

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

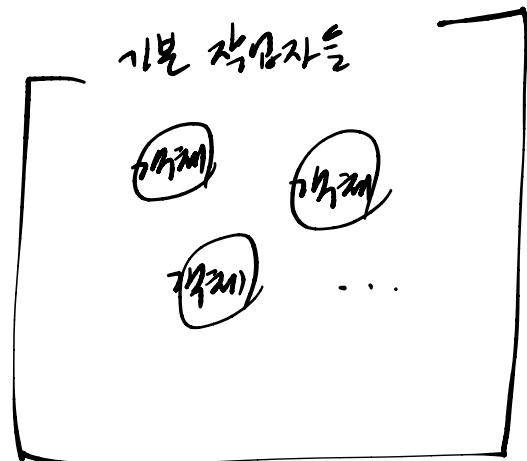
- ③ " 웹으로 뷰를

- 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

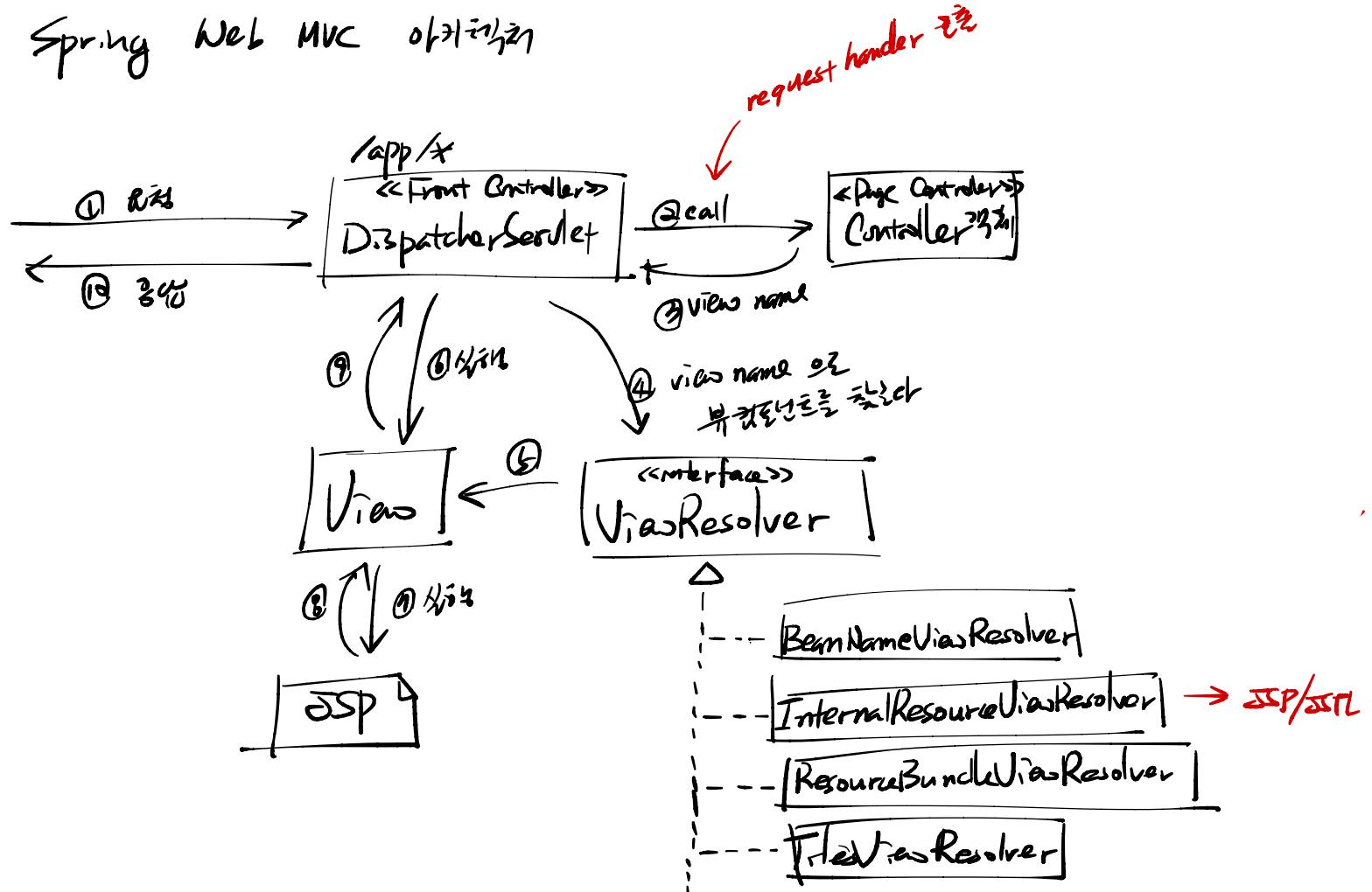
String

login(String email, String password) {

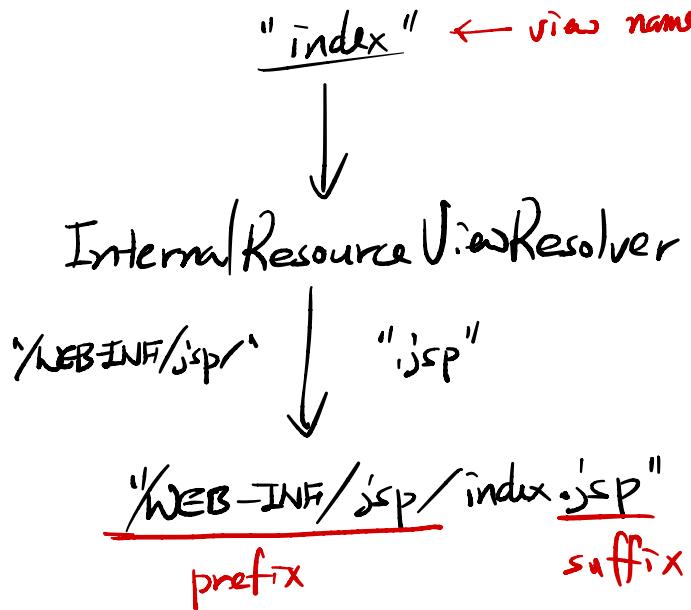
요청 헤더의 자세히 봐

