

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

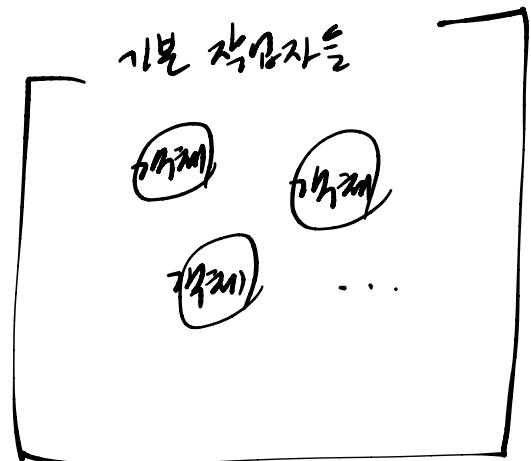
- ③ " 웹으로 뷰를

- 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

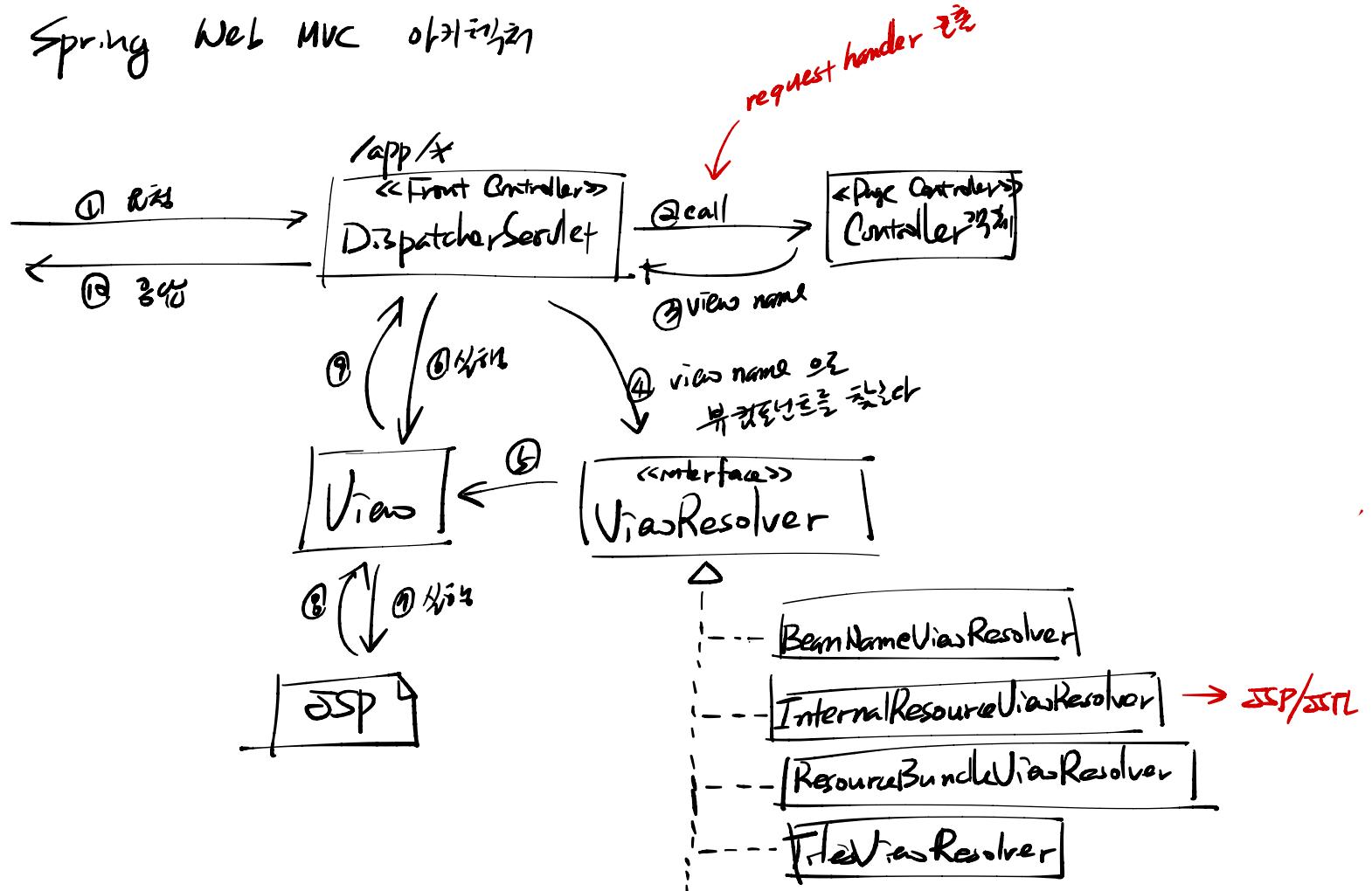
String
}

==

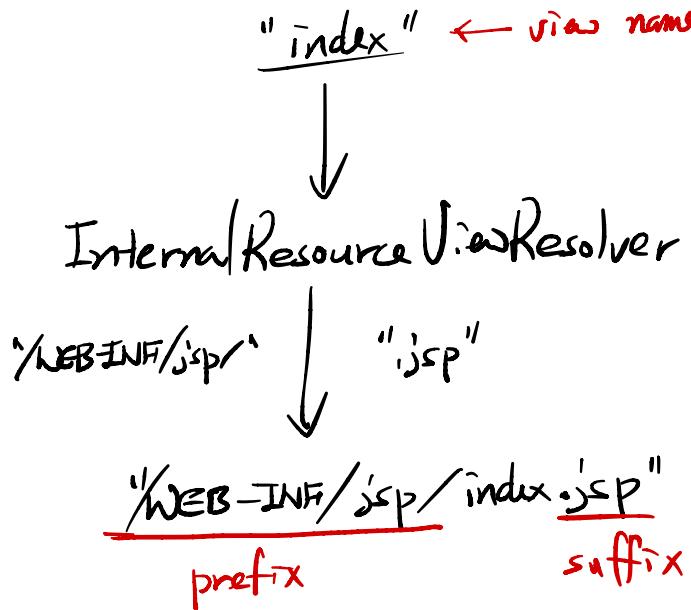
login(String email, String password) {

