

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

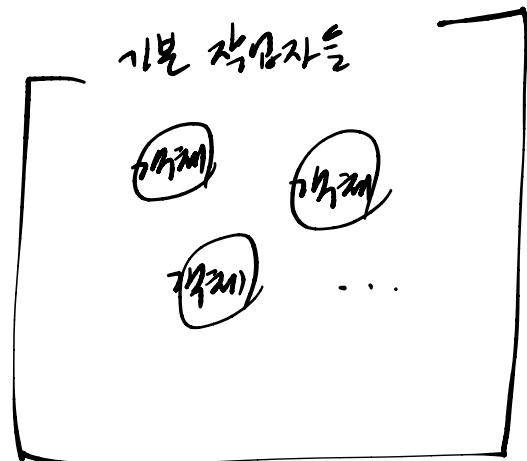
- ③ " 웹으로 뷰를

- 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

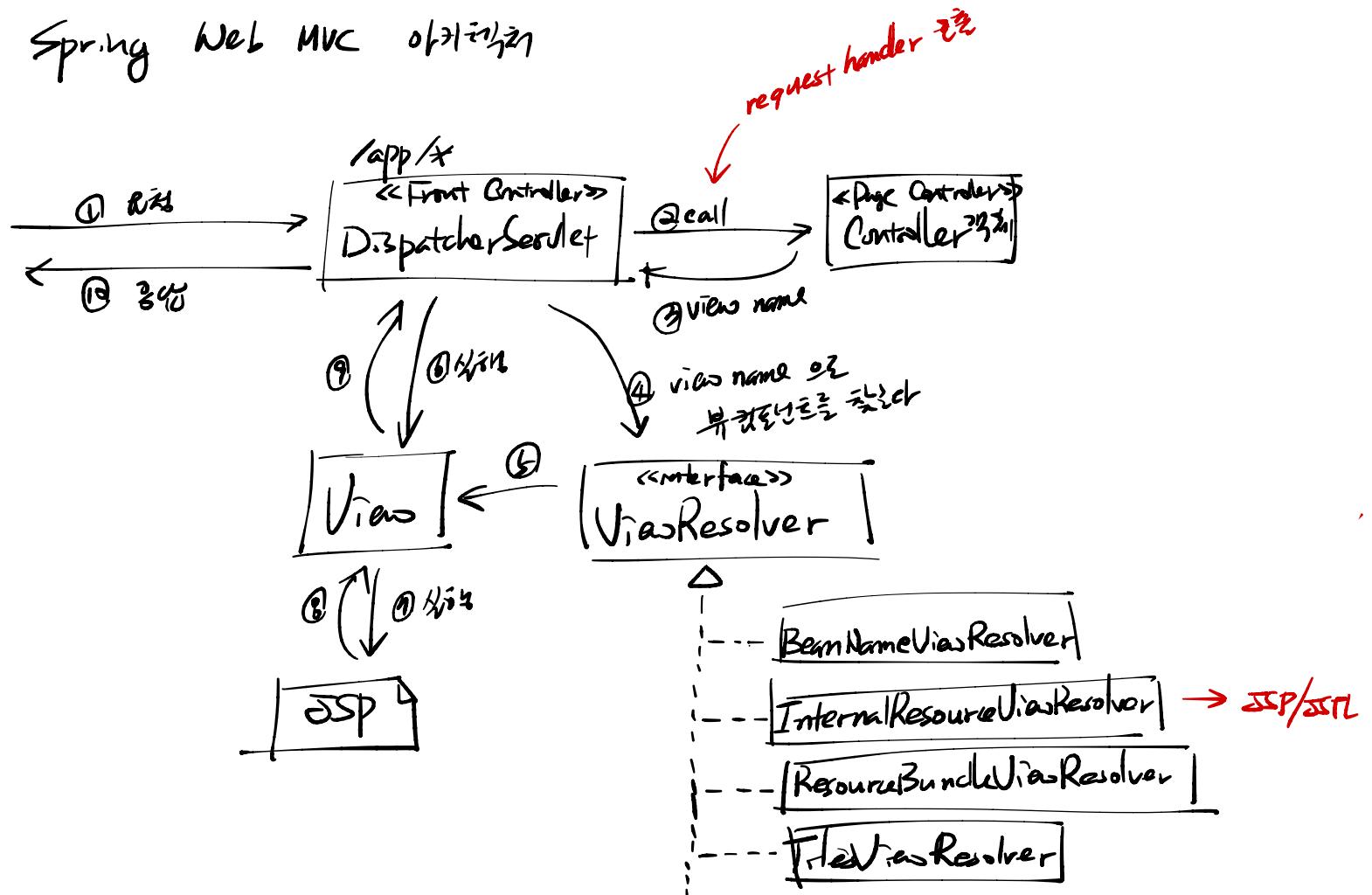
String

login(String email, String password) {

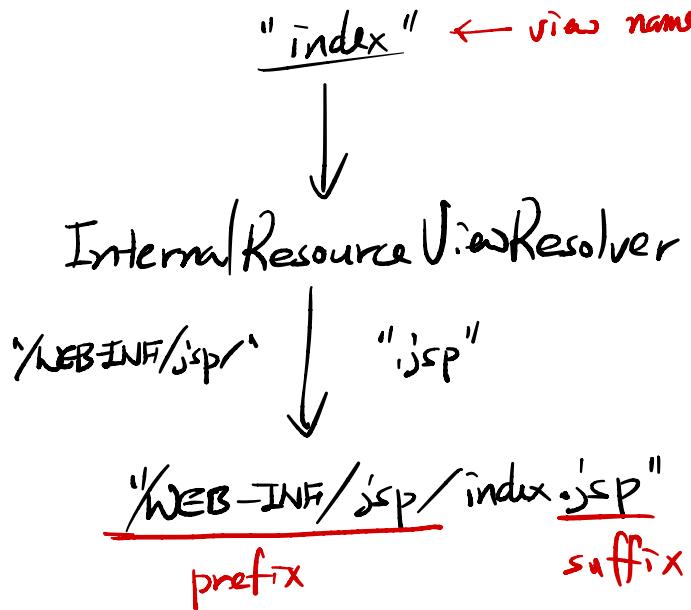
요청 헤더의 자세히 봐

}

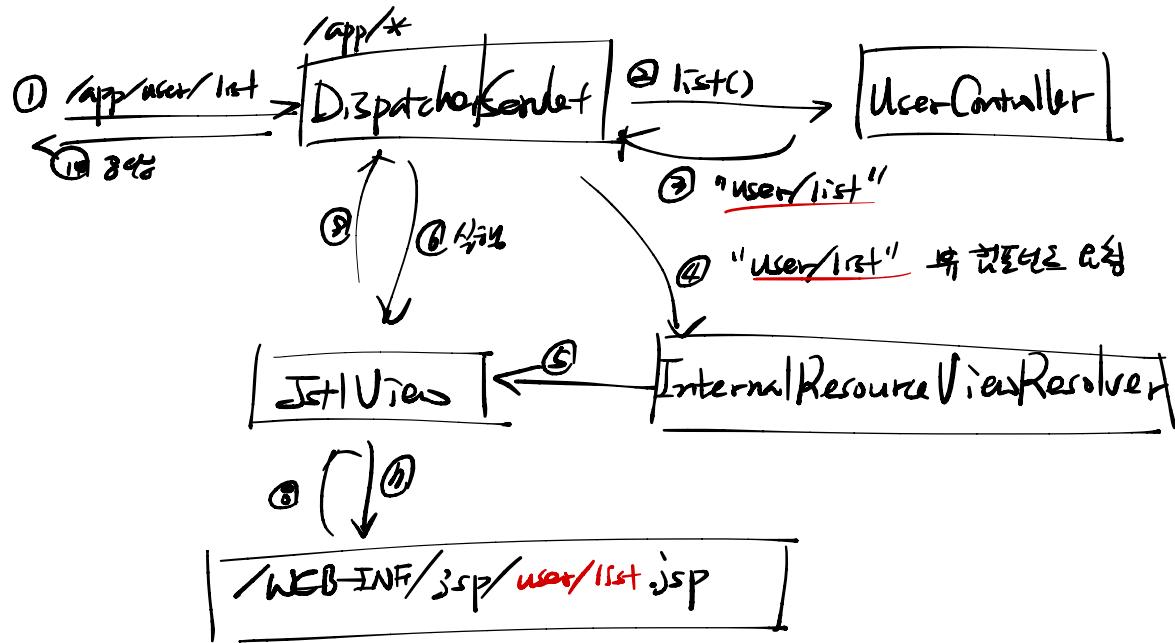
* Spring Web MVC 07/27/21



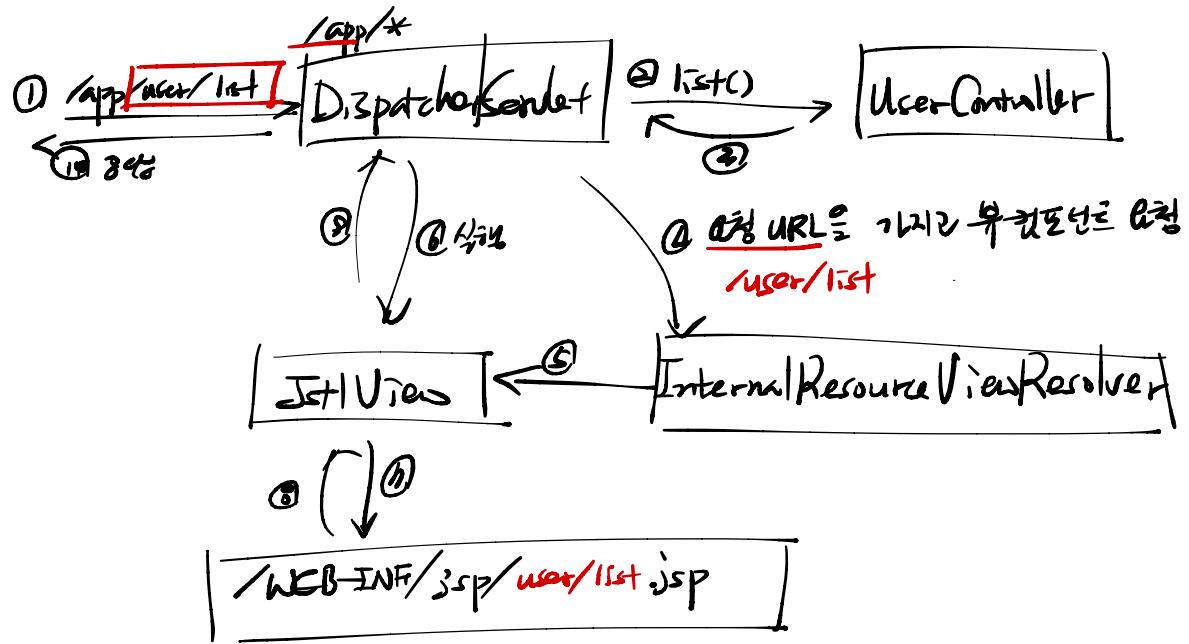
- * Internal/Resource ViewResolver



* view name %*% can

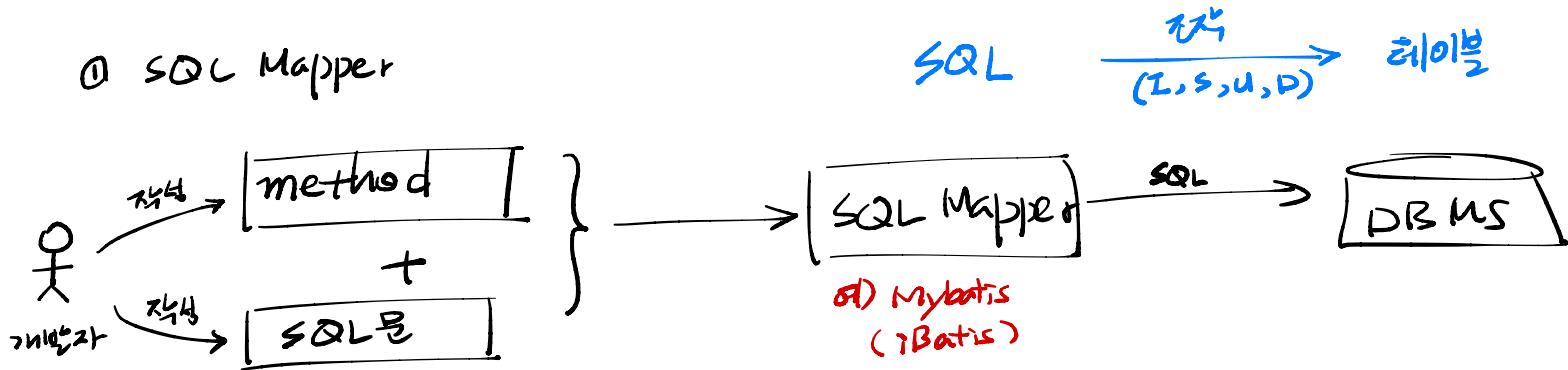


* view name $\stackrel{\text{보통}}{\rightarrow}$ controller