}

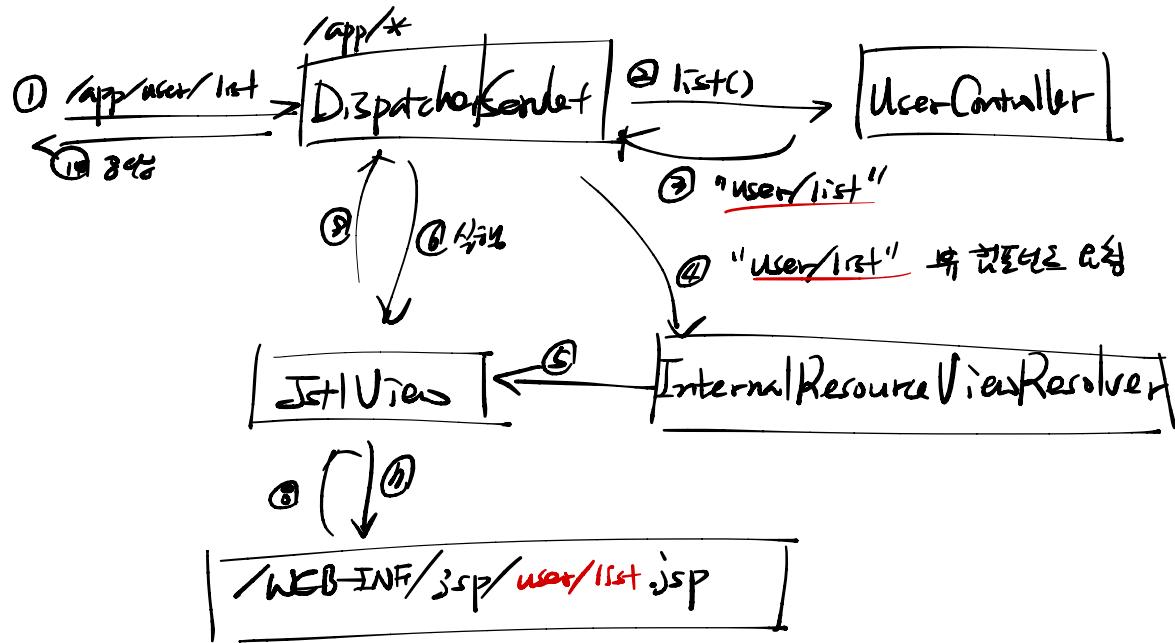
* Spring Web MVC 07/27/21



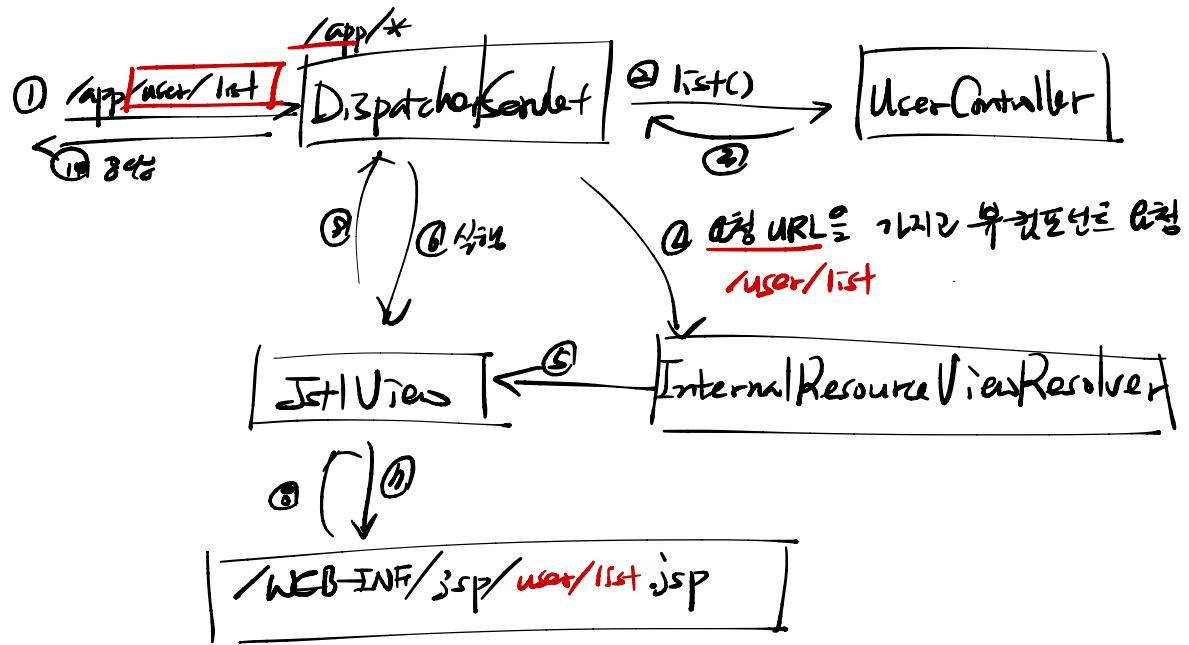
- * Internal/Resource ViewResolver



* view name %*% can

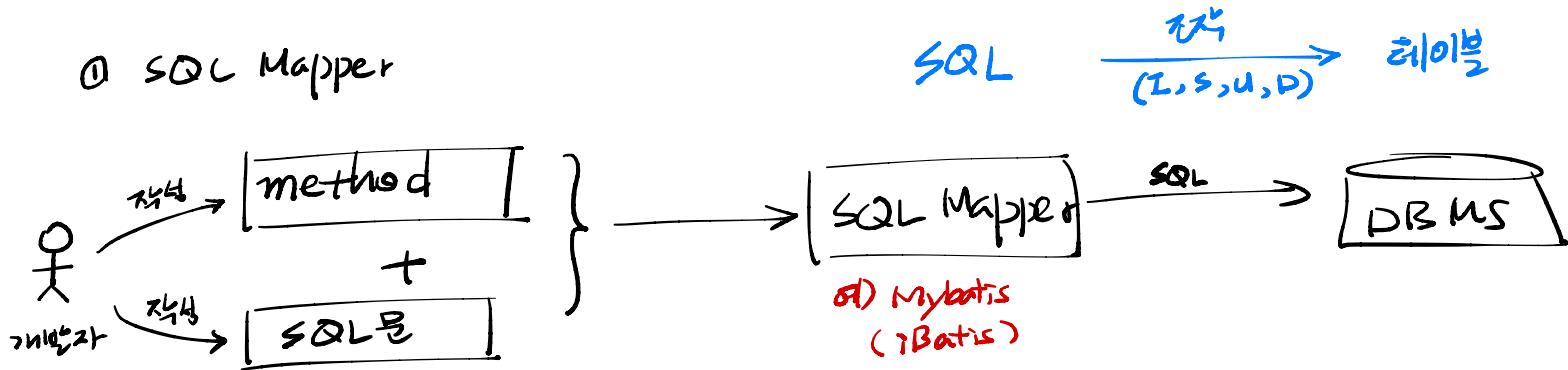


* view name $\stackrel{\text{보통}}{\rightarrow}$ controller

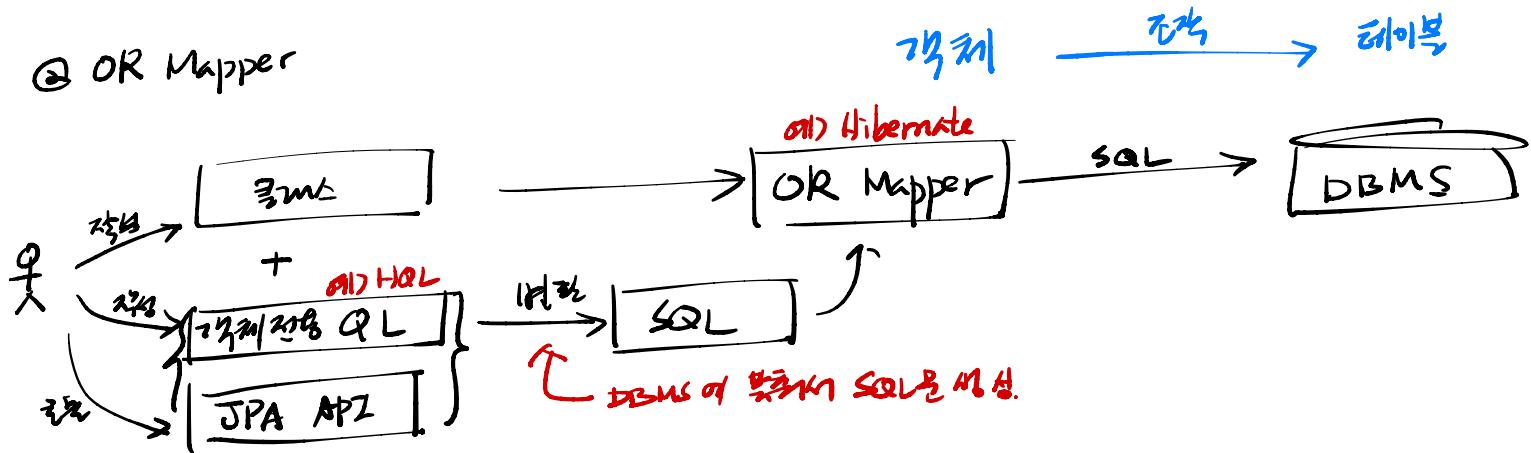