요청 헤더의 자세히 봐

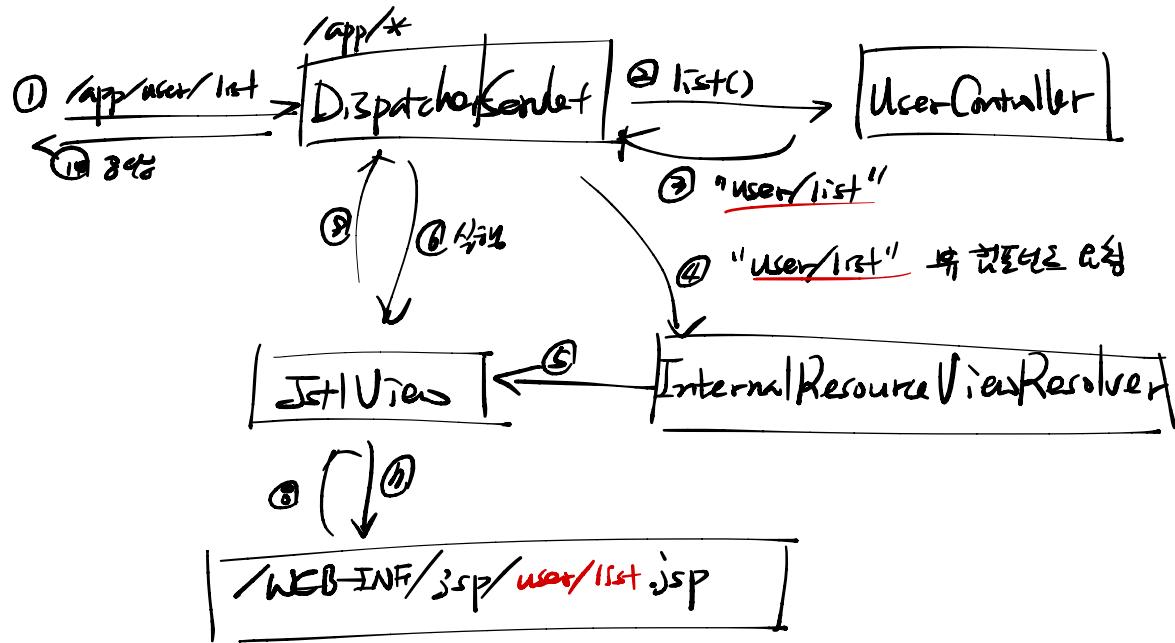
* Spring Web MVC 07/27/21



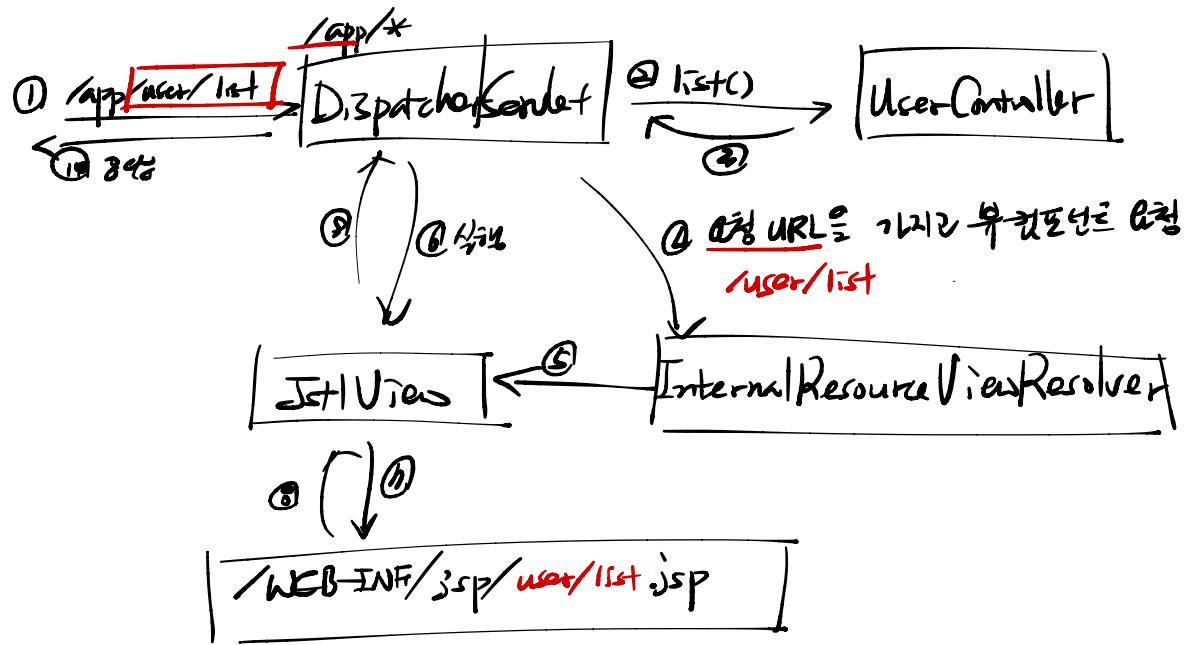
- * Internal/Resource ViewResolver



* view name %*% can

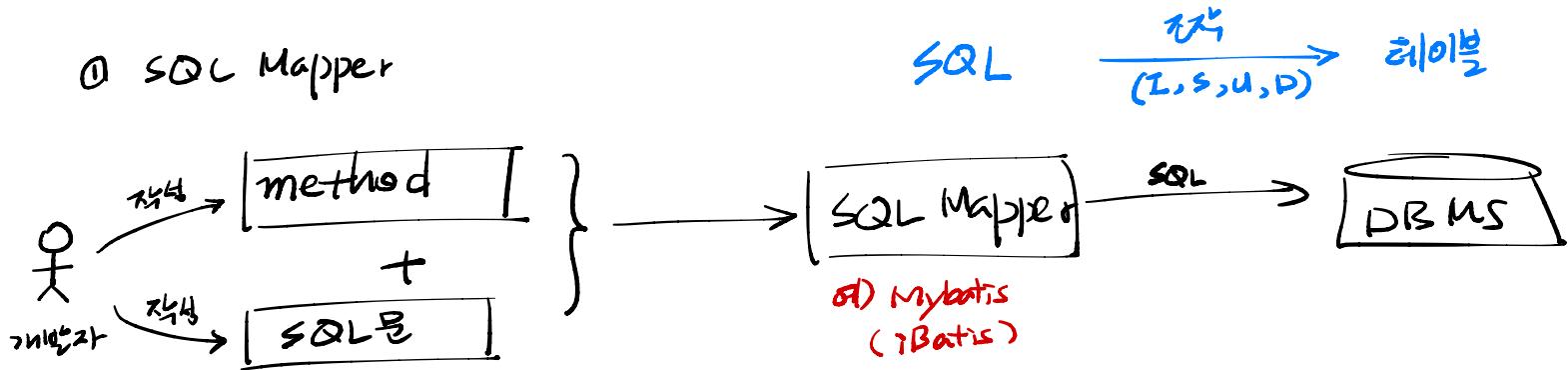


* view name $\stackrel{\text{보통}}{\rightarrow}$ controller