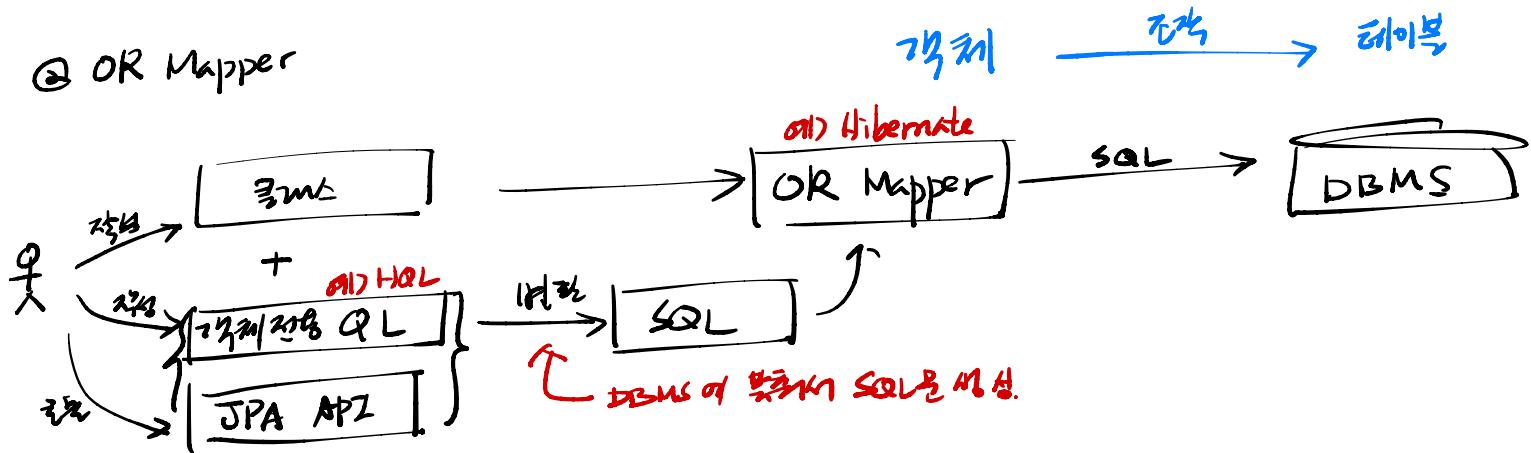


* SQL Mapper vs OR Mapper

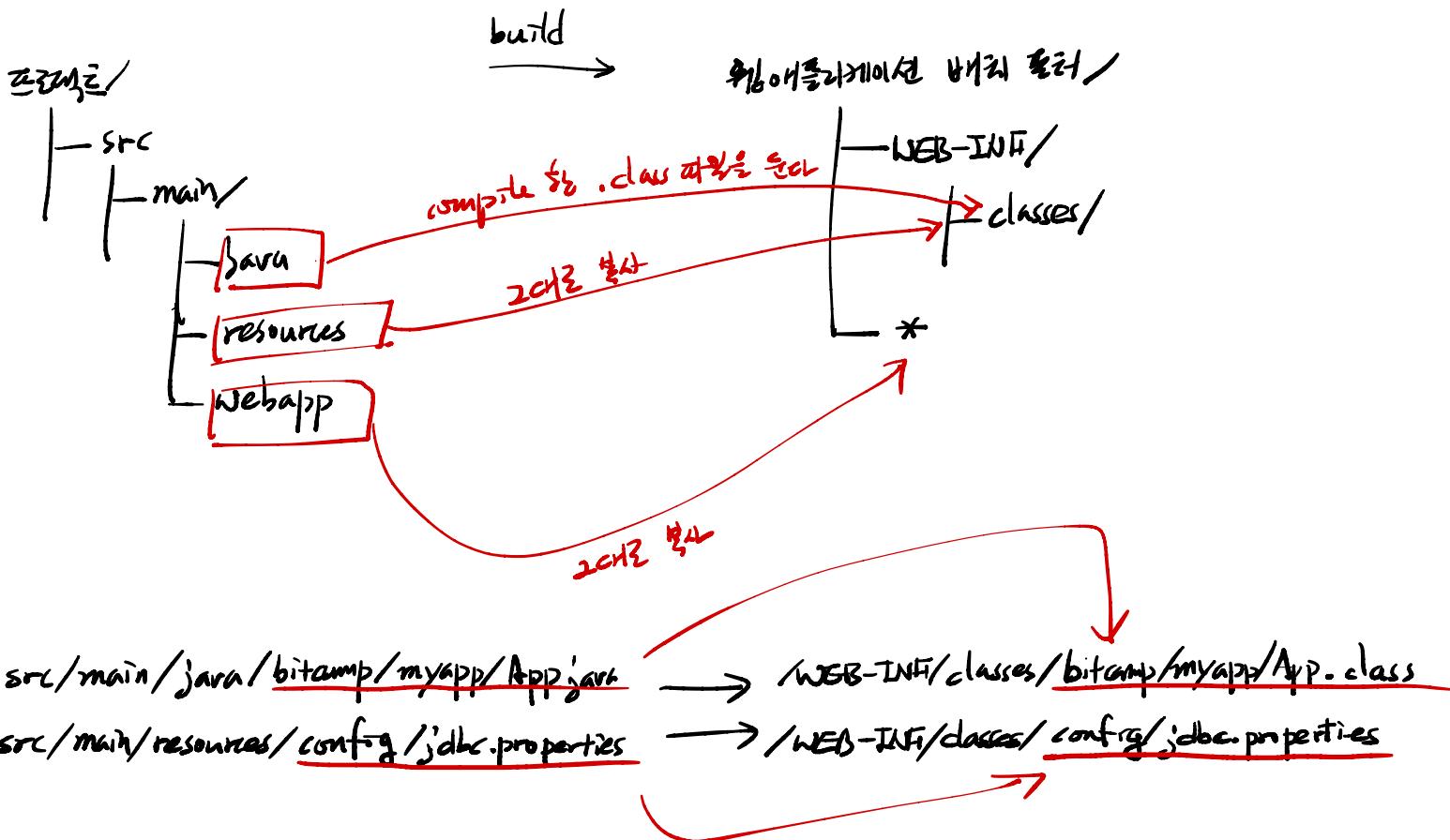
① SQL Mapper



② OR Mapper



* build et classpath

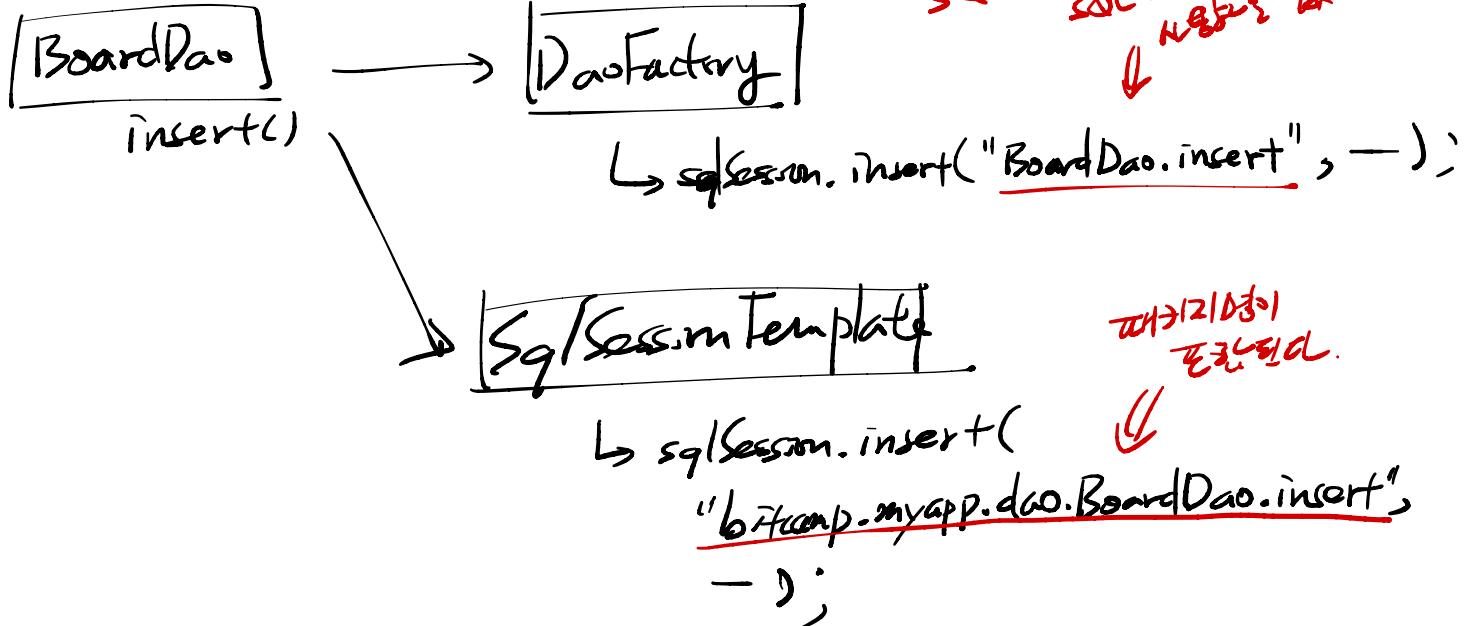


* DaoFactory

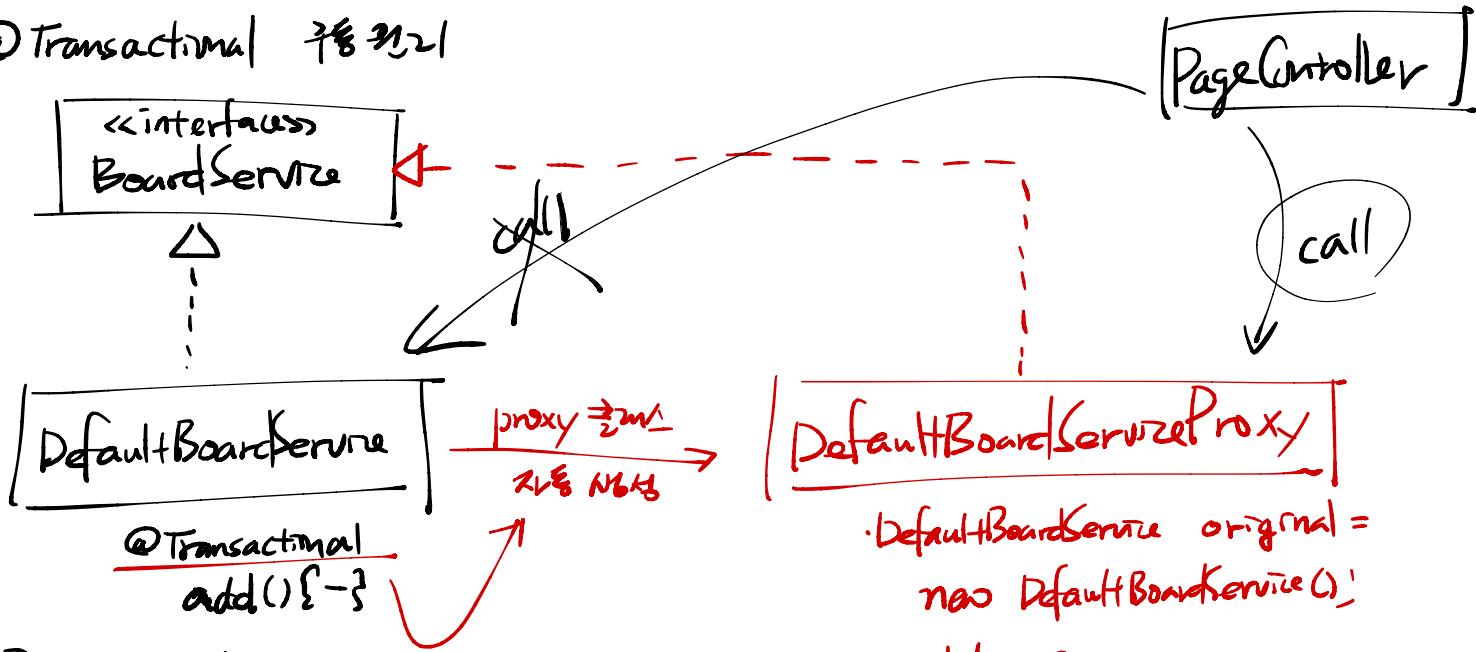
↑ graduate
DAO 대기구

SqSessionTemplate

Myobatrachus嘉陵江
Dorsal 鳃孔



* @Transactional 78 31, 21



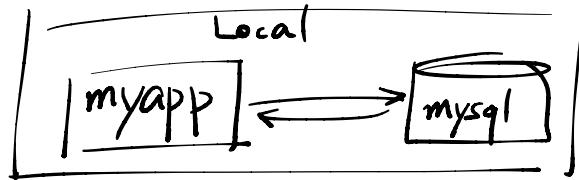
* AOP의 특징, 예제

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

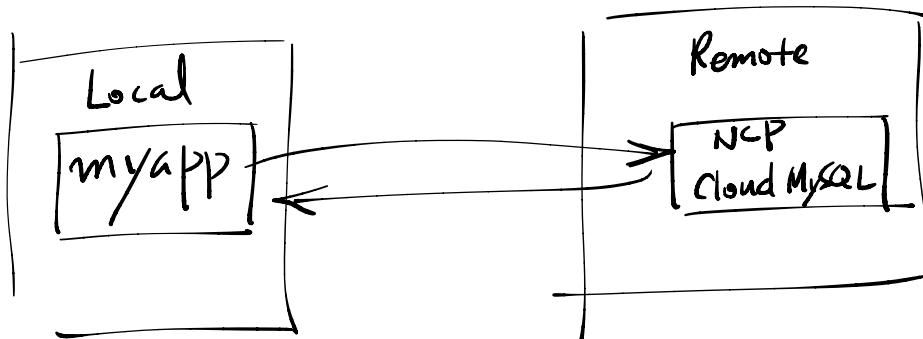
. **add () {**
 try {
 original.add();
 txManager.commit();
 } catch {
 txManager.rollback();
 }