* SQL Mapper vs OR Mapper

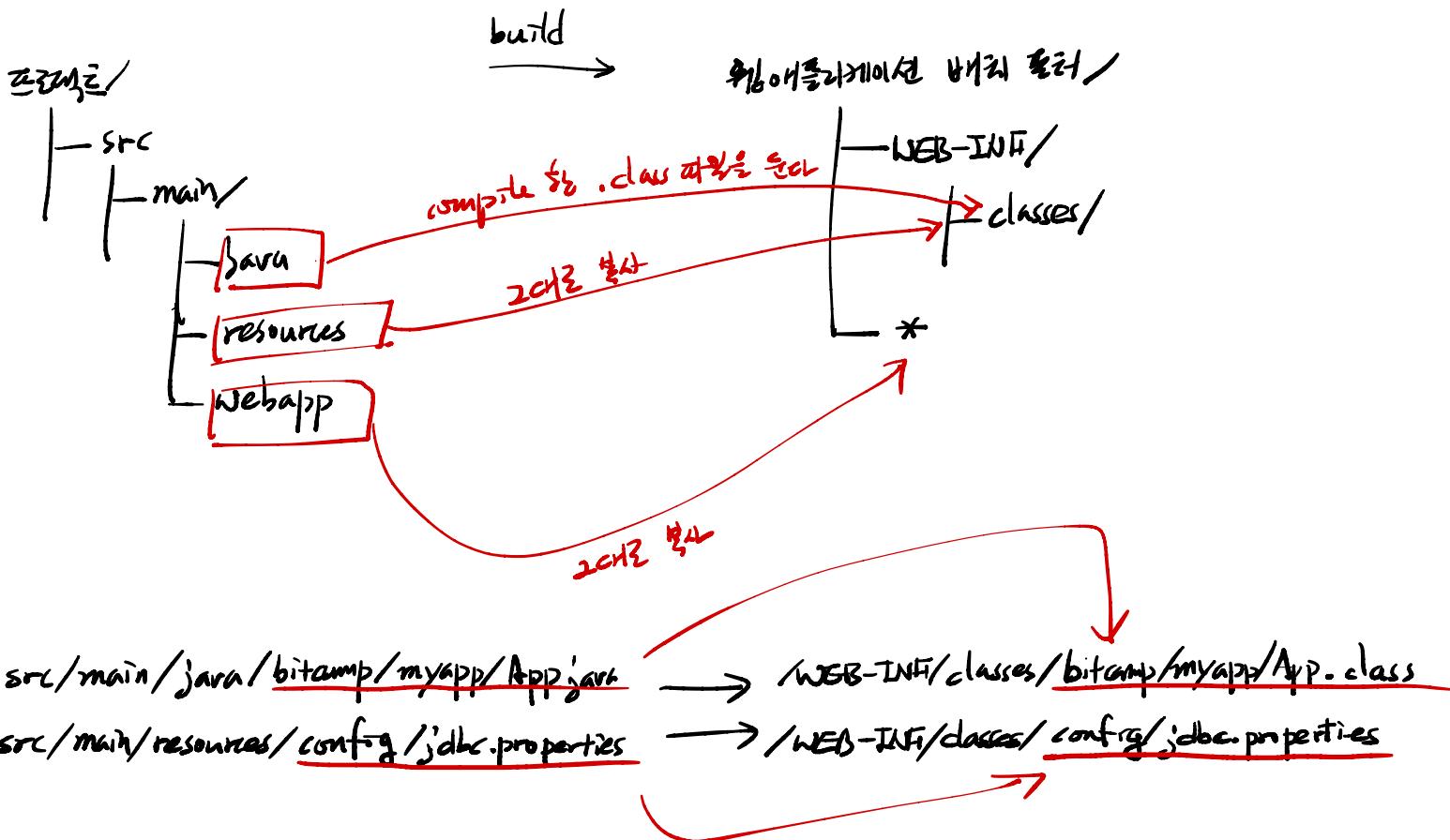
① SQL Mapper



② OR Mapper



* build et classpath



* DaoFactory

↑
생성
DAO

SqISessionTemplate

↑
Mybatis
가져온
DAO

[BoardDao]

insert()

[DaoFactory]

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

[SqISessionTemplate]