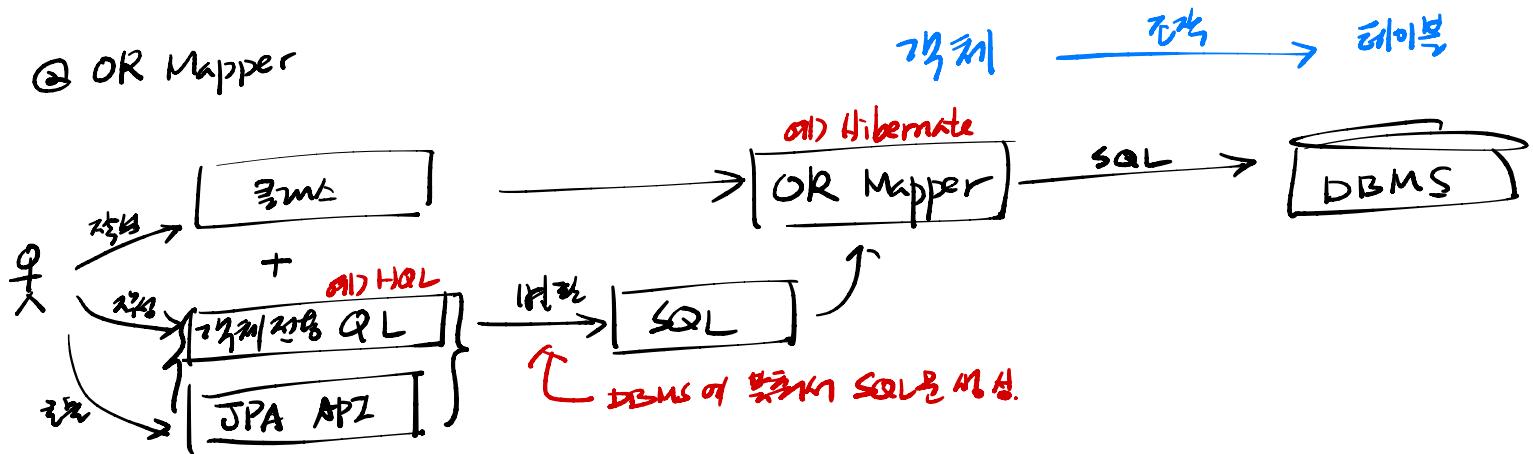


* SQL Mapper vs OR Mapper

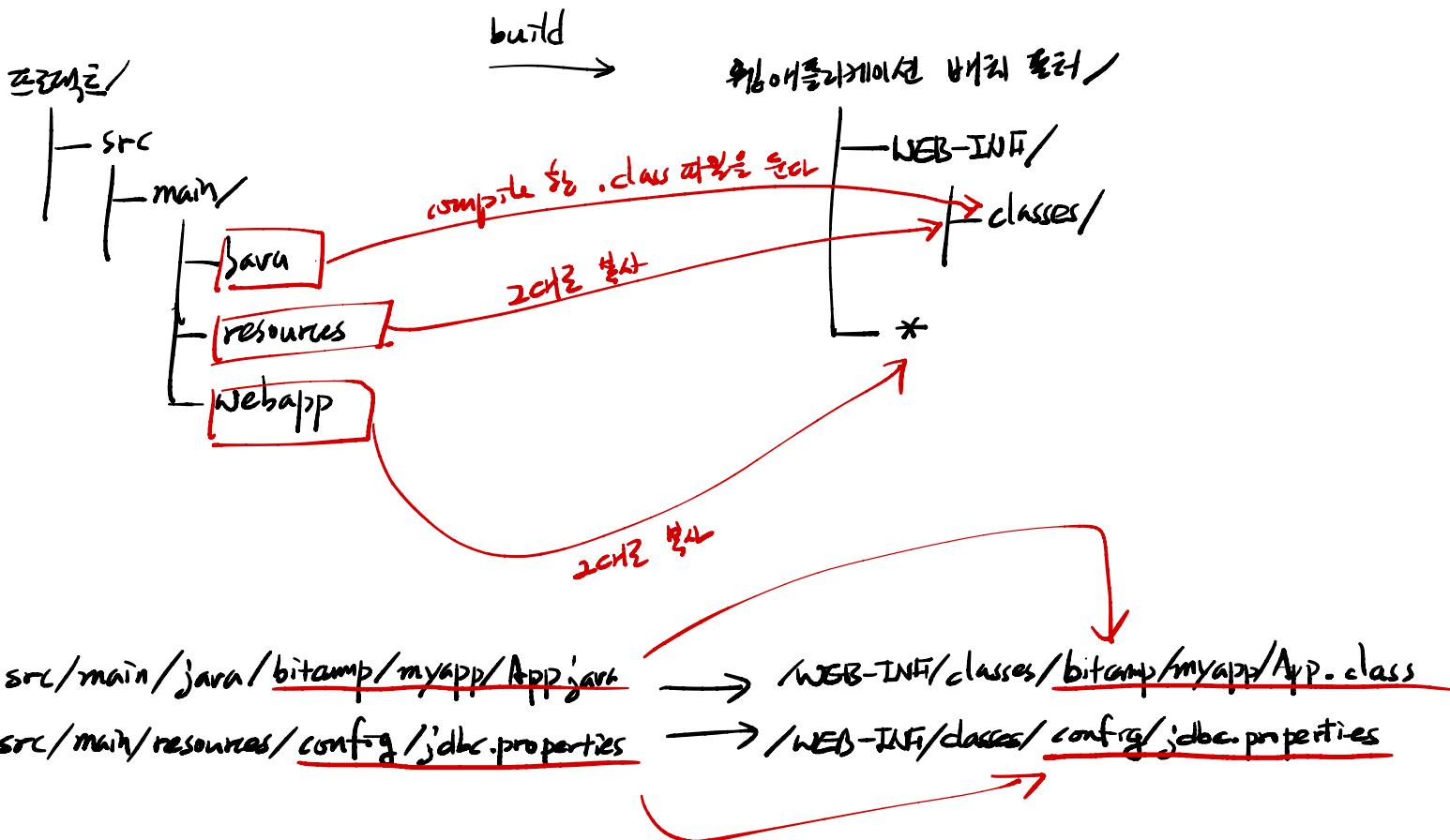
① SQL Mapper



② OR Mapper



* build et classpath

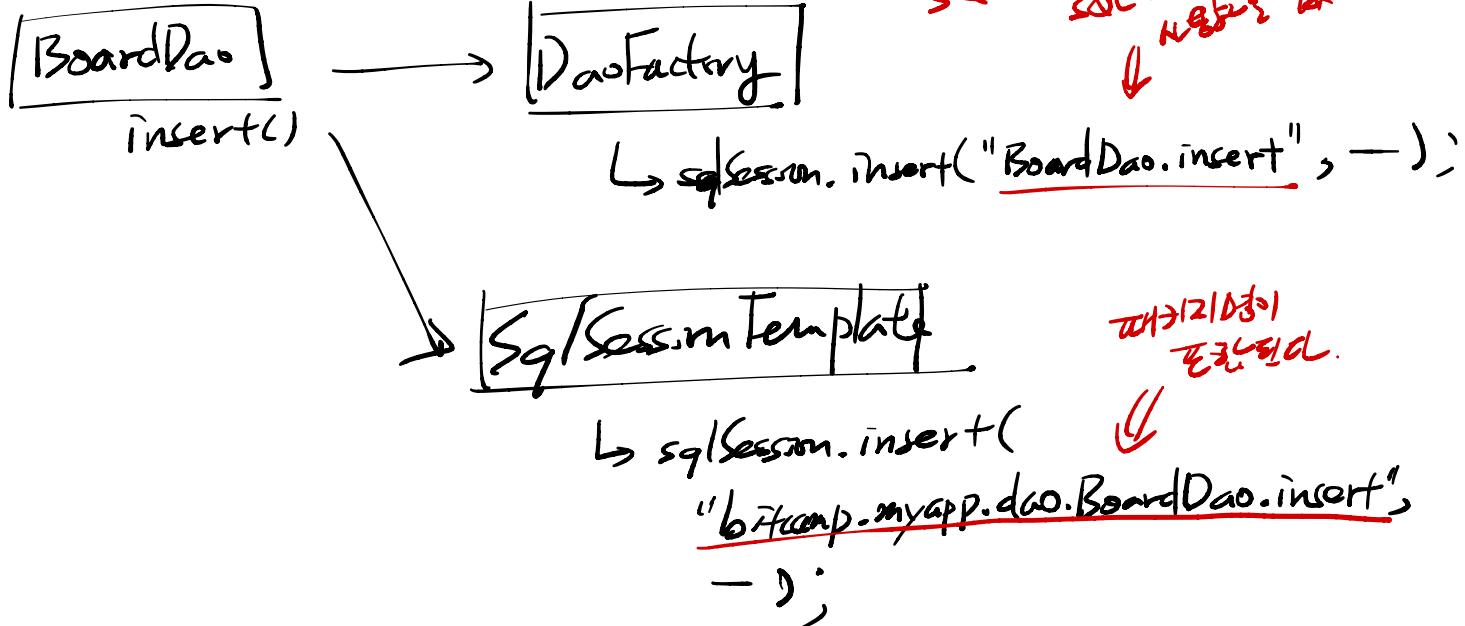


* DaoFactory 4

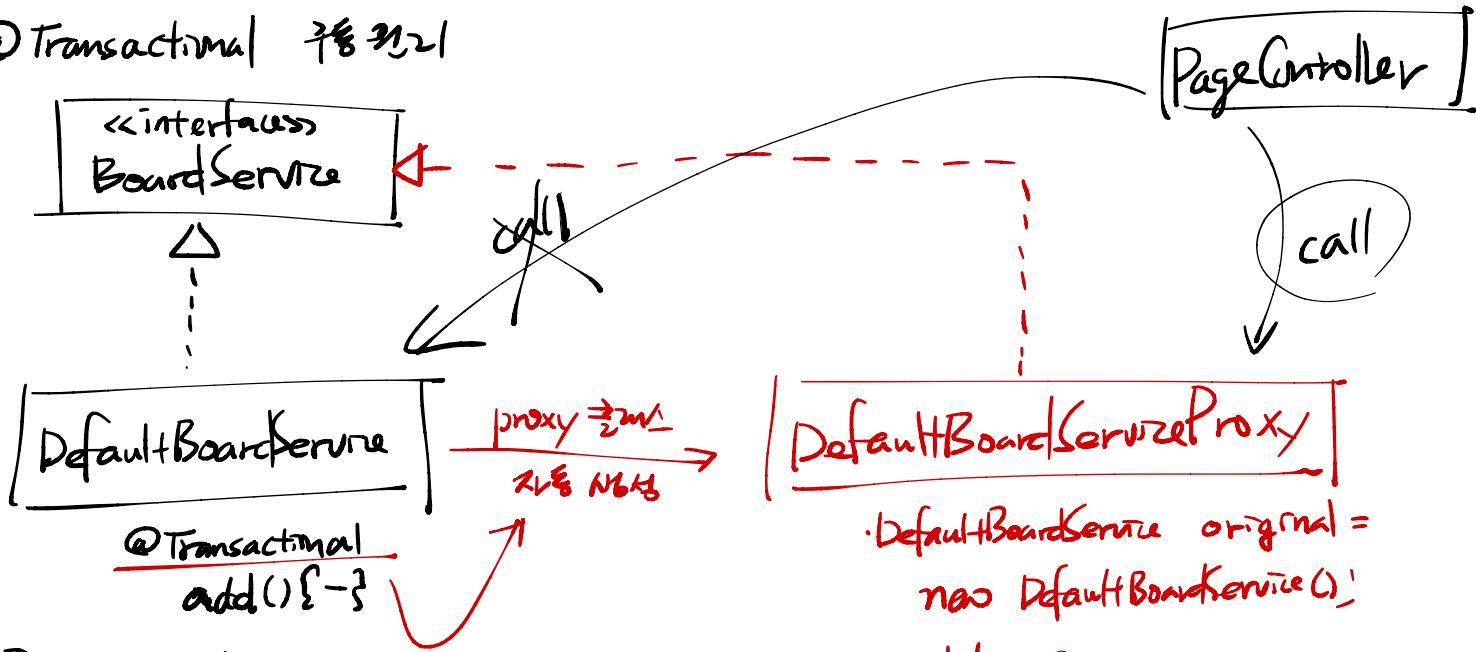
↑ graduate
DAO 04/24

SqISessionTemplate