62. Naver Cloud Platform 사용 예

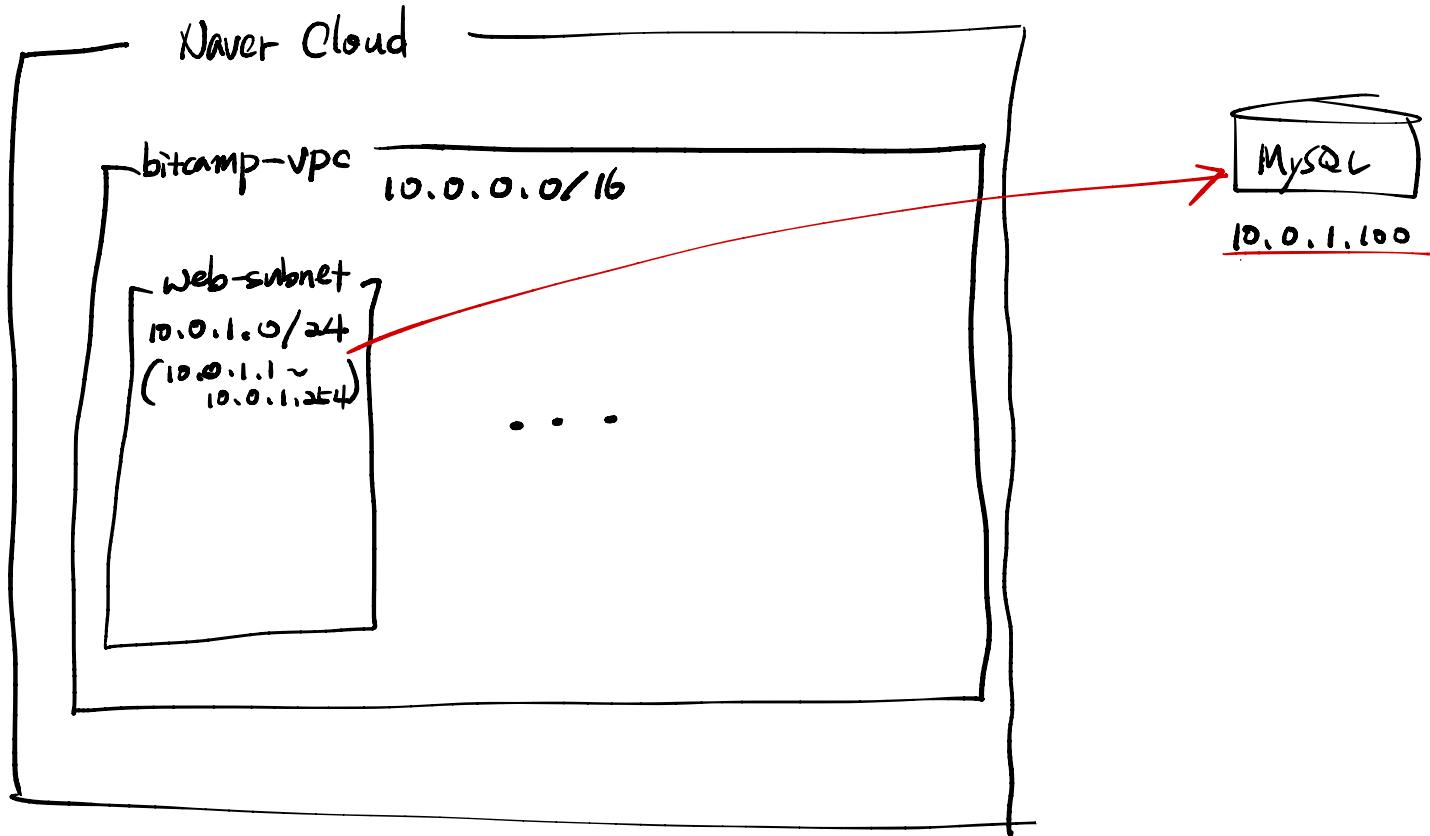
* ORI



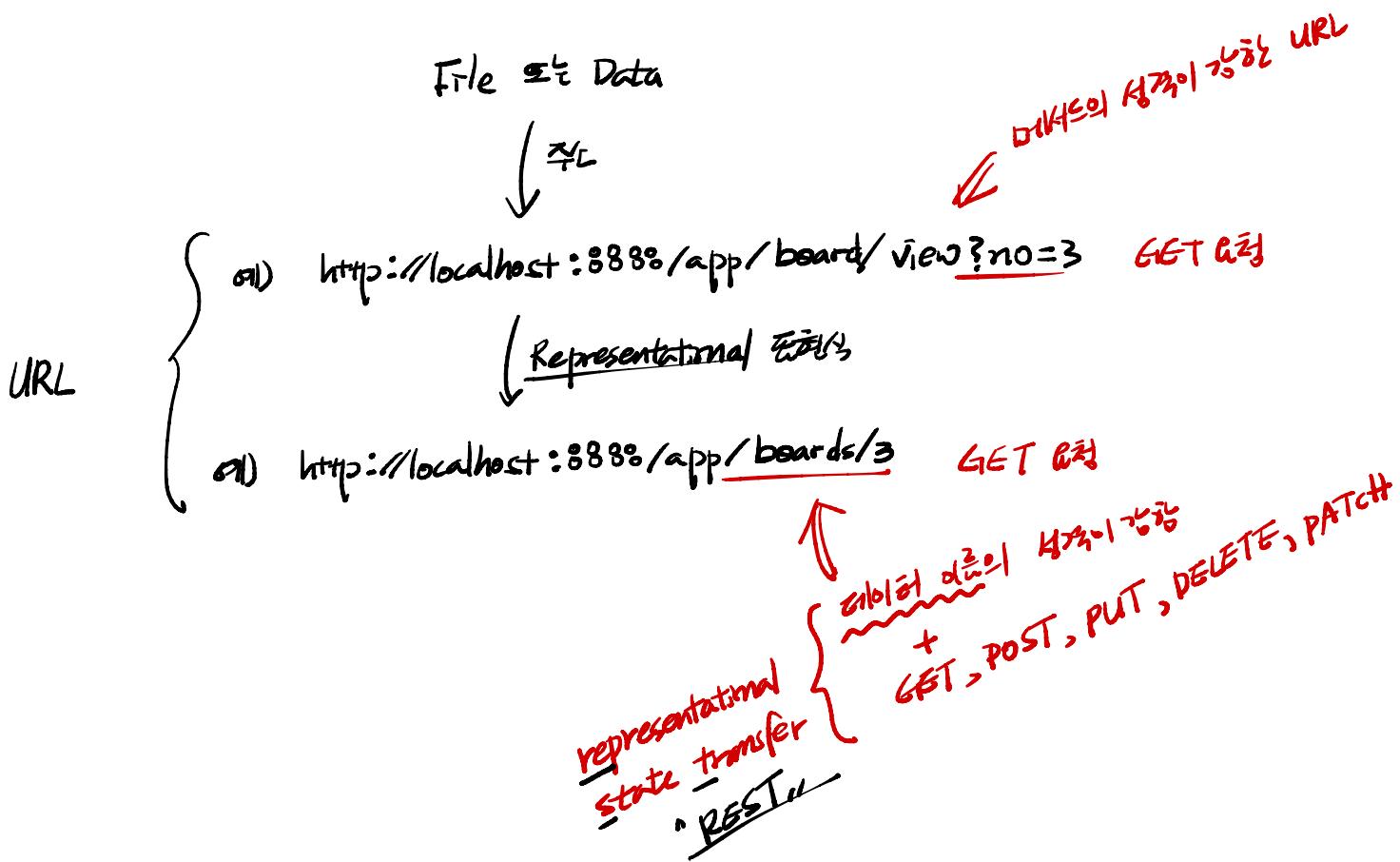
* 대입



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



* RESTful API



* function → REST API

다른 프로토콜

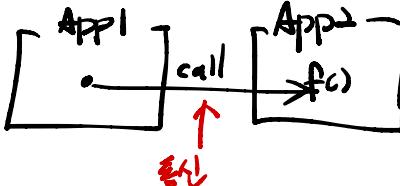
① function

(RPC)

② Remote Procedure Call

(RMI)

③ Remote Method Invocation



C 등 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

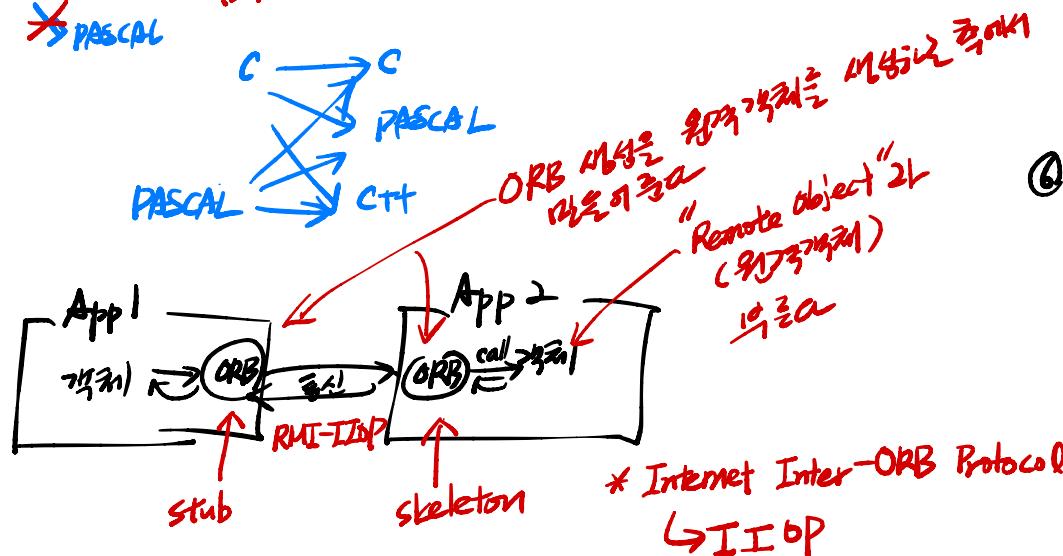
같은 인터프리터
만든 App끼리
함수를 호출하는
방식

C → C
✖ PASCAL

common
Object Request Broker
Architecture

자신의 언어로 만든 App끼리
함수를 호출하는 방식

C → C
✖ PASCAL
✖ C++



⑤ Web-service

✓ 파일과 HTTP 프로토콜을 이용
✓ ORB를 통과해 인터넷에 맞춰
설정되었음

↓
인터넷 규칙에 맞도록 개발된
작품을 찾기!

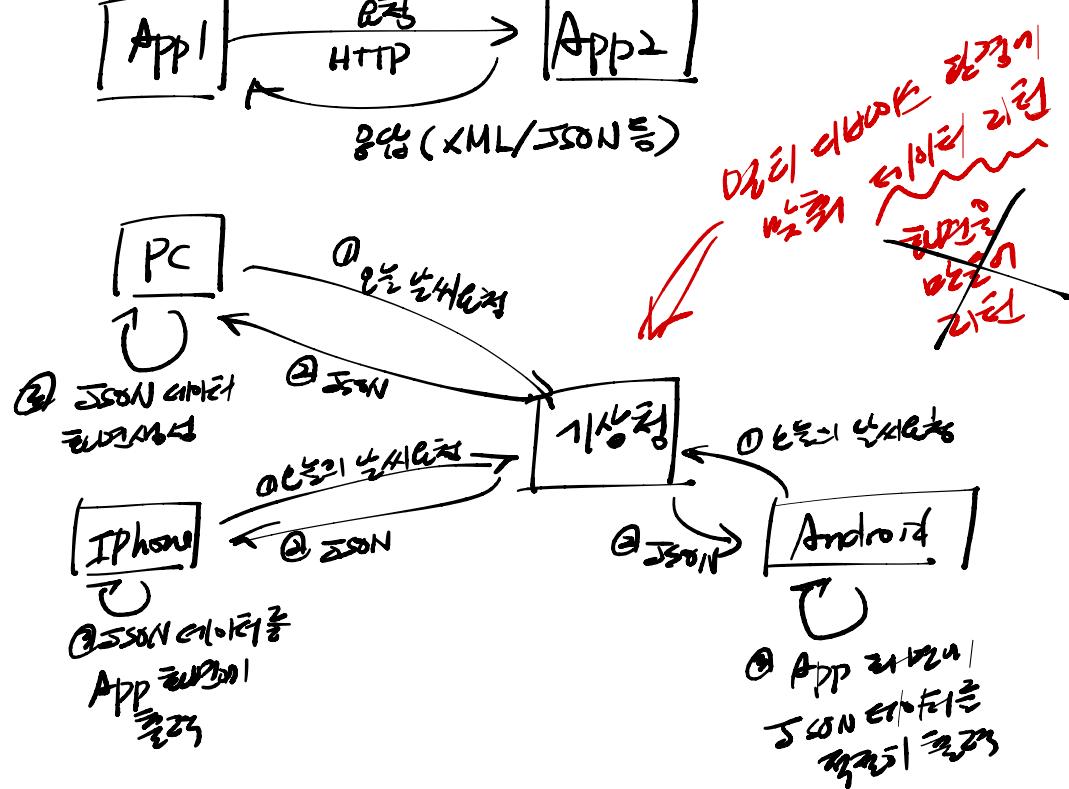
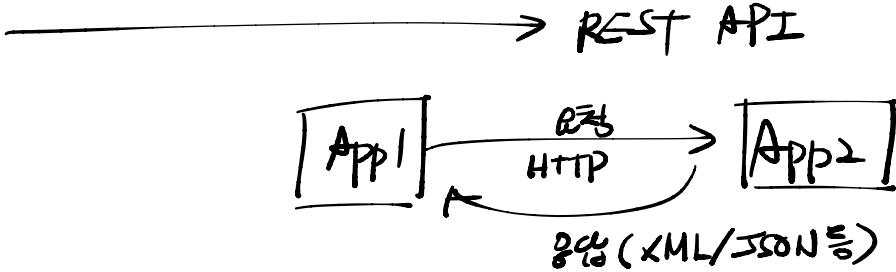
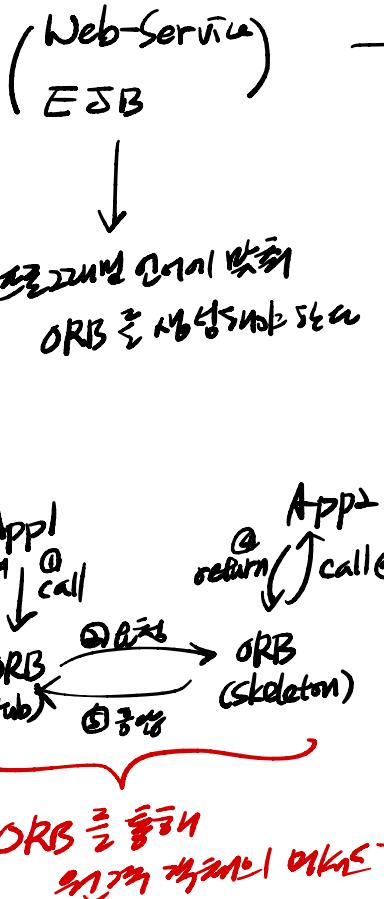
⑥ Enterprise JavaBeans (EJB)

Java에서만 가능 RMI 가짐

있음!