↳ sqISession.insert(

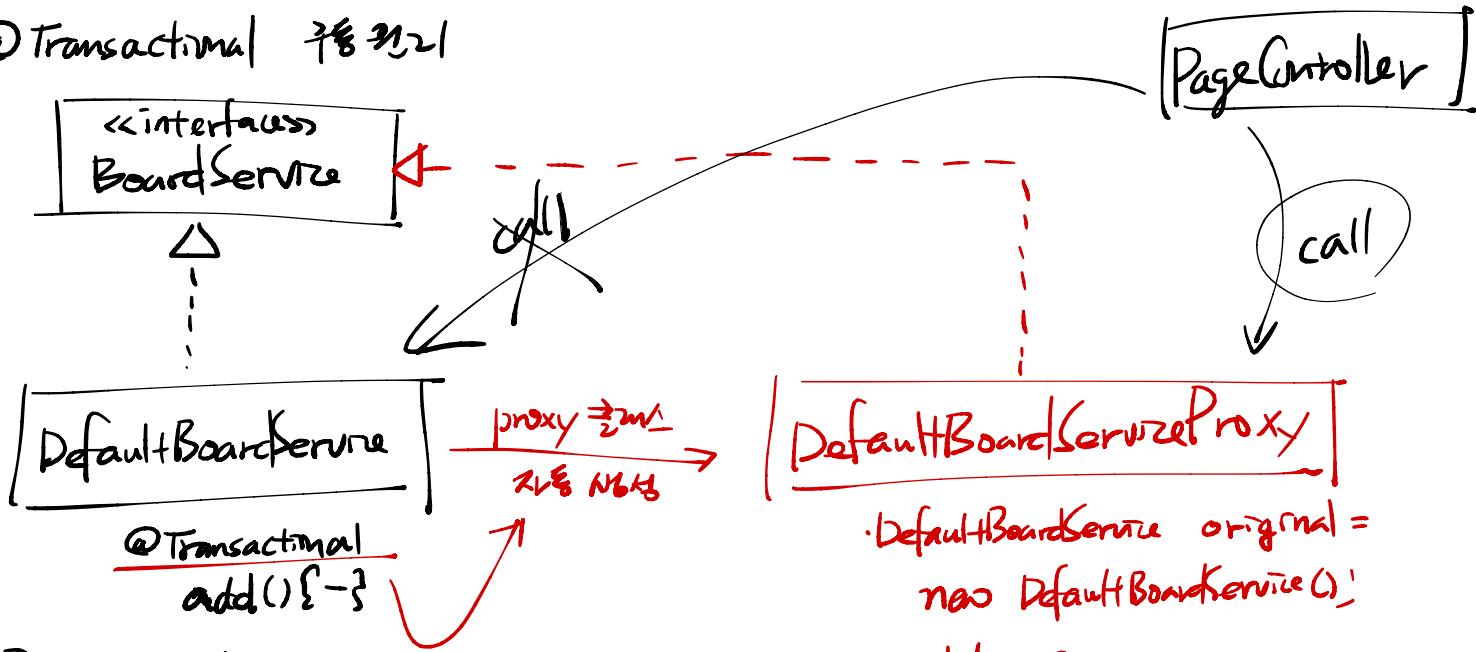
"bifrap-myapp.dao.BoardDao.insert",

-);

SQL
실행
하기위한
SQL문
생성
하기위한
DAO

MyBatis
로
구현

* @Transactional 78 31, 21



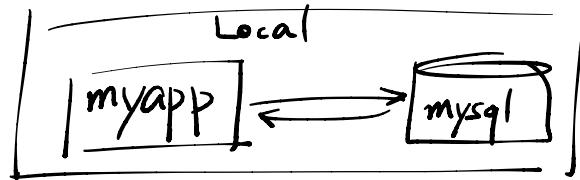
* AOP의 특징, 예제

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

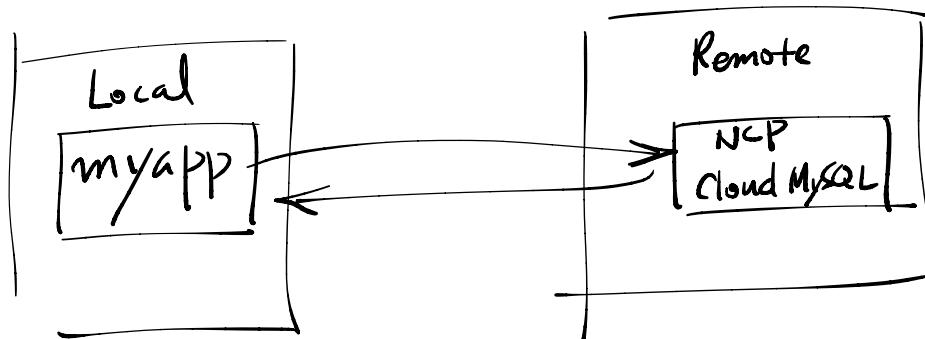
. **add () {**
 try {
 original.add();
 txManager.commit();
 } catch {
 txManager.rollback();
 }