Myobatrachus嘉陵江
Dorsal 鳃孔



* @Transactional 78 31, 21



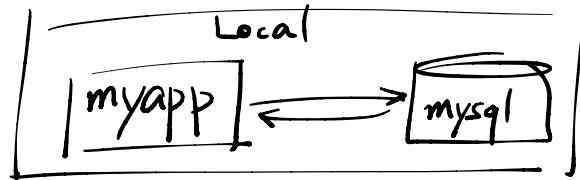
* AOP의 특징, 예제

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

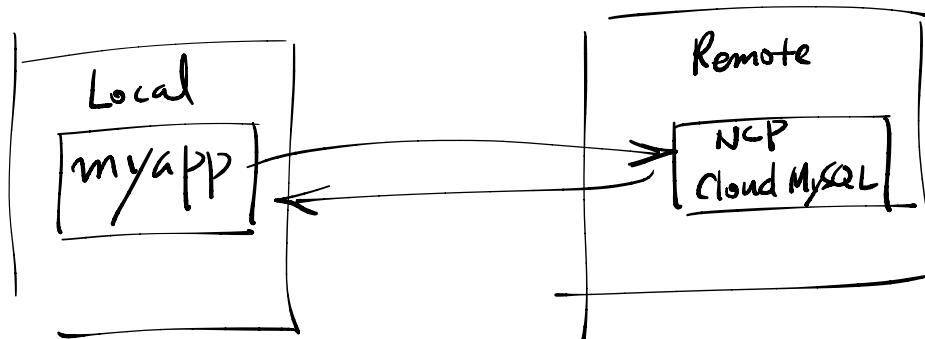
. **add () {**
 try {
 original.add();
 txManager.commit();
 } catch {
 txManager.rollback();
 }