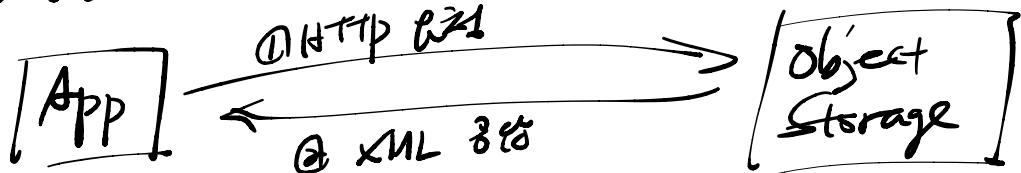
2000년 초반에는 PC를
대상으로 디바이스를 위한 EJB가
제작되었지만 초기에는 제한적이다.

* function → REST API



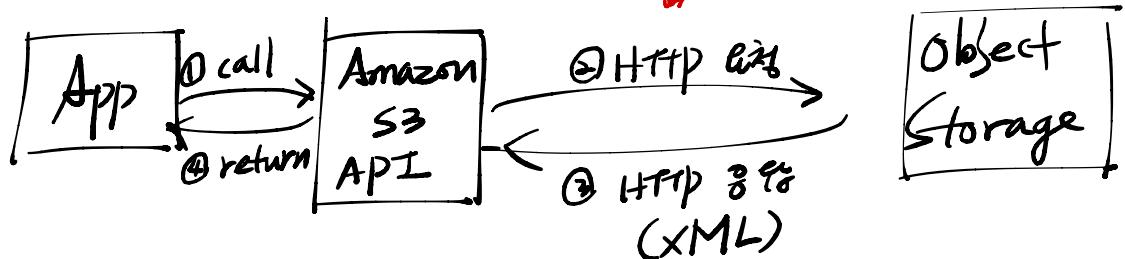
* Object Storage REST API API
↳ HTTP 퀼/값, 키!

① 직접 접근

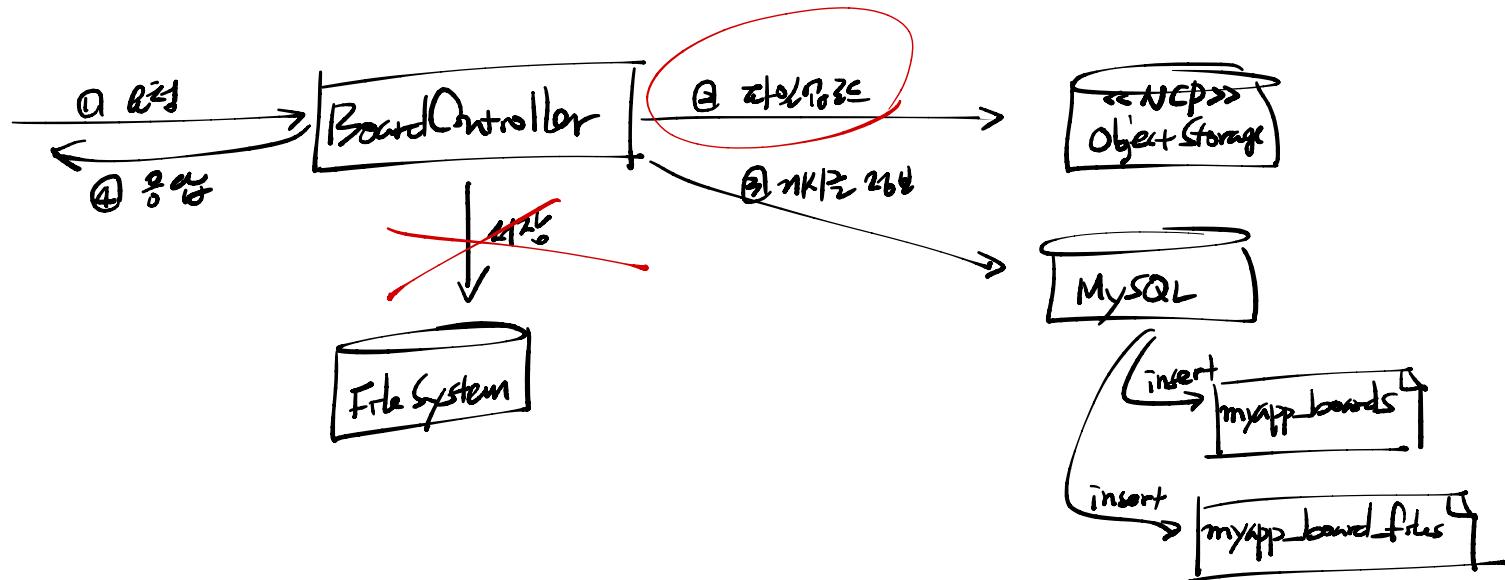


HTTP 퀼/값은 직접 접근하기 편리하다
HTTP 응답을 직접 접근하기 편리하다

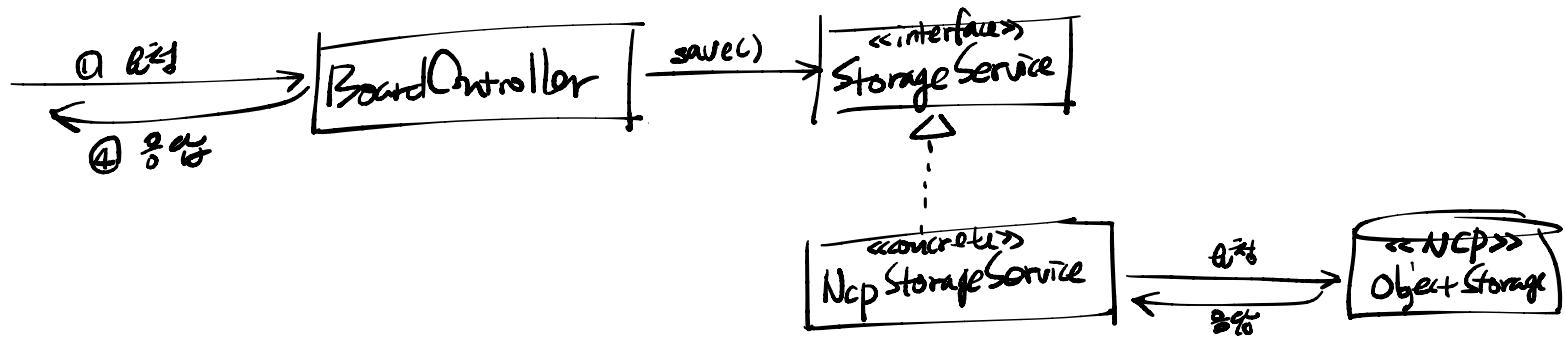
② 경유 애플리케이션



* 내부로 첨부파일은 NCP의 Object Storage에 저장

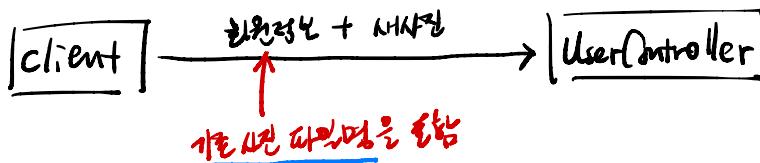


* 게시글 첨부파일은 NCP의 Object Storage에 저장 → NCP 연동로직을 서비스 객체로 분리



* 사용고객과 보안

① 보안에 악용은 예 : 회원정보 변경



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



이제, 변경, 삭제 시

기존 정보를 사용할 때는

항상 새롭게 프로그램 사용하자!

기존 기반코드를 클라이언트에서 끌어올리자! → 꼭 새롭게 프로그램 사용하자!

- i) 기존 사용자인증 → 클라이언트가 보낸 세션키로
세션을 흡수하여 쓰자
- ii) NH 네임 지정 → 사용자 지정
- iii) 정보변경 → 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

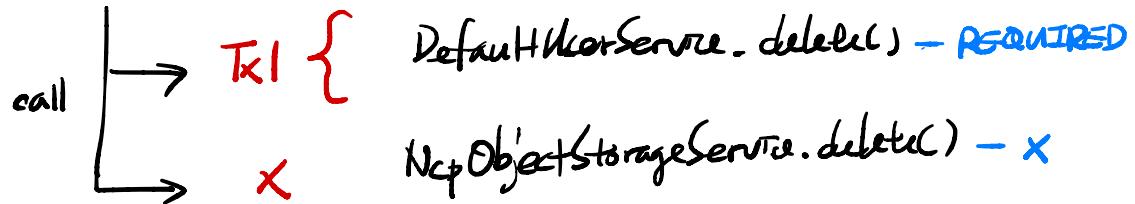
UserController.delete() X
↓ call
Tx1 { DefaultUserService.delete()