62. Naver Cloud Platform 사용 예

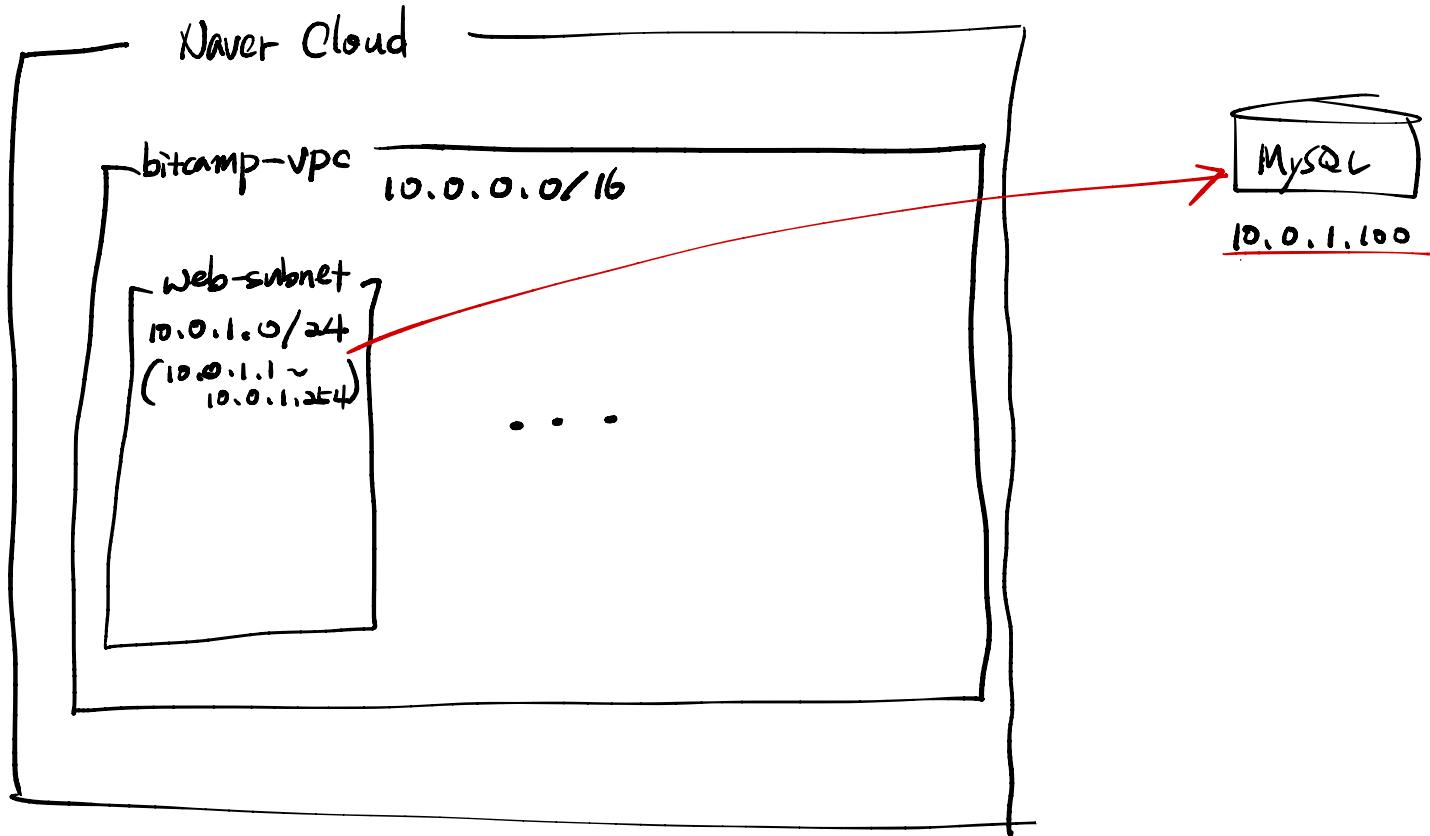
* ORI



* 대입



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



* RESTful API

File \equiv Data

URL

Representational Entity

global view HTTP method
+ GET, POST, PUT, DELETE, PATCH

representational state transfer
"REST"

File \equiv Data

DRAFTING HTTP REQUESTS URL

URL

Representational Entity

global view HTTP method
+ GET, POST, PUT, DELETE, PATCH

representational state transfer
"REST"

File \equiv Data

* function → REST API

다른 프로토콜

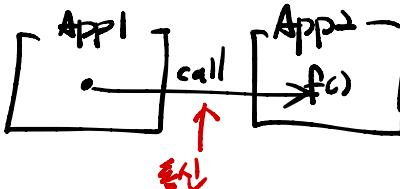
① function

(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

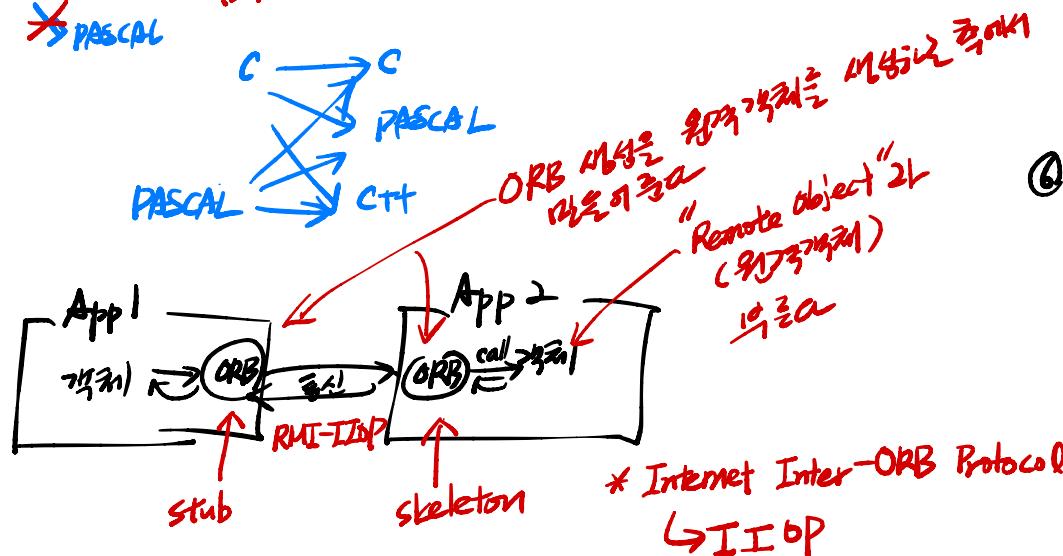
같은 인터프리터
만든 App끼리
함수를 호출하는
방식

C → C
✖ PASCAL

common
Object Request Broker
Architecture

자신의 언어로 만든 App끼리
함수를 호출하는 방식

C → C
✖ PASCAL
✖ C++



⑤ Web-service

✓ 웹으로 HTTP 프로토콜을 활용
✓ ORB를 통과하지 않고 서비스를 제공
→ 단점은?

단점은?
인터넷 규칙에 맞도록 개발하는
작업은 훨씬!

⑥ Enterprise JavaBeans (EJB)

Java에서 제공하는 RMI 기반

단점!

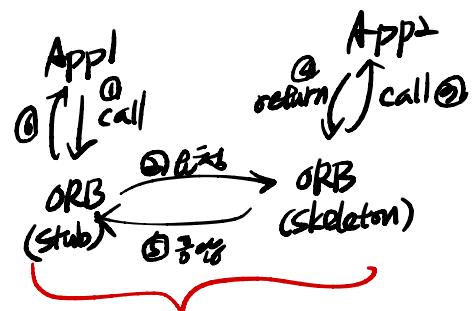
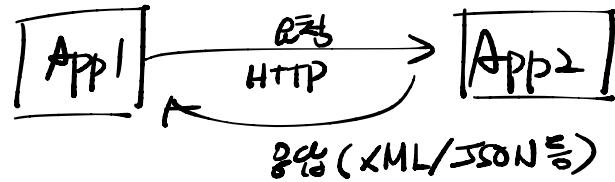
2000년 초반에는 PC를 두
대의 디바이스를 써서 EJB를 기반
으로 한 단점이 크게 문제가 되었지만,