62. Naver Cloud Platform 사용 예

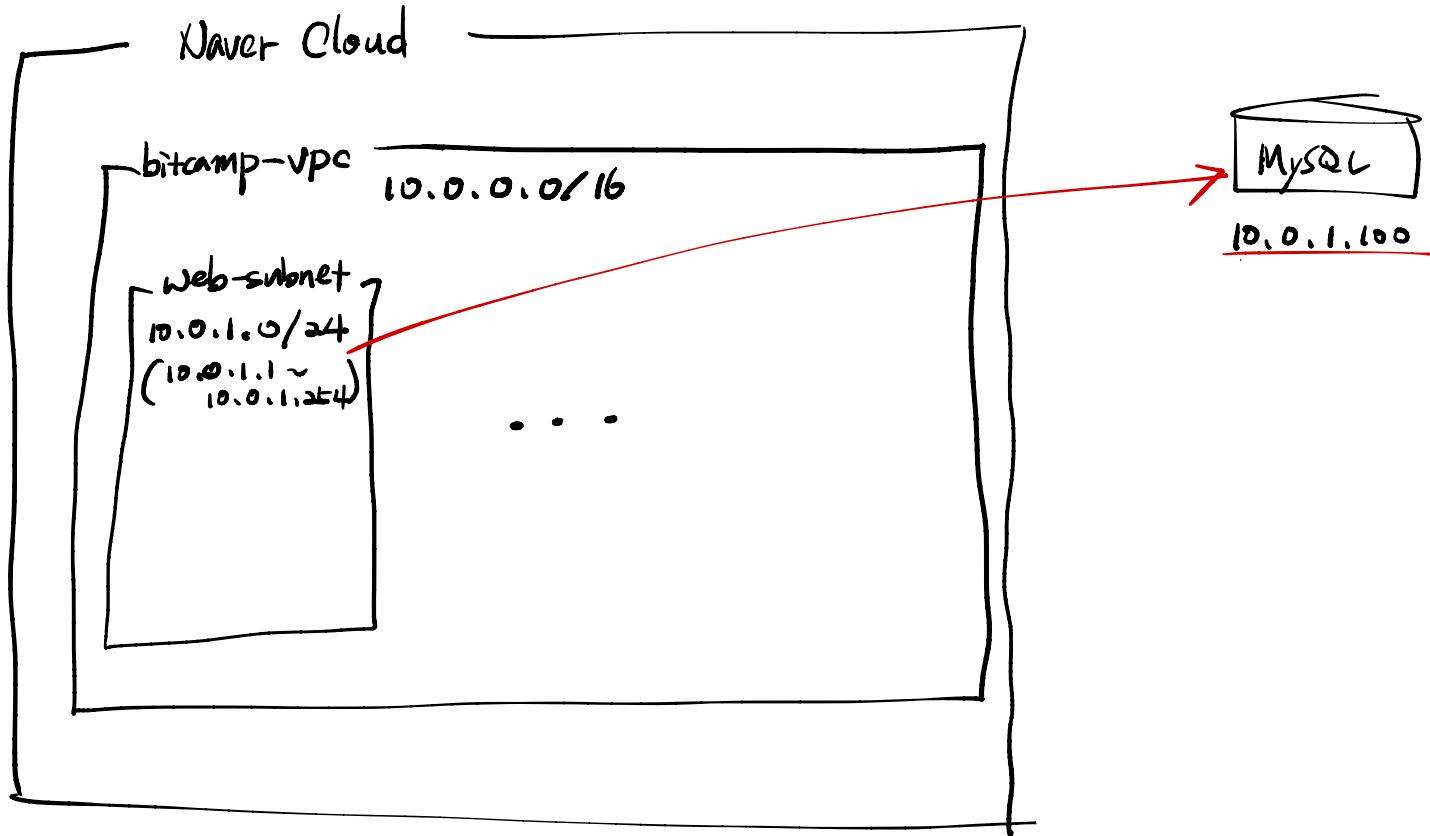
* ORI



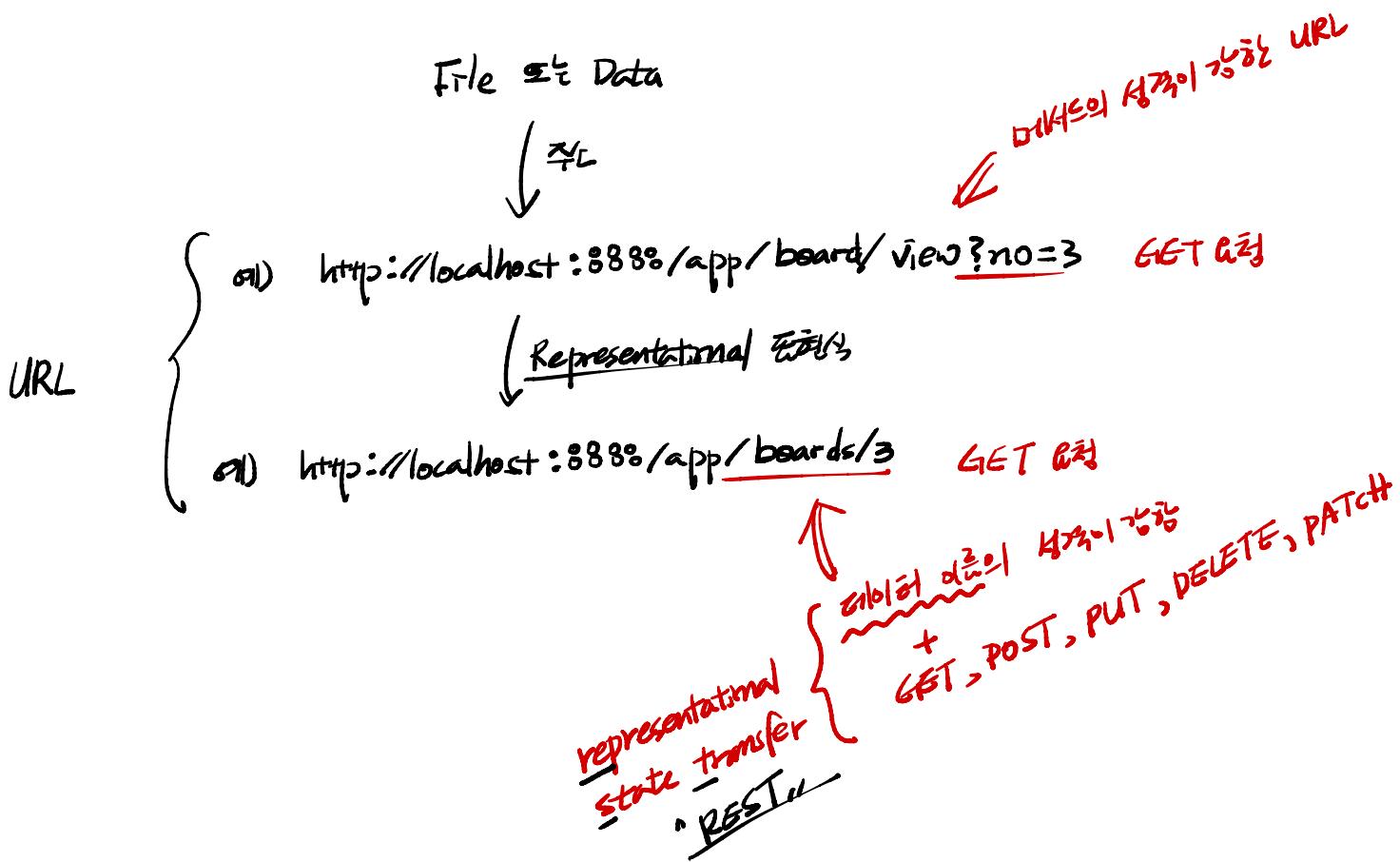
* 대입



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



* RESTful API



* function → REST API

다른 프로토콜

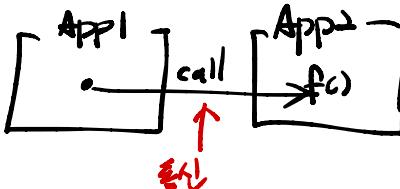
① function

(RPC)

② Remote Procedure Call