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

* 키워드 어노테이션 정리

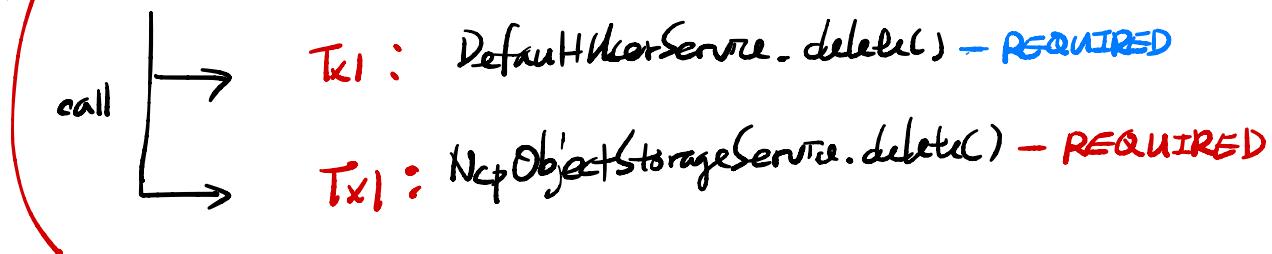
① 허용

X UserController.delete() X

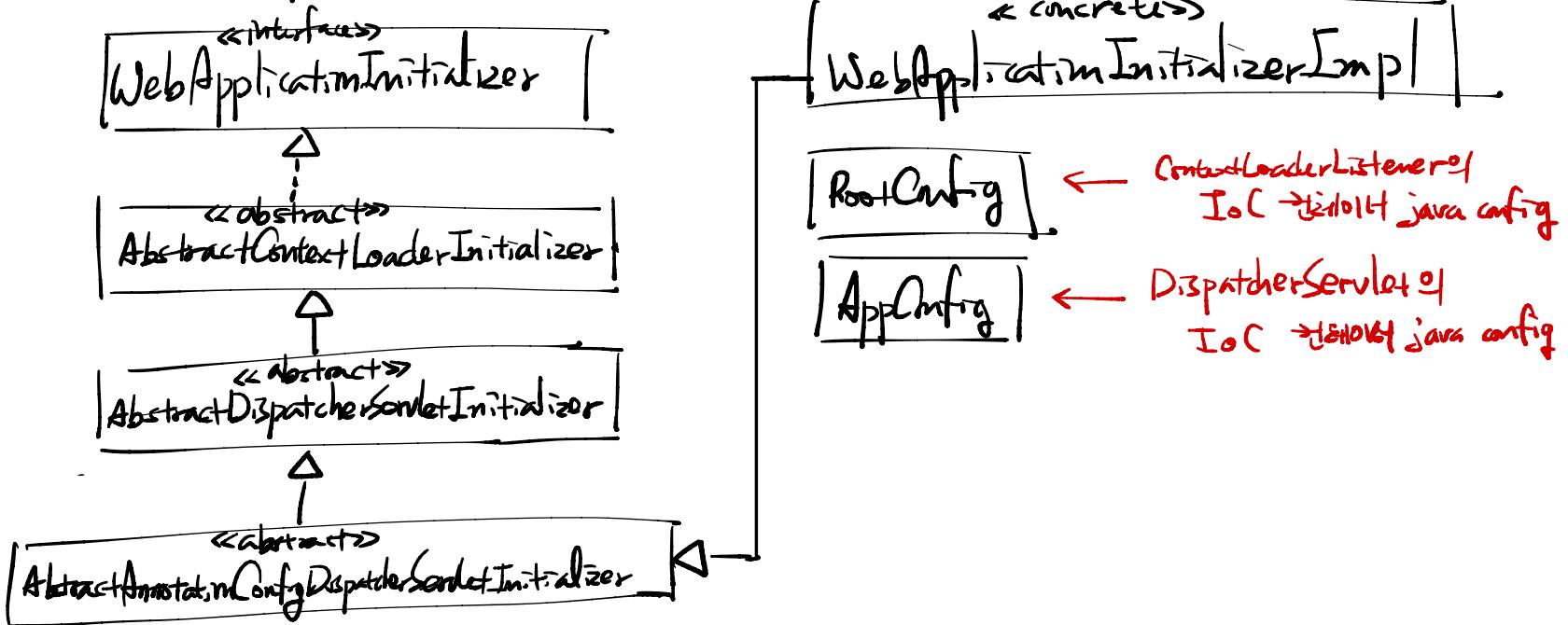


② 허용

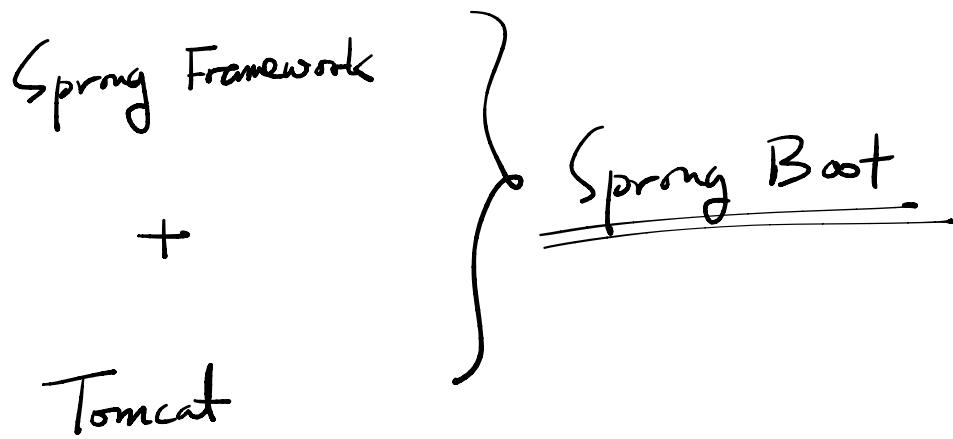
Tx1 UserController.delete() - REQUIRED



63. WebApplicationInitializer



65. Spring Boot mpls



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

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

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 2150121 |
| 1 | _____ | |
| 2 | _____ | |
| 3 | _____ | |
| 4 | _____ |) 2 2150121 |
| 5 | _____ | |
| 6 | _____ | |
| 7 | _____ | |
| 8 | _____ |) 3 2150121 |
| 9 | _____ | |
| 10 | _____ | |
| 11 | _____ | |
| 12 | _____ |) 4 2150121 |
| 13 | _____ | |
| 14 | _____ | |
| 15 | _____ |) 5 2150121 |
| 16 | _____ | |
| 17 | _____ | |

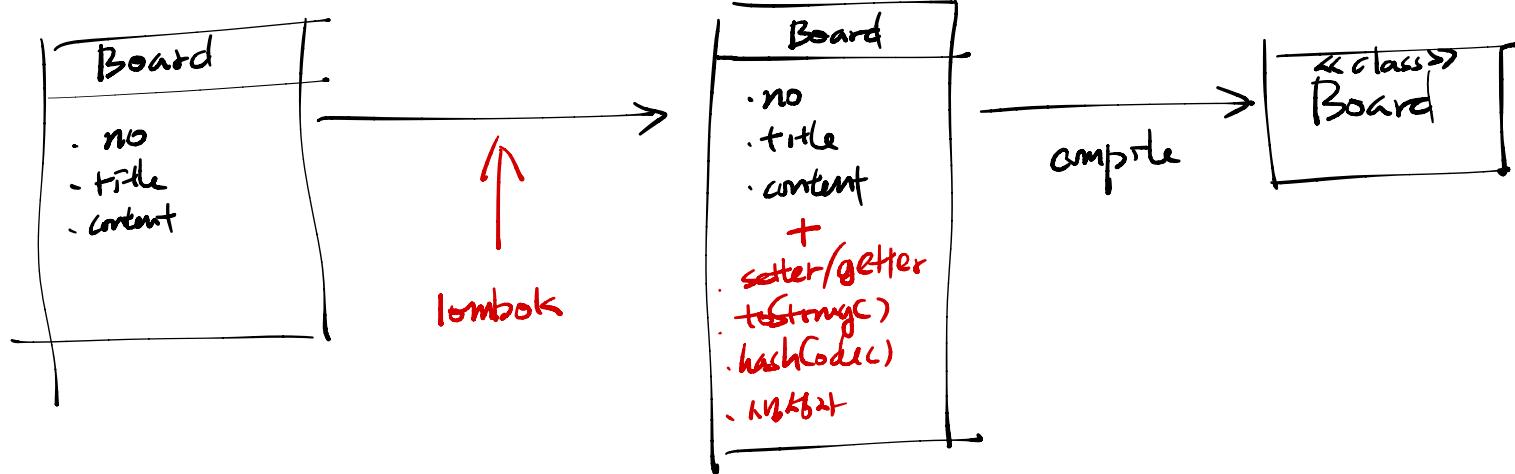
67. Lombok 2010-2021年版

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

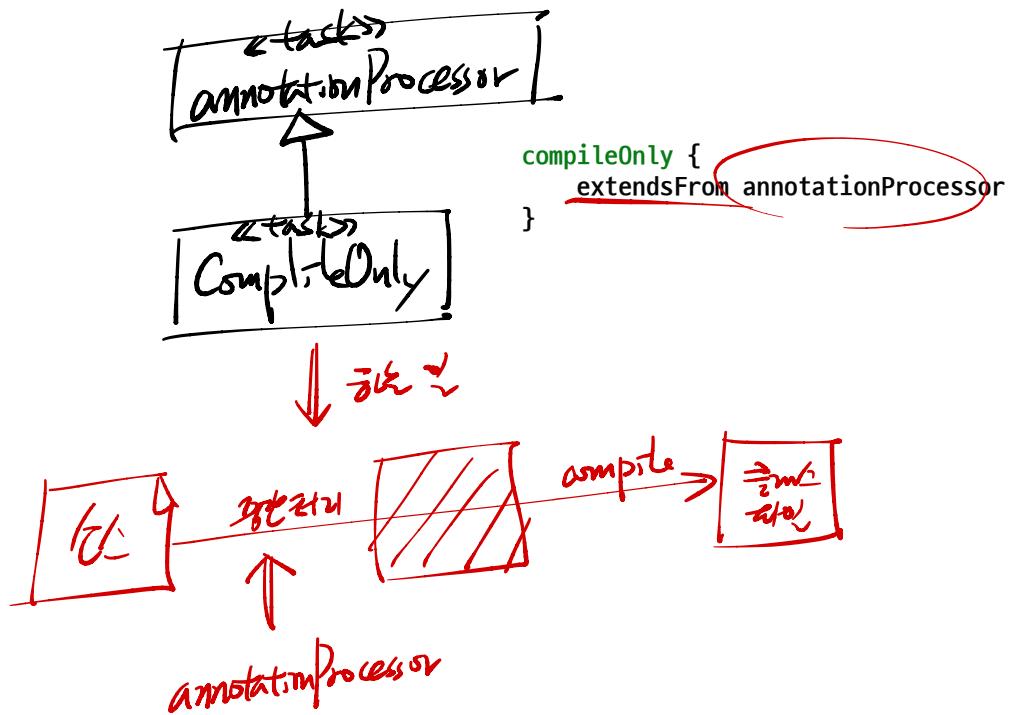
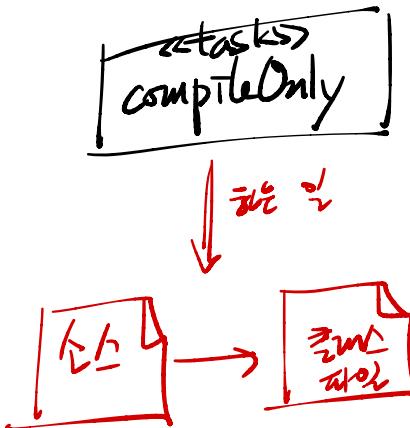
Gradle