* function → REST API

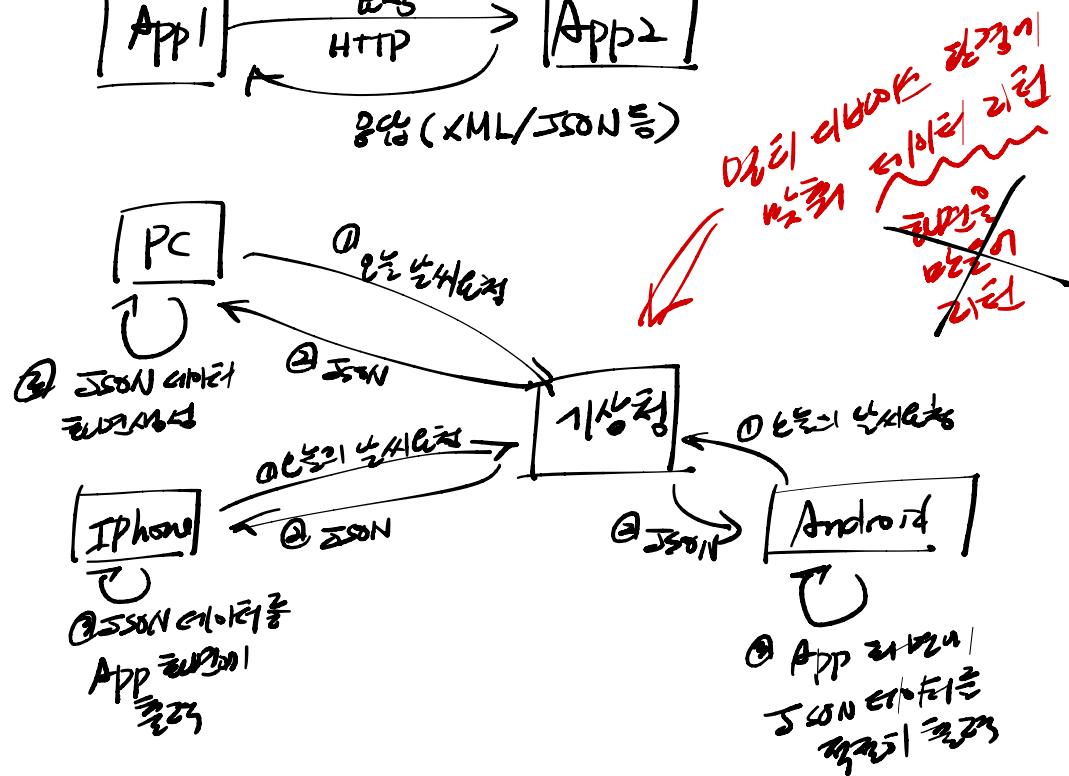
(Web-Service)
EJB

→ REST API

인터넷으로
ORB를 통한
서비스 제공

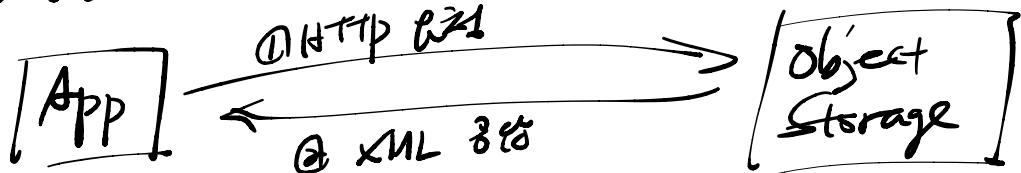


ORB를 통한
서비스 제공의
방법은?



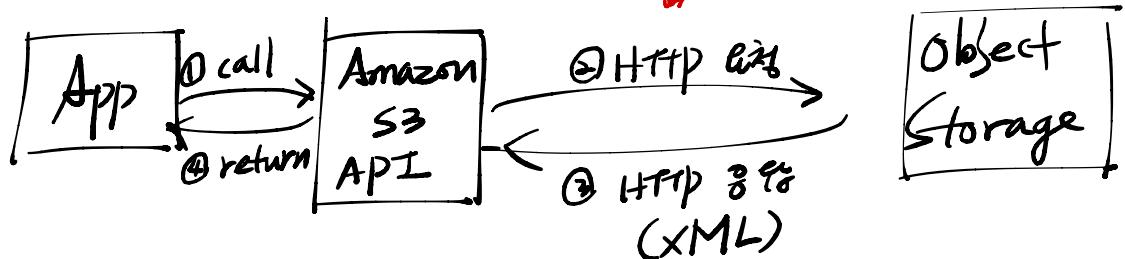
* Object Storage REST API API
↳ HTTP 퀼/값, 키!