(RMI)

③ Remote Method Invocation



C 등 프로그래밍 언어

기법의 암호화



App의 행동을 분석 (분석 컴퓨터)



한 App이 하는 일을

여러 App으로 번역해 보내는

App 한 번 행동이 번역된다



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

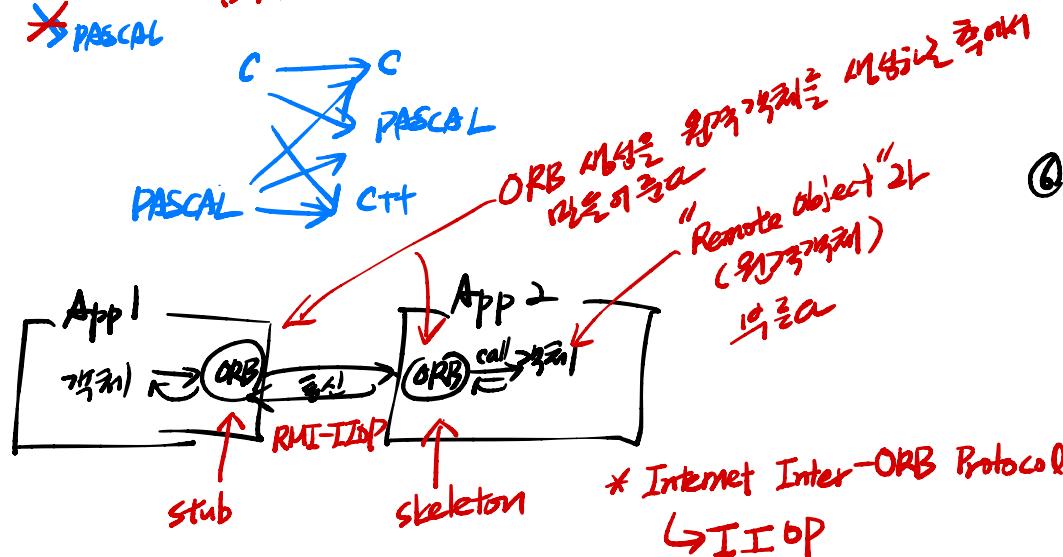
같은 인터프리터
만든 App끼리
함수를 호출하는
방식

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!

