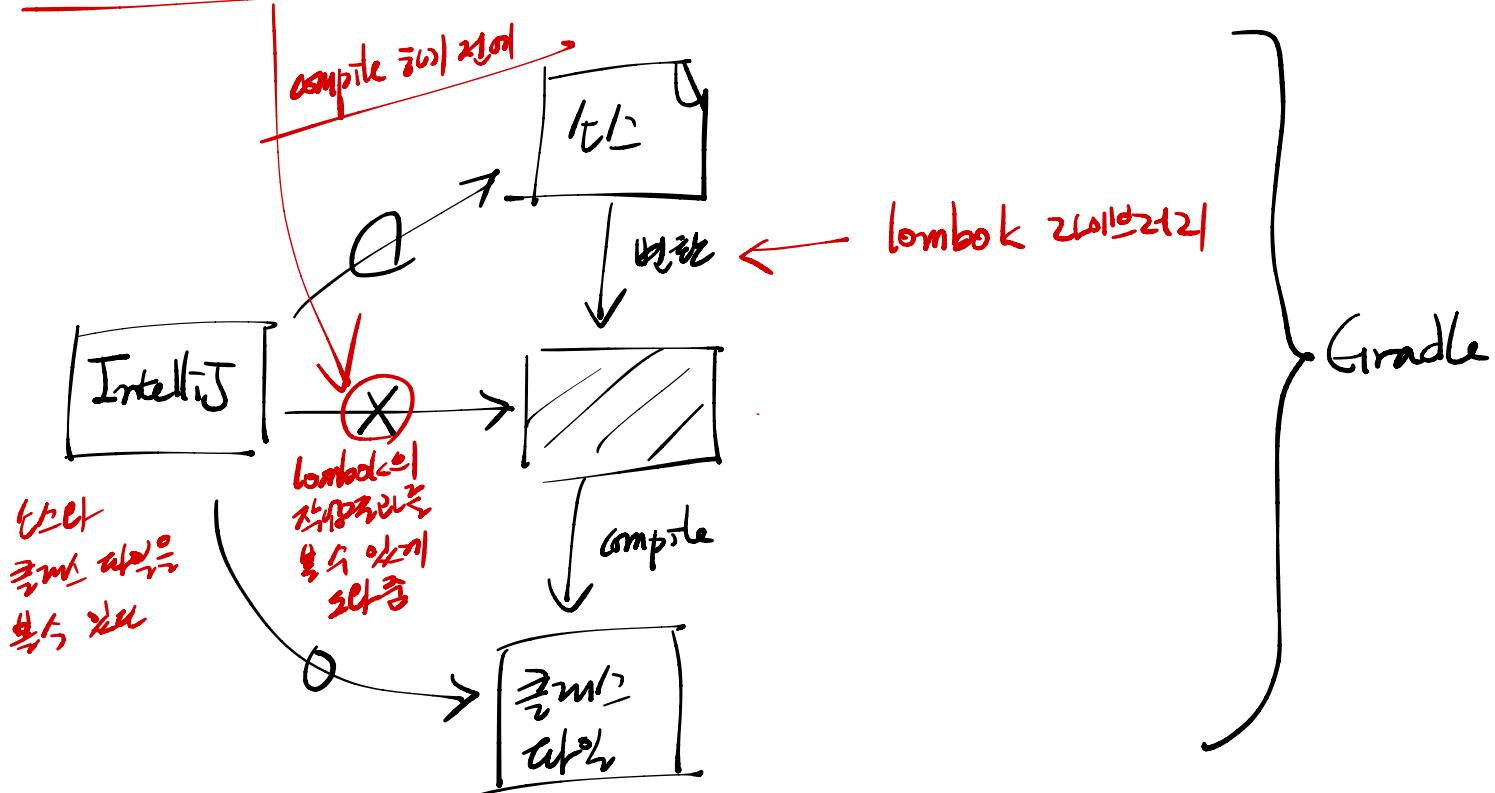
Build



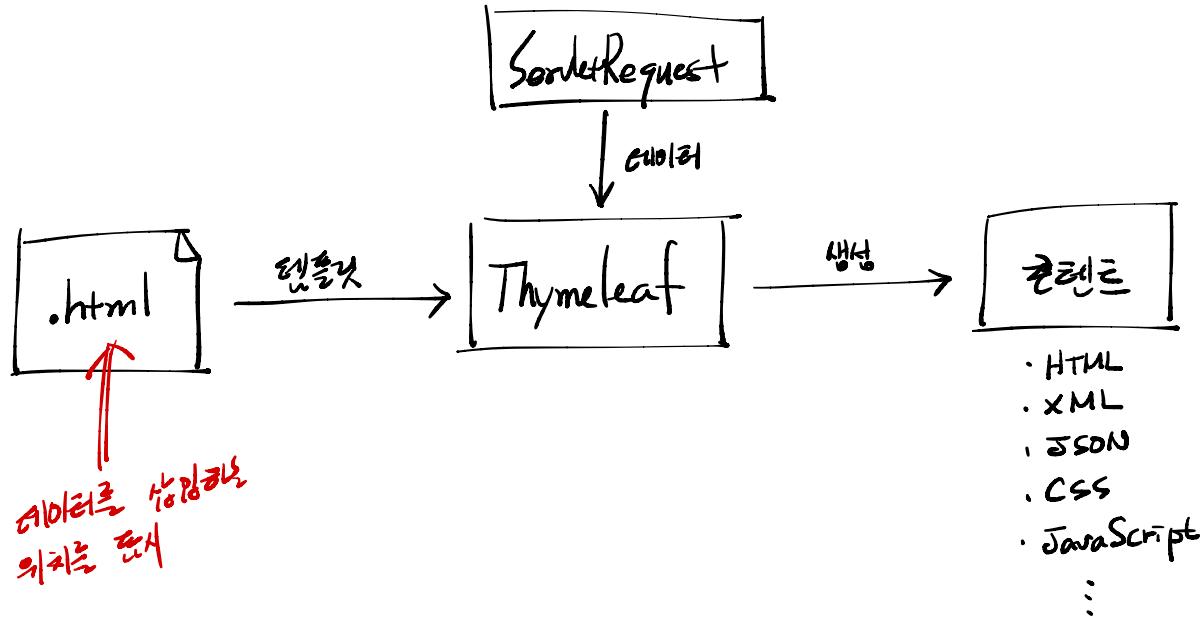
* Gradle et Task



* lombok IntelliJ 풀이방법의 차이



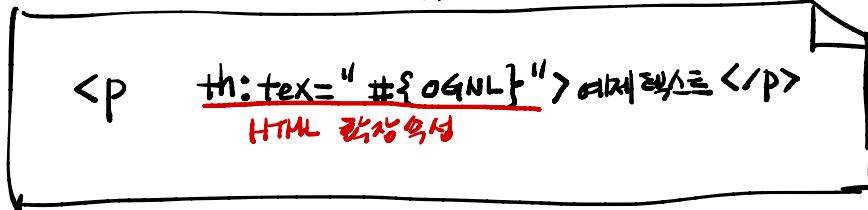
* Thymeleaf 템플릿 인자



* Thymeleaf 템플릿과 HTML

.html

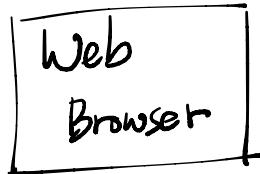
템플릿 파일



JSP → \${변수명}

Mybatis → #{} выражение

Thymeleaf → #{변수명}



HTML 편집과 편집을 허용



디자이너

디자이너는 Thymeleaf가
수동화한 코드에 맞게 편집
하고 편집을 확인할 수 있다

Thymeleaf의 특징

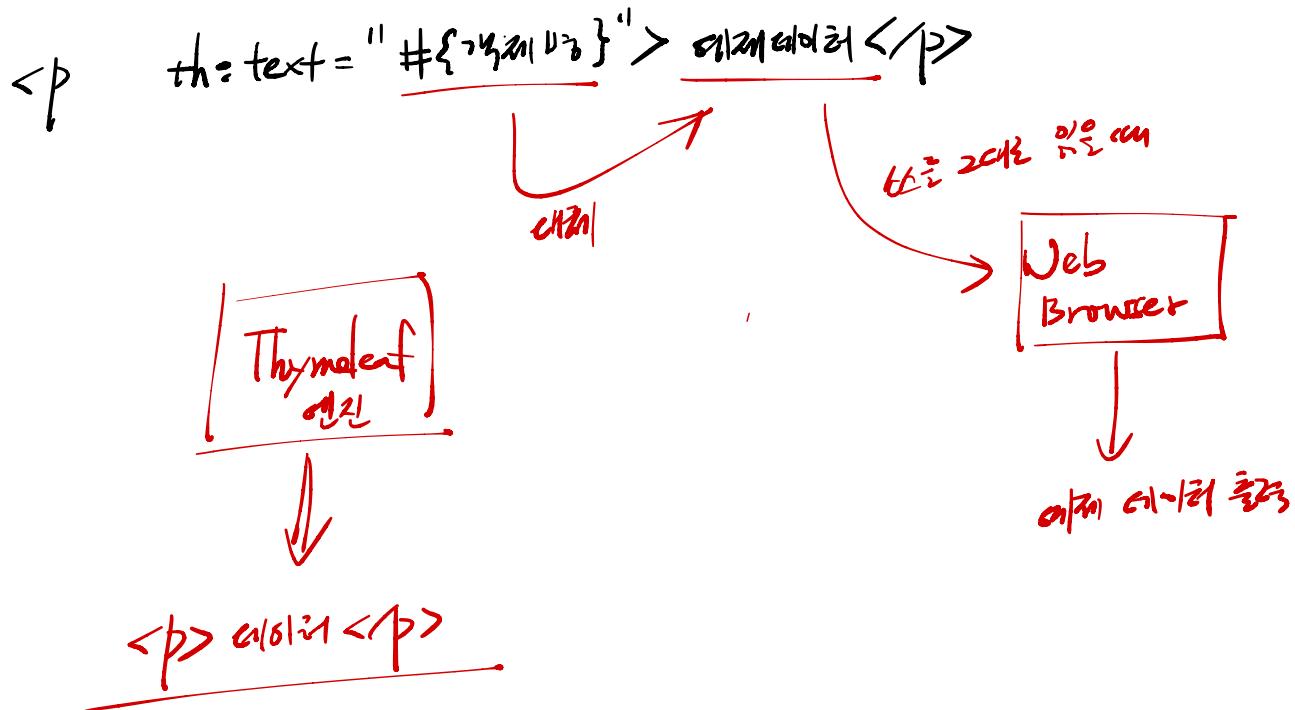
- ① JSP와 다르게 템플릿이 HTML 파일이다.