① 직접 접근

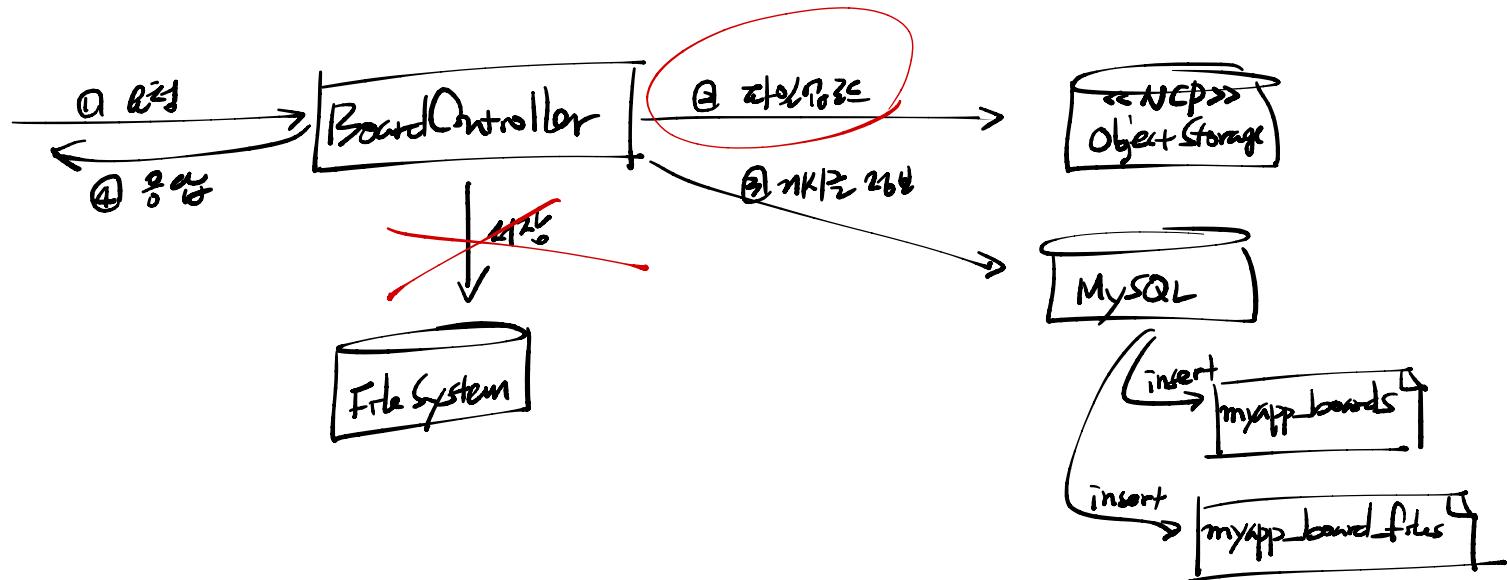


HTTP 퀼/값은 직접 접근하기 편리하다
HTTP 응답을 직접 접근하기 편리하다

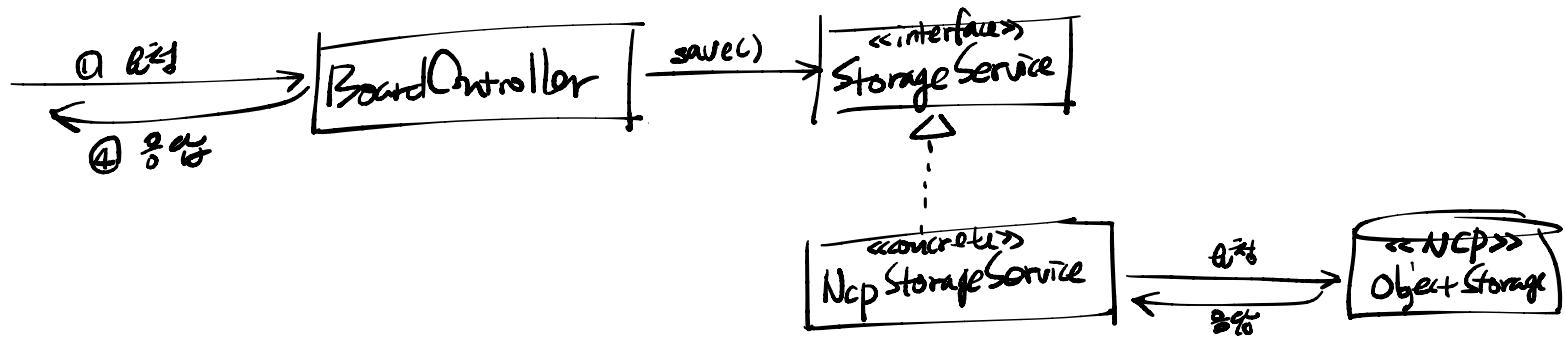
② 경유 애플리케이션



* 내부로 첨부파일은 NCP의 Object Storage에 저장

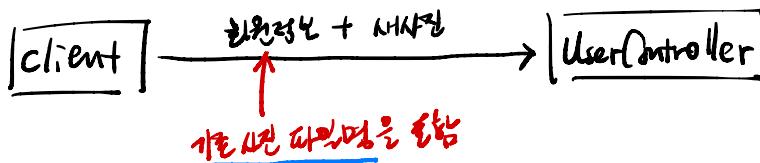


* 게시글 첨부파일은 NCP의 Object Storage에 저장 → NCP 연동로직을 서비스 객체로 분리



* 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.



이제는, 변경, 삭제 시

기존 정보를 사용할 때는

항상 새비밀번호를 사용해야!

기존 계정정보를 클라이언트가 조작해
할 수 있다! → 즉, 이제는, 변경, 삭제할 때
새비밀번호를 사용해야!

* 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

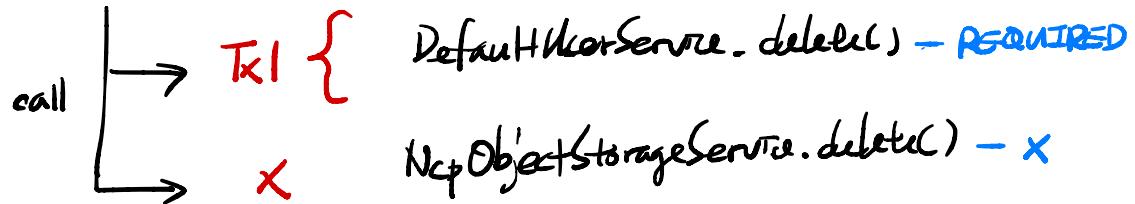
UserController.delete() X
↓ call
Tx1 { DefaultUserService.delete()