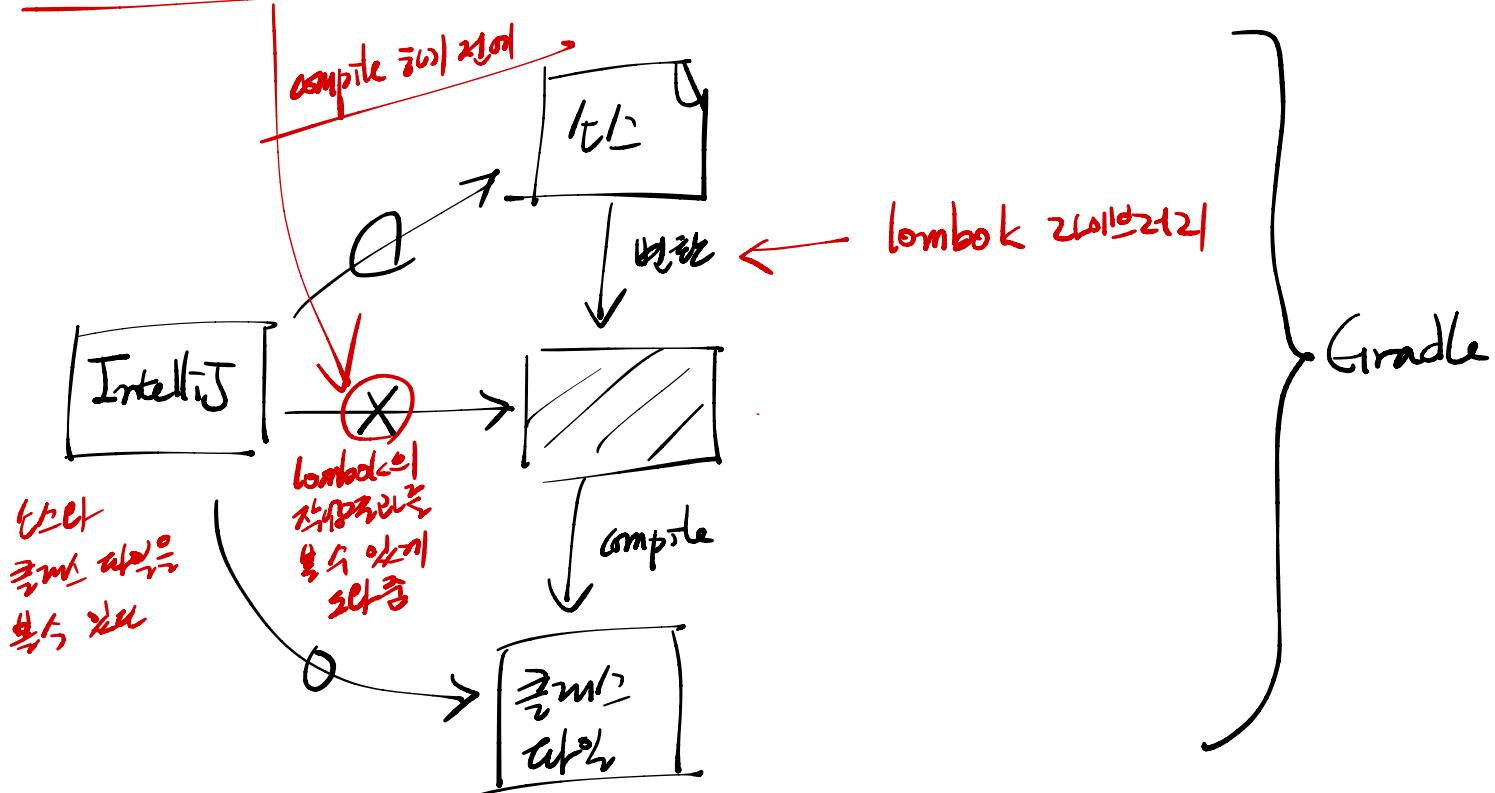
Build



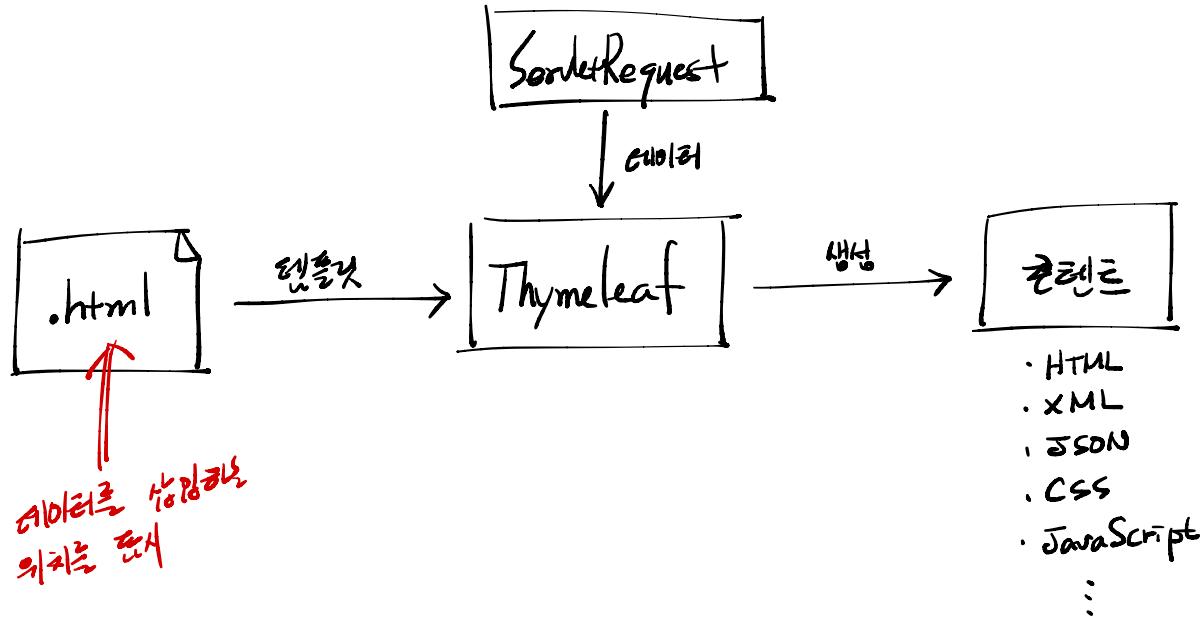
* Gradle et Task



* lombok IntelliJ 풀이방법의 차이



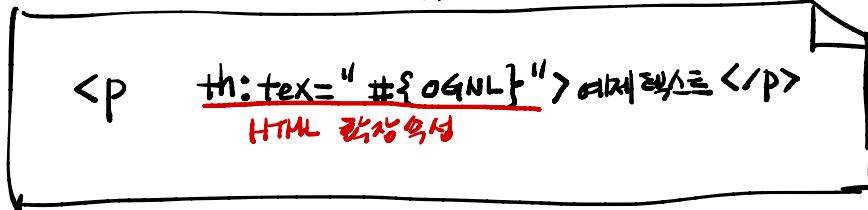
* Thymeleaf 템플릿 인자



* Thymeleaf 템플릿과 HTML

.html

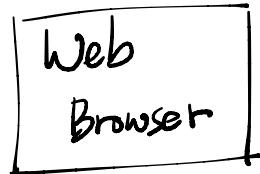
템플릿 파일



JSP → \${변수명}

Mybatis → #{} выражение

Thymeleaf → #{변수명}



HTML 편집과 편집을 허용



디자이너

디자이너는 Thymeleaf가
수동화한 코드에 맞게 편집
하고 편집을 확인할 수 있다

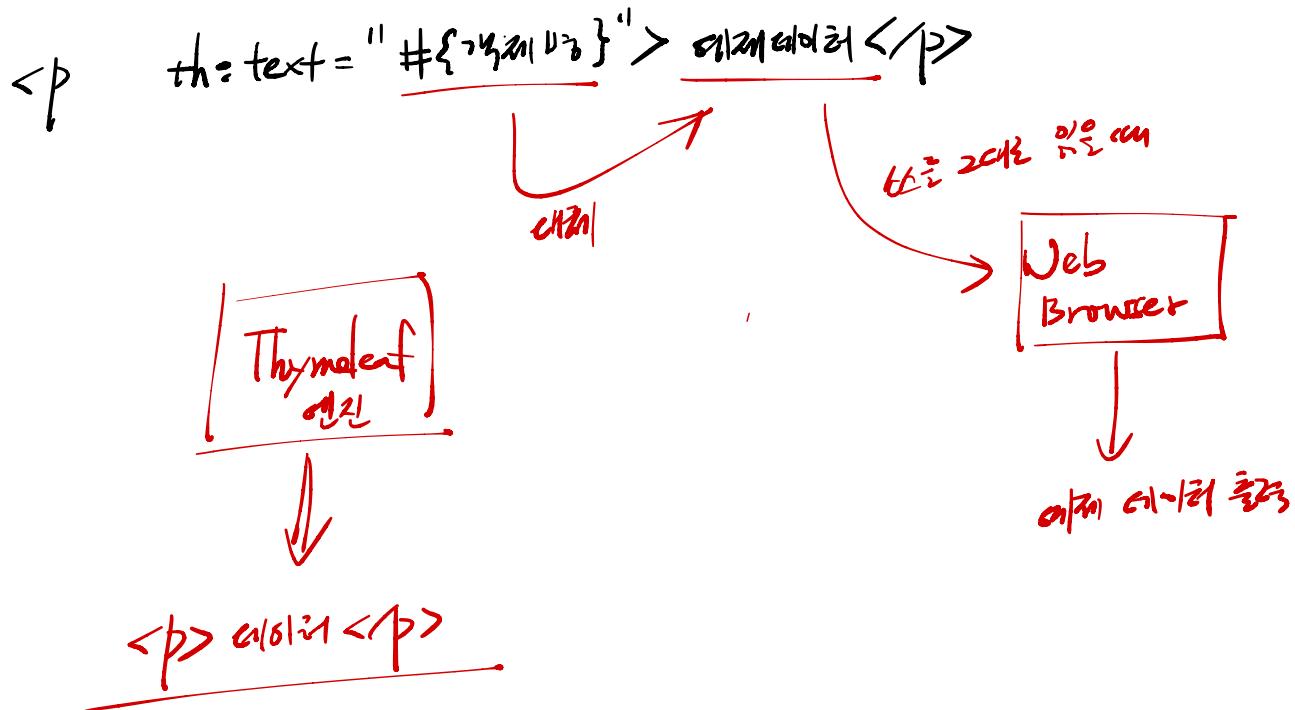
Thymeleaf의 특징

- ① JSP와 다르게 템플릿이 HTML 파일이다.

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

- ② Thymeleaf 파일이 HTML 편집을 허용하지 않는다.
스스로로 편집을 해 편집이 되지 않는다.

* 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

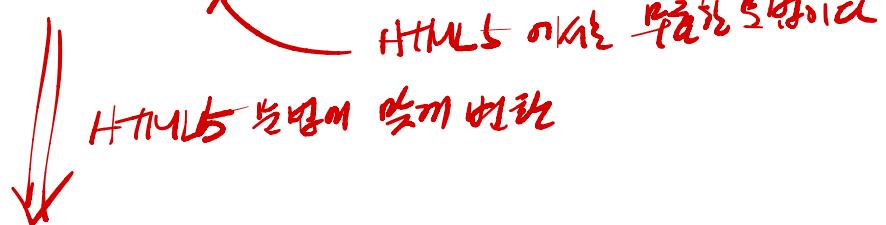
※ XML namespace 처리는
HTML 처리는 처리

th: 을 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 사용

} HTML 키워드가 아닌
단순 문자로 취급하기
위함.

data-th-utext

unescape

escape 문자를
차단하지
않는다

HTML 키워드가
포함된다.

<u> 안녕<u> 친구님.

→
단일 텍스트로만
HTML 특수기로
변경된다.

* 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)

69. Handler & Method ArgumentResolver چی가 뭔가?

request handler

파라미터 ← 사용자에게 주는 값들

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

====

}

어디서?

request handler of
atmosphere 티켓
MultipartFile

기본값
세션값
atmosphere 티켓

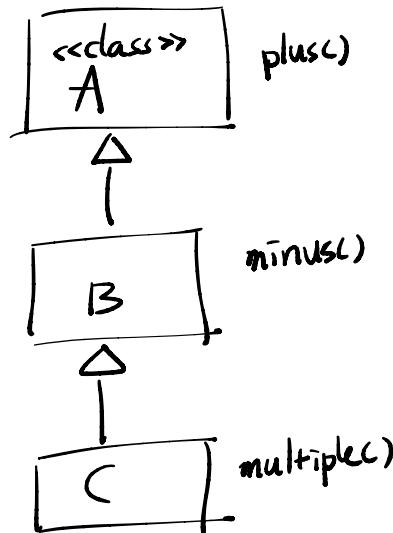
} {
 SendRequest
 ReceiveResponse
 HttpSession
 @RequestParam
 :

* 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 레퍼런스가 B type의 인스턴스를 할당할 수 있나?

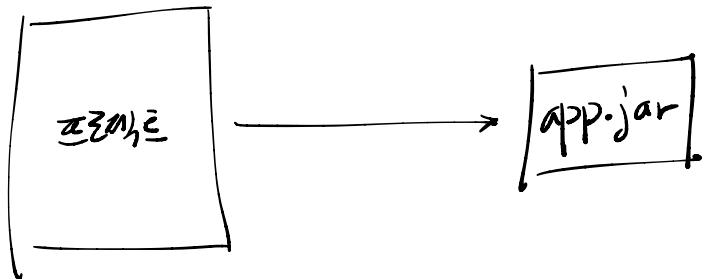
A obj = new B();
obj.plus()A+&

$B \text{ obj}$ ~~has~~ $A(1)$

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

10. Jenkins et Docker이 결합된 CI/CD 활용하기

- ① Dockerfile 작성 및 실행 확인

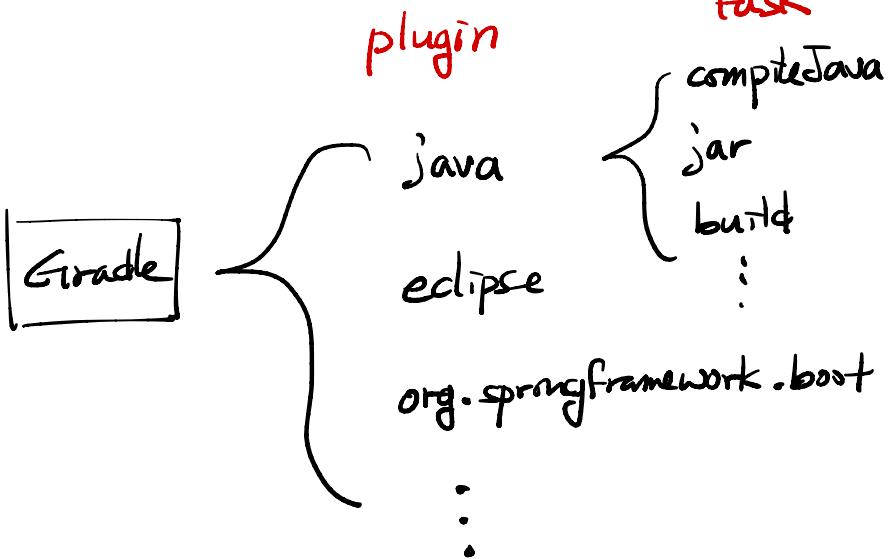


\$ gradle build --console=plain

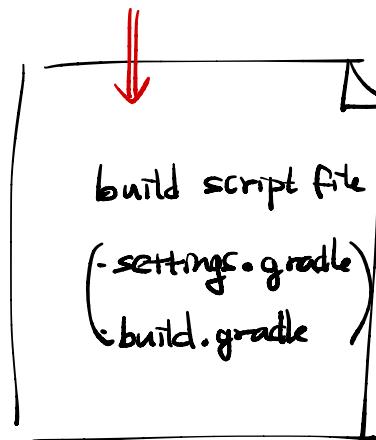
\$ java -jar .../myapp.jar

task ~~MyTask~~
↓

* Gradle - plugin - task - build script file



(프로젝트
구조) on chit 보정



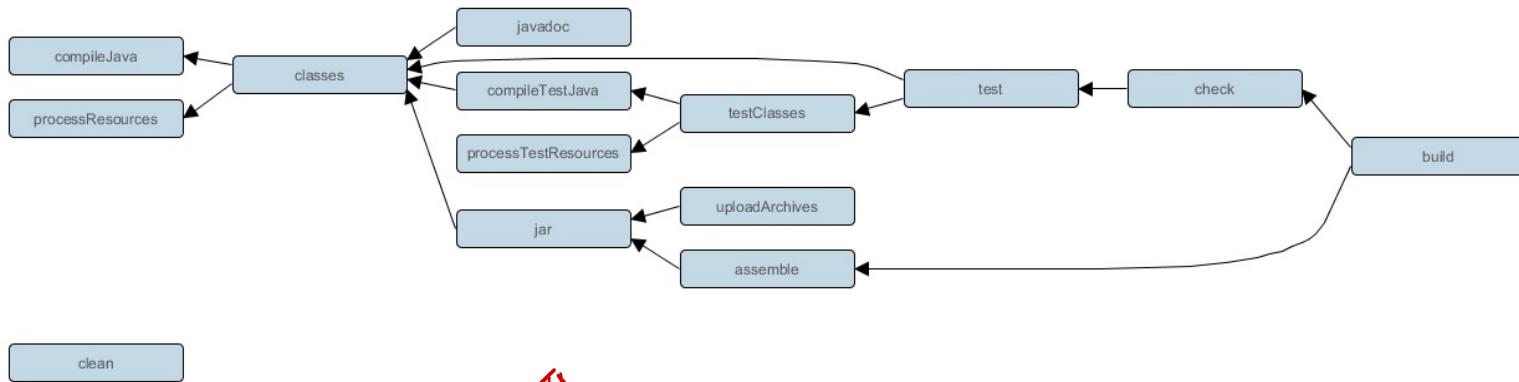
java {} - }

bookJar {} - }

:

* Gradle 키워드

\$ gradle build
build task의 종속성도 포함된 디펜던시 그래프



'java' gradle 툴과 거의 대체적 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

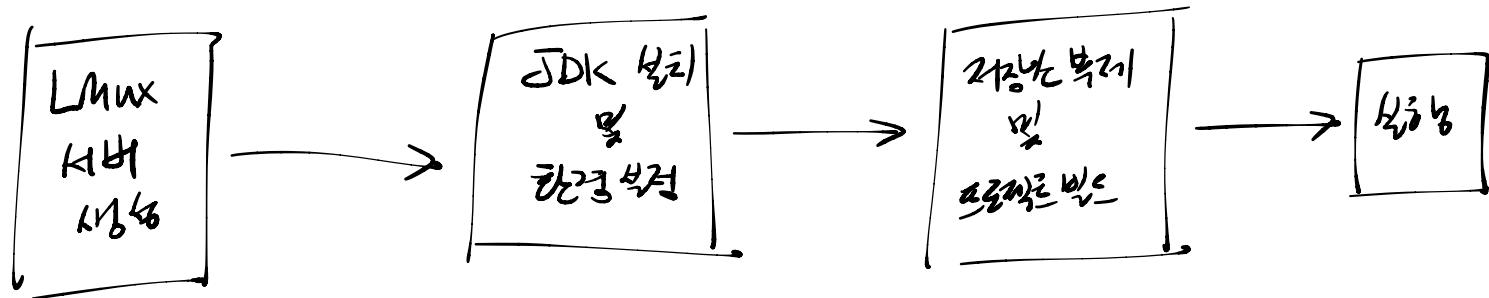
#java \Rightarrow m D

Java (space)
Java Java Java
Program Argument

#java -Dm=2 -Dn=2 \Rightarrow m=2
JVM n=2

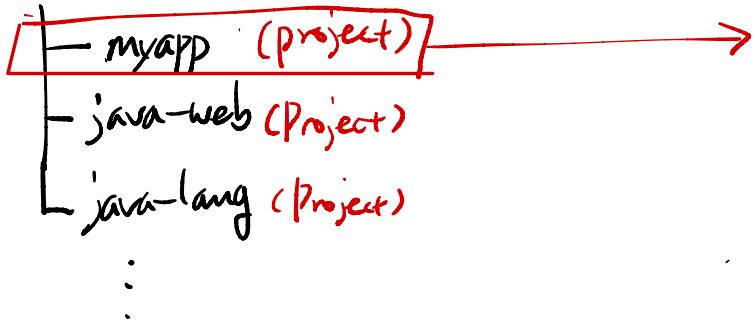
10. Jenkins et Docker을 이용한 배치 사용

- ③ 클라우드에서 진단을 위하여 구축한 애플리케이션 배치 파일 실행



* 언제 Repo 할지

bitcamp-mystudy (Repository)



언제 Repository 할지

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

pull or push?