↳ 템플릿 언어의 사용성이
스스로로 웹브라우저에서
처리하도록 디자이너로
작성되는 수 있다.

- ② Thymeleaf 파일이 HTML 편집을 허용하지 않는다.

스스로로 편집은 대체로
하지 않다.

* Thymeleaf 템플릿 HTML



* HTML의 속성과 属性

<table> th: text = "국민당" <td> </td>

↑
국민당
국민당



<td> 13424 </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

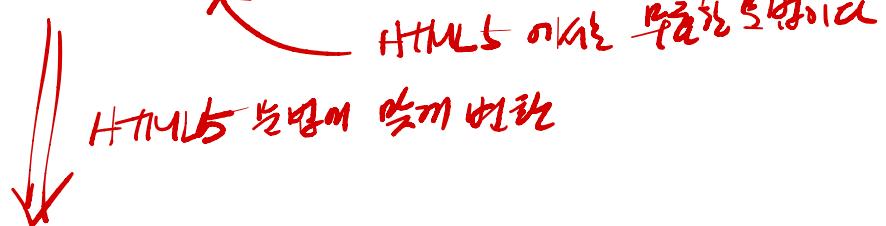
th:* 대화 속성을 만들기
Thymeleaf는 대화 속성을 템플릿으로 인식하여
처리하는 것이다

※ XML namespace 처리는
HTML 템플릿

th: 를 기본으로 인식한다
그것은 가능할 생각하지 않아!

* 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 = " \${welcome}"

"data-th-text"

* th-text vs th-utext

welcome = <u>안녕</u> 친구님.

data-th-text

escape 문자를 차단한다

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

< b > 안녕<u> / b > 친구님.

↑

HTML Entity

(HTML 특수기)

↳ HTML keyword와 종종나는 문자를
표현하기 위한 특수기로 사용된다.

} HTML 키워드가 아닌
단순 문자로 취급하기
위함.

data-th-utext

unescape

escape 문자를
차단하지
않는다.

< b > 안녕<u> / b > 친구님.

단일태그로만
HTML 특수기로
표현된다.

* Link

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

``

각각의 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
(설정을 통해 가능)