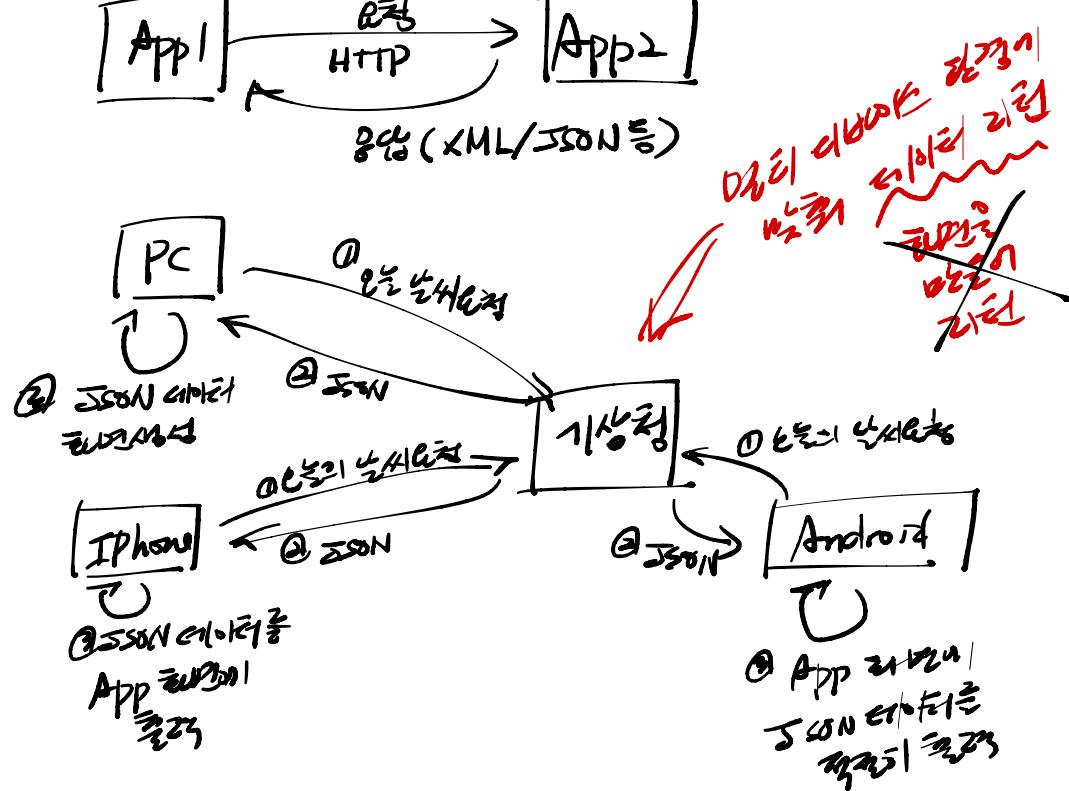
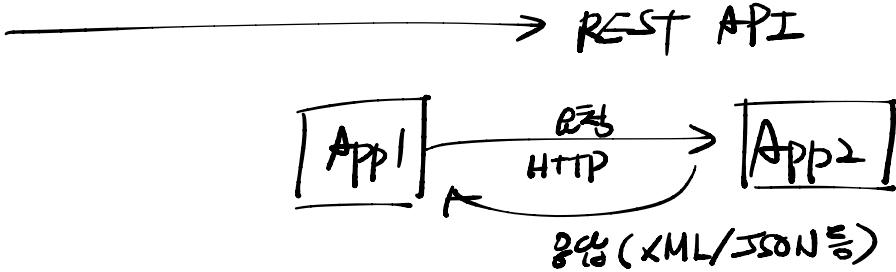
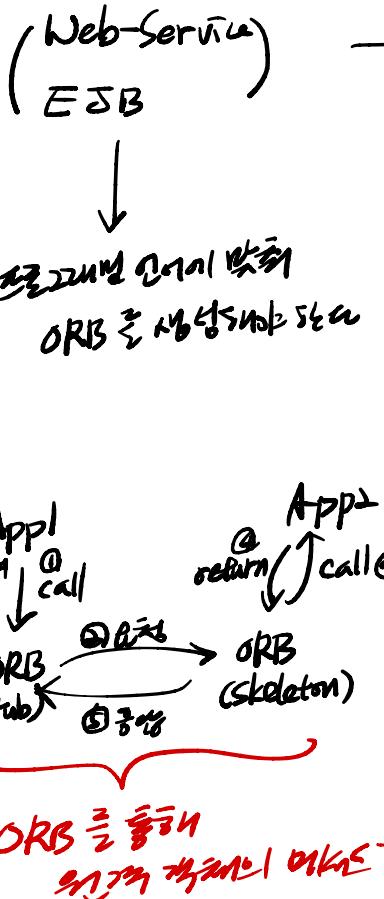
⑥ Enterprise JavaBeans (EJB)

Java에서만 가능 EJB 가능

단점!

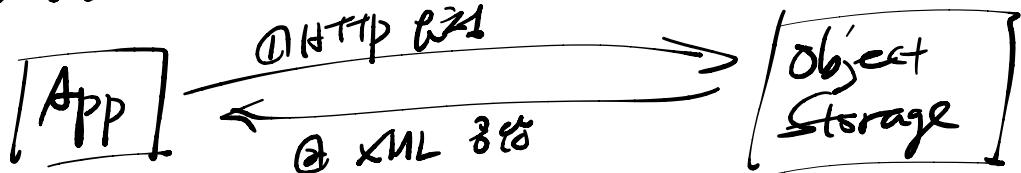
2000년 초반에는 PC를 두
대의 디바이스를 써서 EJB를 개발
하면서 성능이 크게 떨어지거나 오류가 많았던

* function → REST API



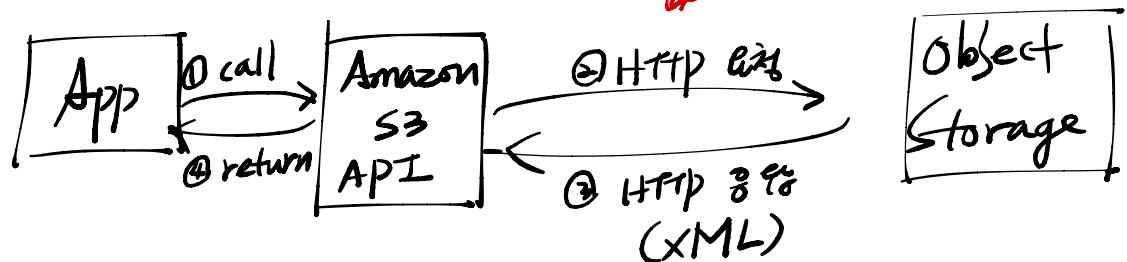
* Object Storage REST API API
↳ HTTP 퀼/값, 키!