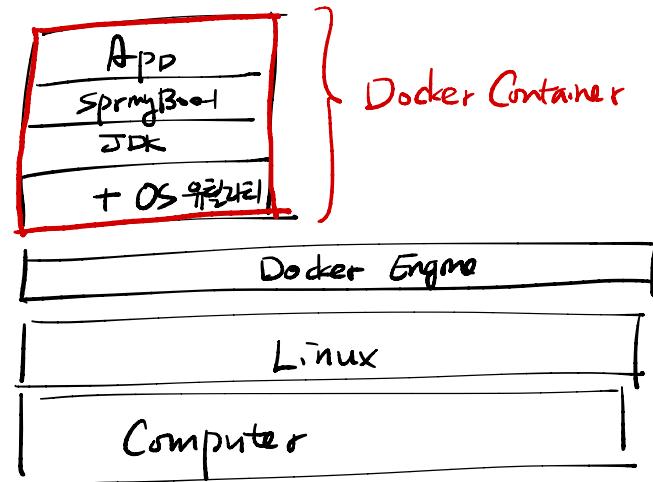
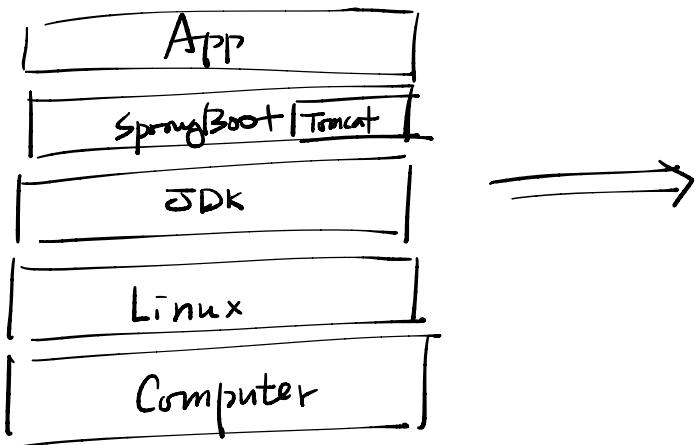
Repo. 단위로

작성된다

↳ 파일/파일夹을 선택

10. Jenkins et Docker'i oþisom iþnizi stüft

④ Docker'inin iþi nelerdir?

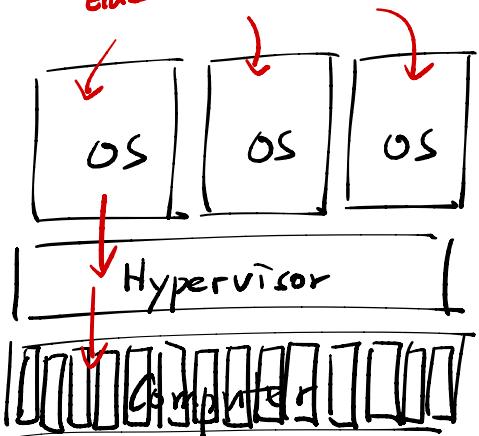


* 가상화와 Docker 차이점

① Type I (bare metal)

a) Xen, KVM, Hyper-V

Guest OS



Guest OS +
Computer = H/W 할당 및 관리
할당 및 관리

a) 클라우드 컴퓨팅

* 가상화

H/W는 물리적으론
여러 개로 훈련하는 경우,
공유하는 경우

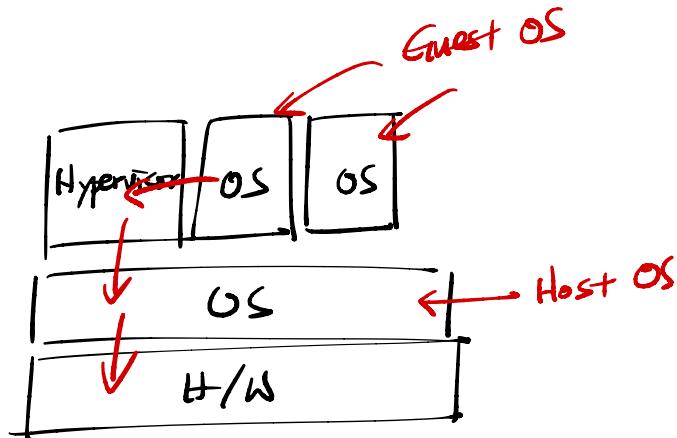
H/W 할당 및 관리
Who? (H/W 할당 및 관리)
(CPU, 메모리)

여러 OS가
갖는 H/W (CPU, memory)를
다른 사용자에게 할당하는 경우
공유 모드로 운영되는 경우
제어

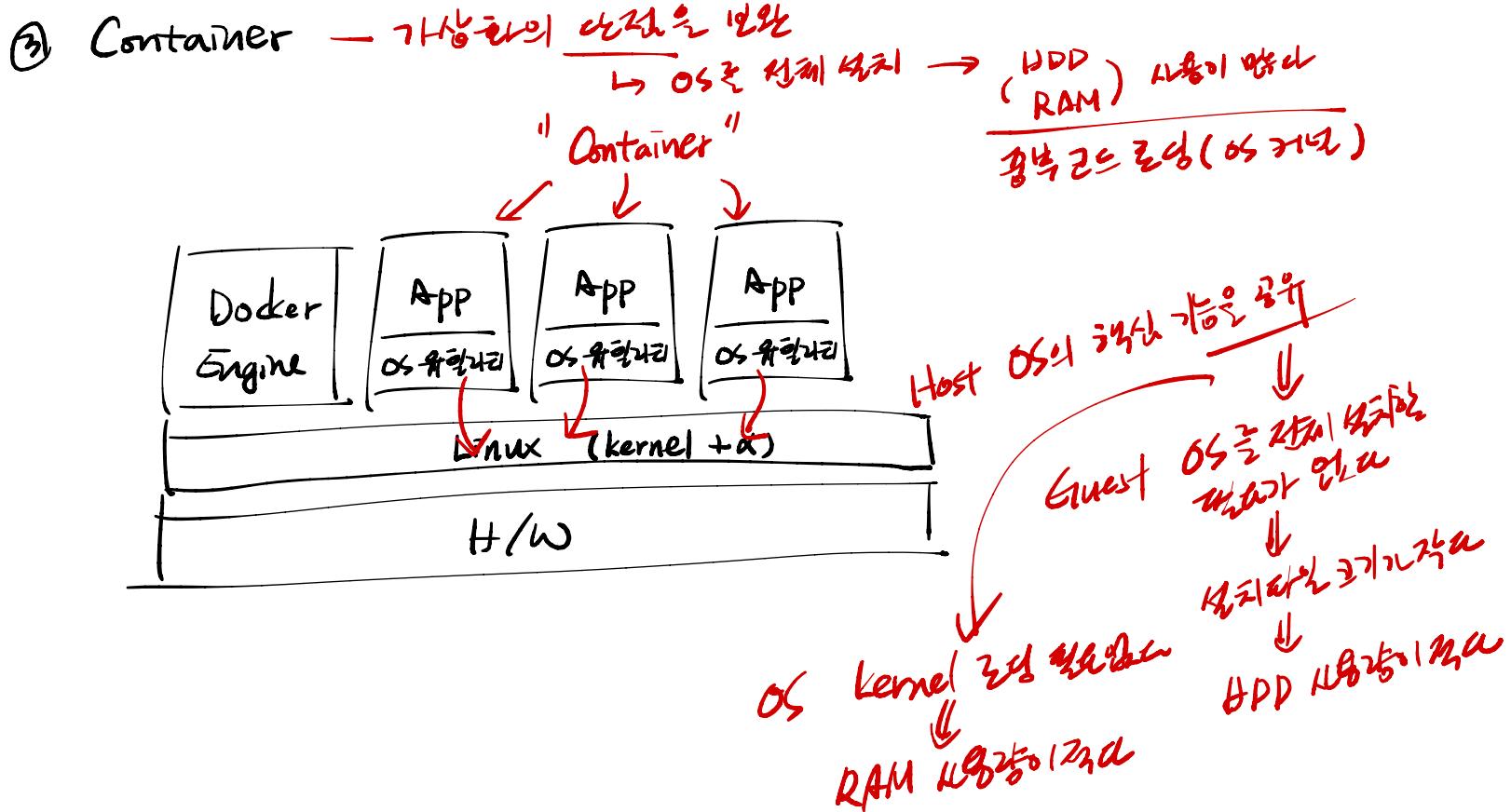
* 가상화와 Docker 관계이야

② Type 2

a) VMWare, VirtualBox, Parallels 등



* 가상화와 Docker 차이점



* container 허브
↑ 정답 ↑ 허브 컨테이너

```
docker run -i -t ubuntu:14.04
      image 이름    tag 번호
```



① local에서 허브로 이미지를 찾는다



② docker 커맨드 실행으로 한다



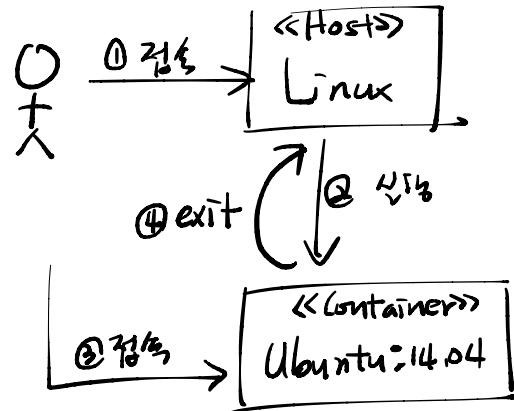
③ 이미지를 가지고 컨테이너를 시작한다.



④ 컨테이너를 실행한다



⑤ 컨테이너 모드로 실행된다



* Docker Image Համակարգ

ՀՀ օլույն պահ

docker pull centos:7

ՀՀ շեշտուի այլ կերպ

docker create

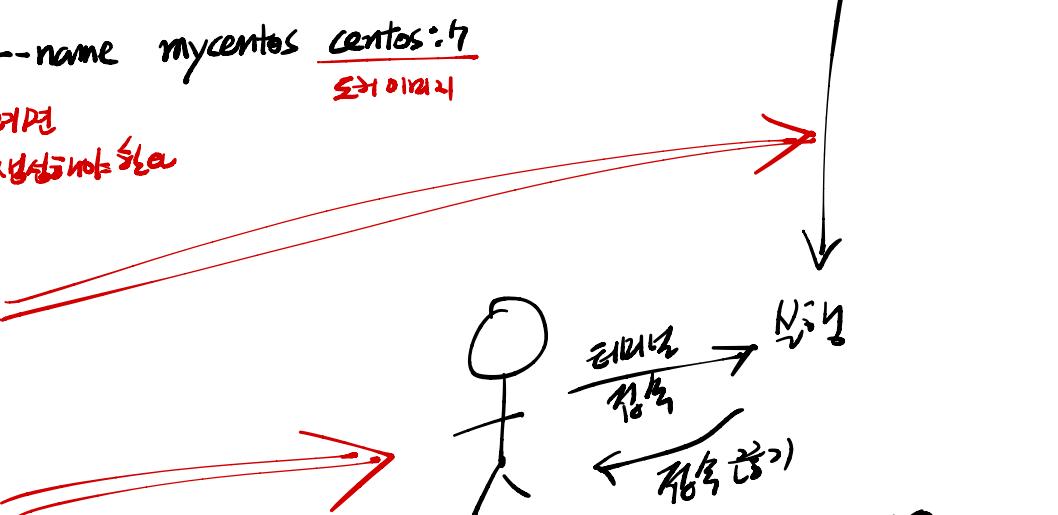
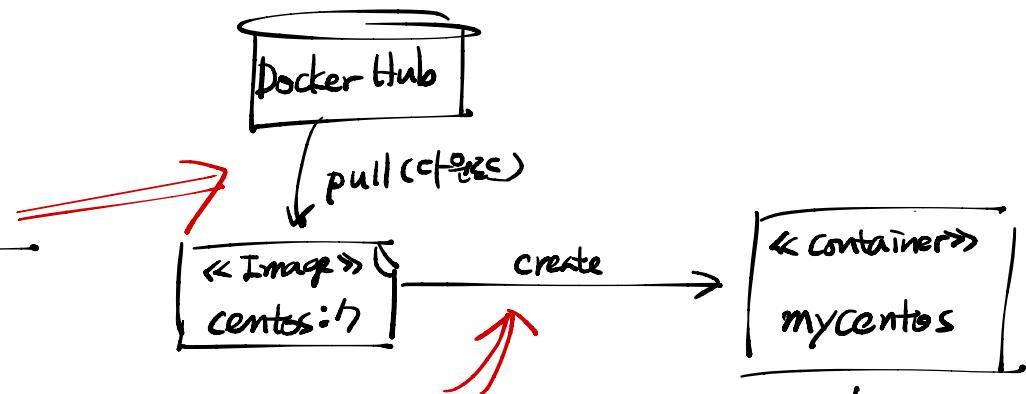
-i -t --name mycentos centos:7
սահմանափակություն
առաջարկություն
առաջարկություն