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

* 키워드 어노테이션 정리

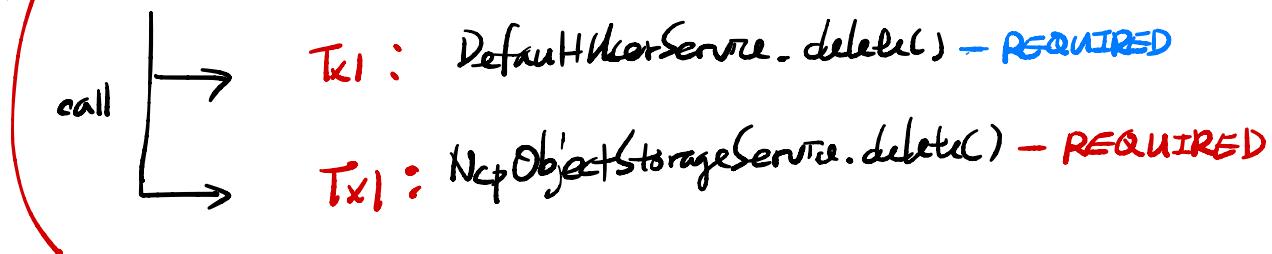
① 허용

X UserController.delete() X

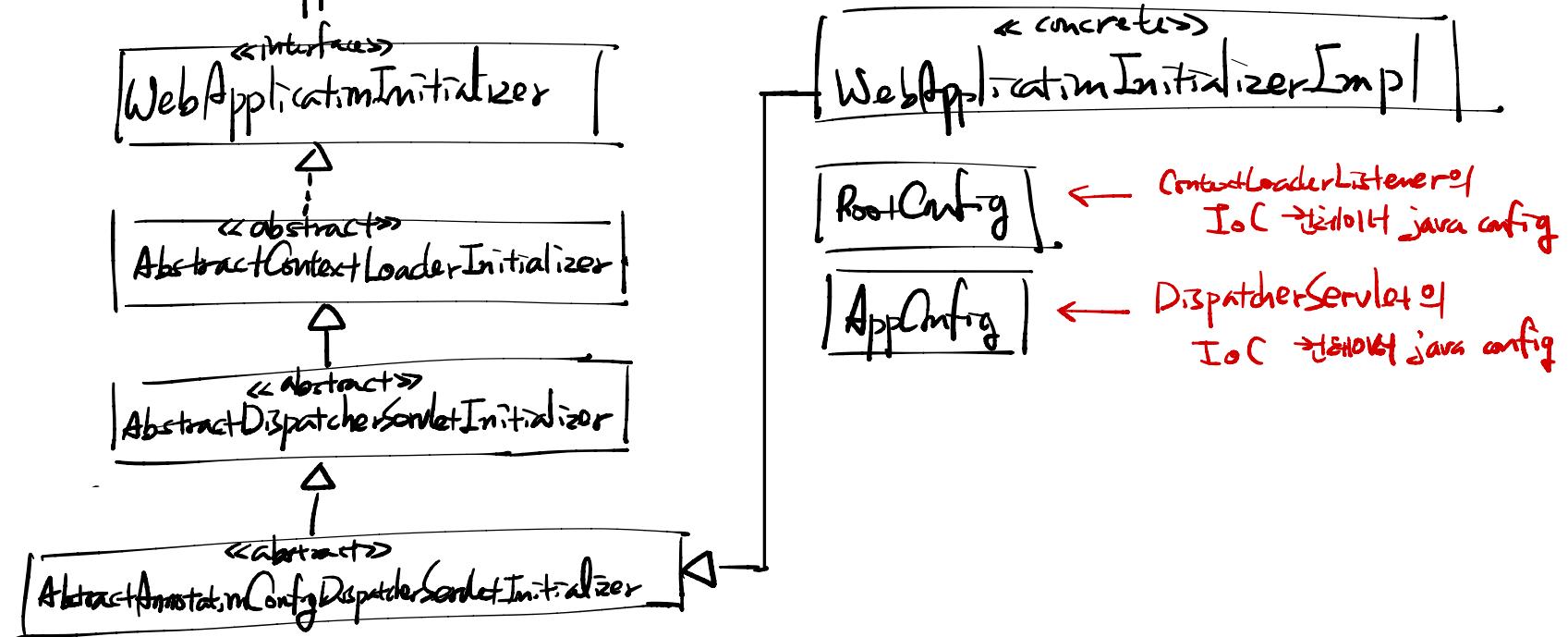


② 허용

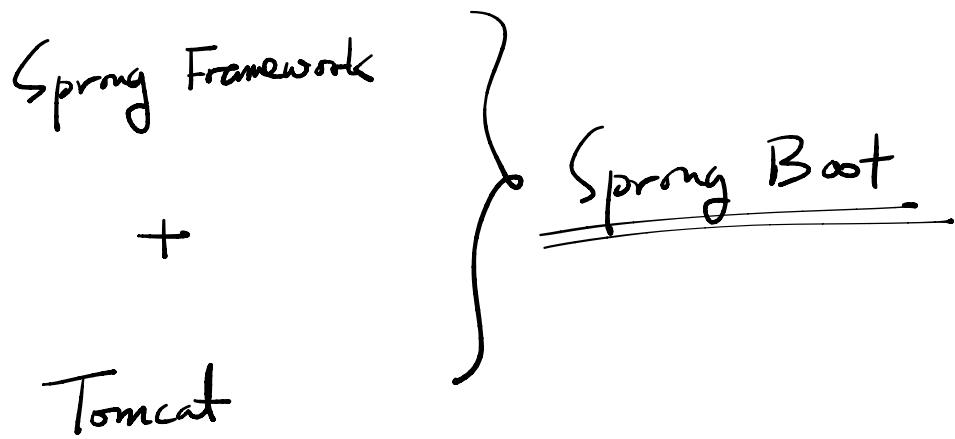
Tx1 UserController.delete() - REQUIRED



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 | _____ | |

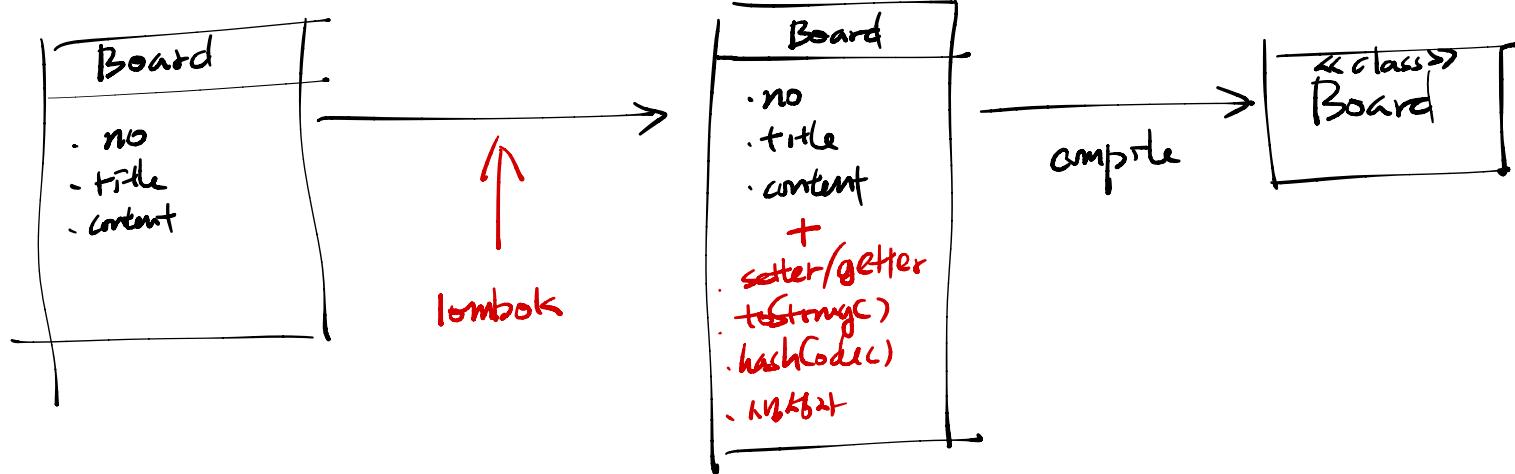
67. Lombok 2010-2021年版

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

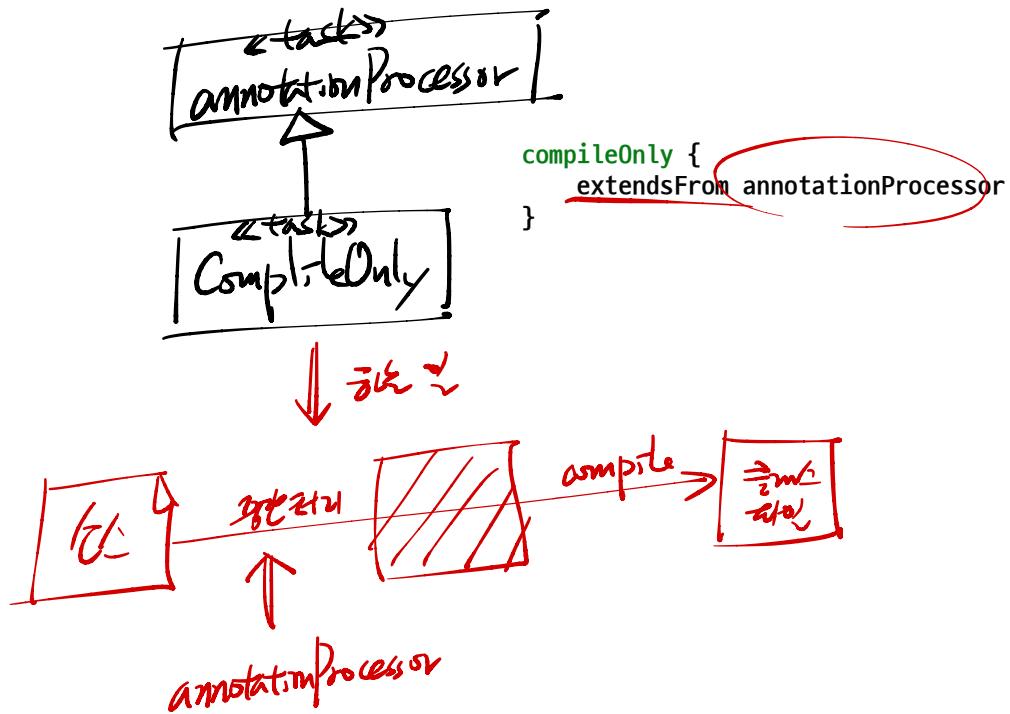
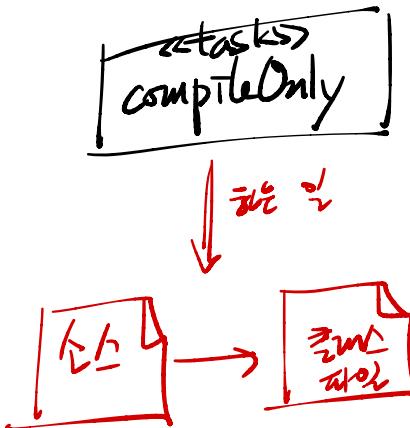
Gradle



Build



* Gradle et Task



* lombok IntelliJ 풀이방법의 차이