① 직접 접근

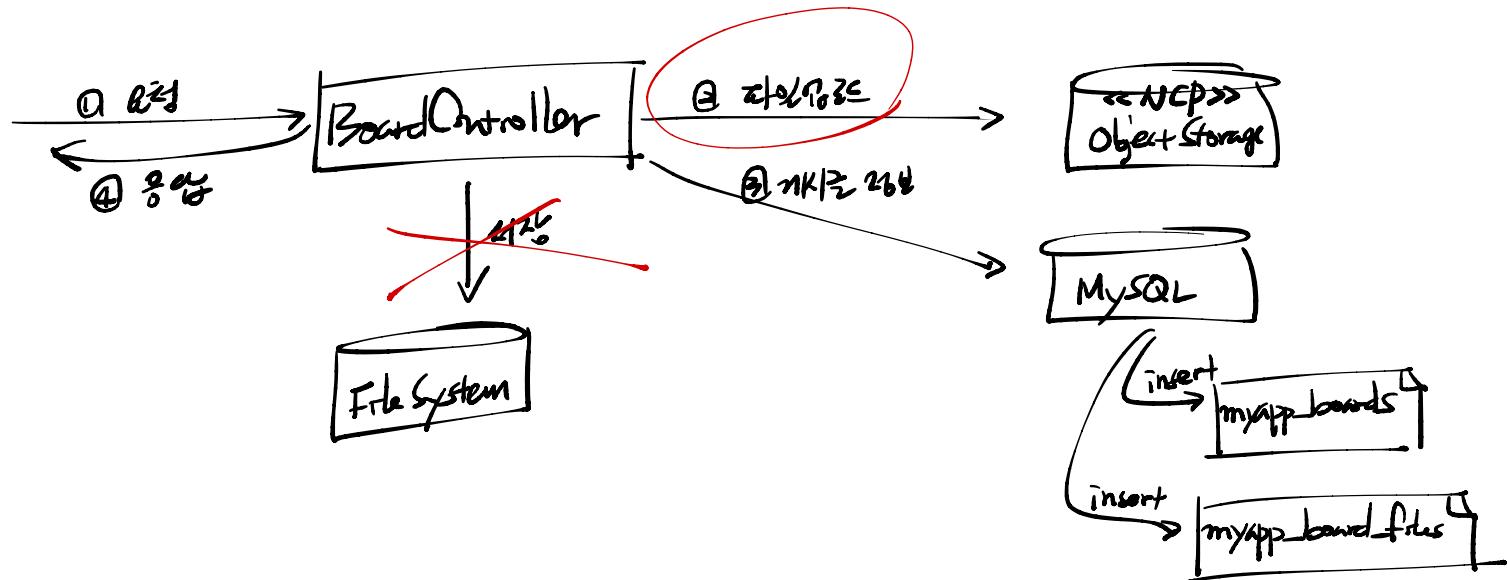


HTTP 퀼/값은 직접 접근하기 편리하다
HTTP 응답을 직접 접근하기 편리하다

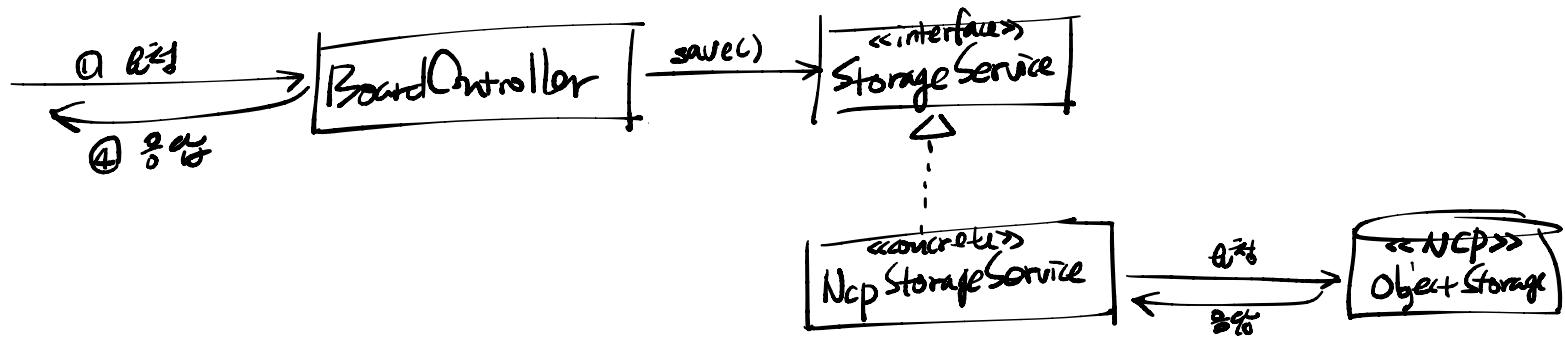
② 경유 애플리케이션



* 내부로 첨부파일은 NCP의 Object Storage에 저장

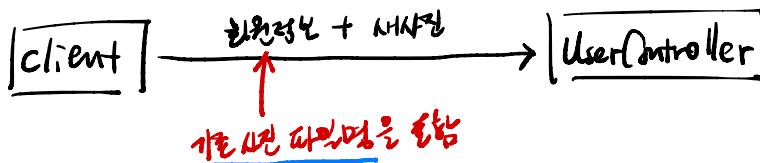


* 게시글 첨부파일은 NCP의 Object Storage에 저장 → NCP 연동로직을 서비스 객체로 분리



* 사용고객과 보안

① 보안에 악용은 예 : 회원정보 변경



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



이제, 변경, 삭제 시

기존 정보를 사용할 때는

항상 새롭게 프로그램 사용하자!
기존 기반정보를 클라이언트가 알지 못하! → 꼭 새롭게 프로그램 사용하자!

- i) 기존 사용자인증 → 클라이언트가 보낸 세션키로
세션을 흡수하여 쓰자!
- ii) NH 네이버 인증 → 네이버 인증
- 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

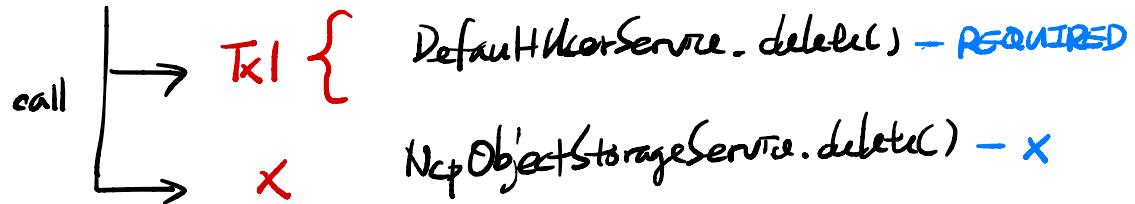
UserController.delete() X
↓ call
Tx1 { DefaultUserService.delete()

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

* 키워드 어노테이션 정리

① 허용

X UserController.delete() X



② 허용

Tx1 UserController.delete() - REQUIRED