ՀՀ շեշտուի այլ կերպ

docker start mycentos

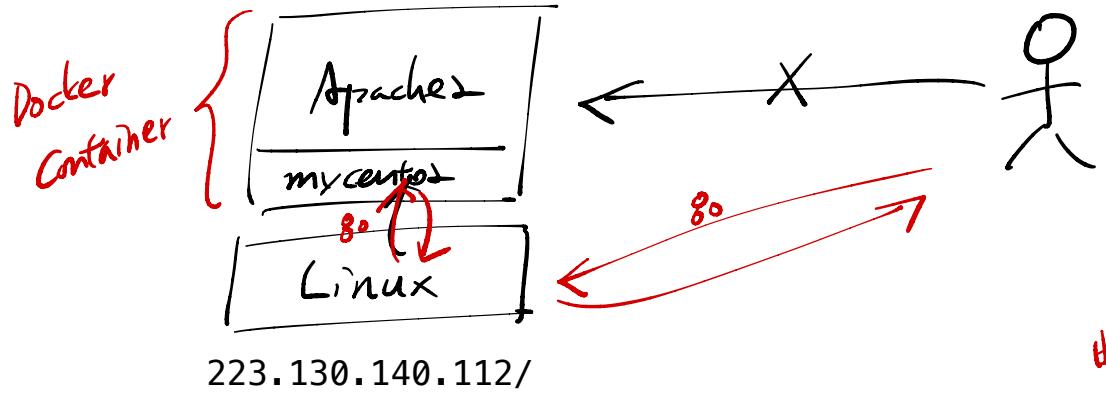
ՀՀ շեշտուի այլ կերպ

docker attach mycentos



CTRL+P → CTRL+Q

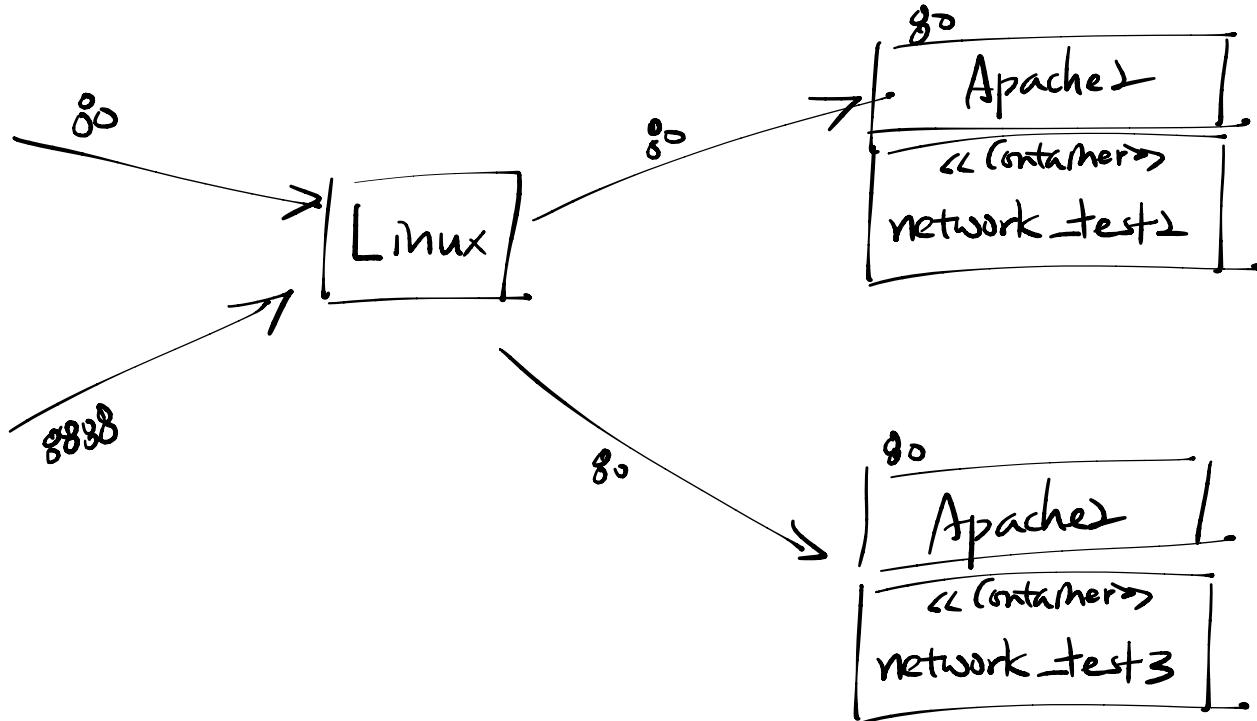
* Docker 컨테이너에서 포트开放하기 → 포드포워딩으로 처리



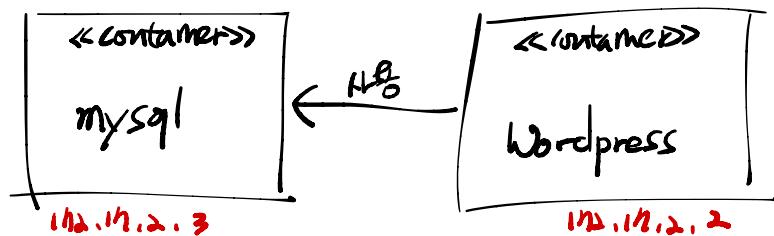
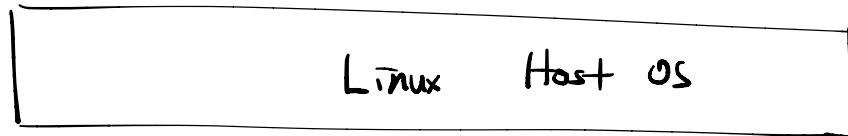
`docker run -i -t --name mycentos -p 80:80 ubuntu:14.04`

Host
Container
포드포워딩

* 유익한 기술이나 툴 소개 → 포드포워딩의 처리 II



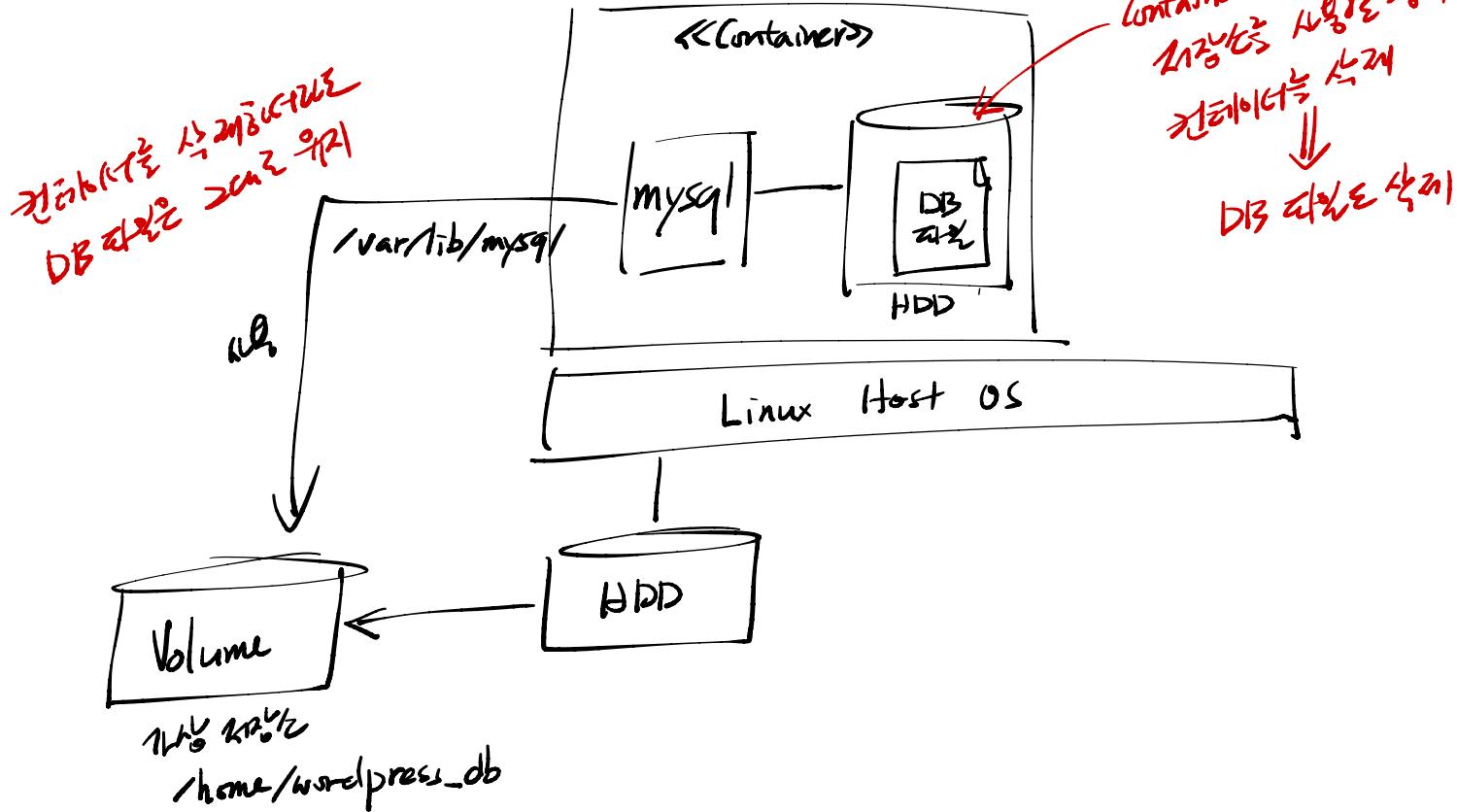
* Docker 환경 - Wordpress 구조



--link 컨테이너명:포트명

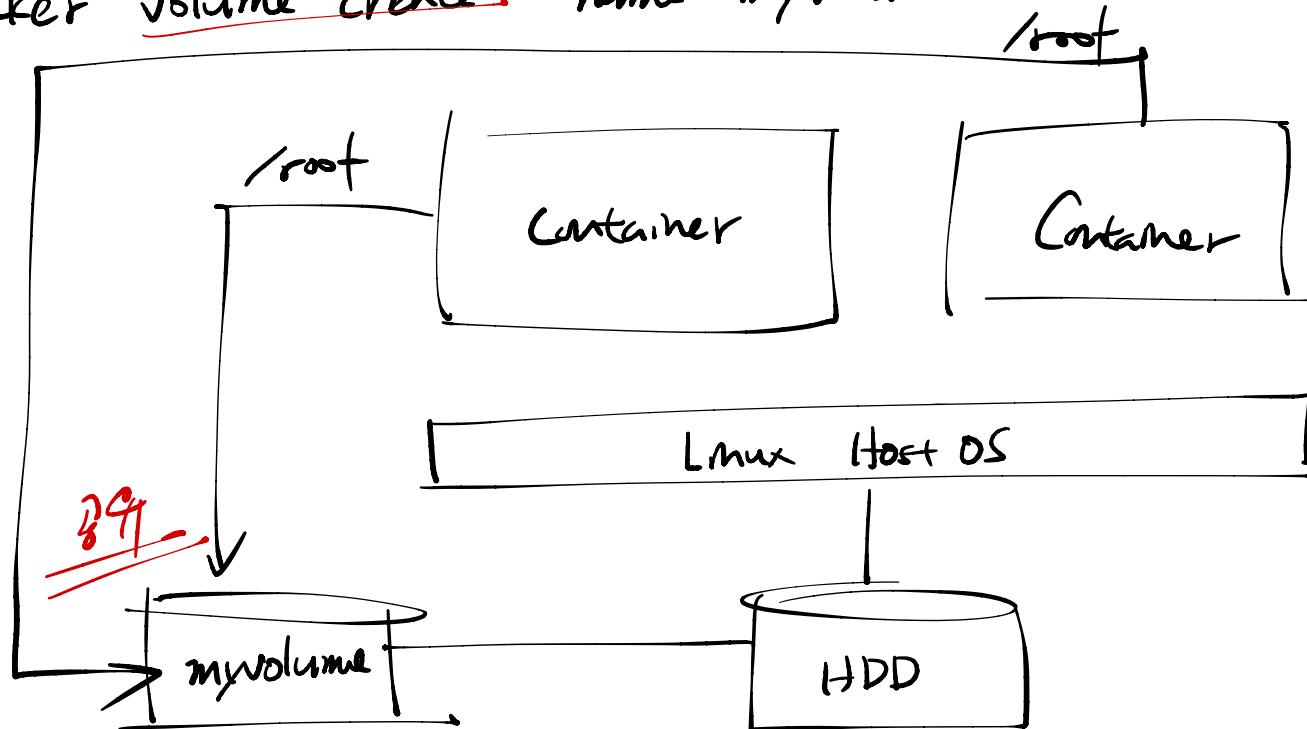
↑
IP 주소는 반드시 같아야 한다.
컨테이너 이름과 포트번호가 같아야 한다.

* Docker Volume - Host의 디렉토리를 지정하는 경우



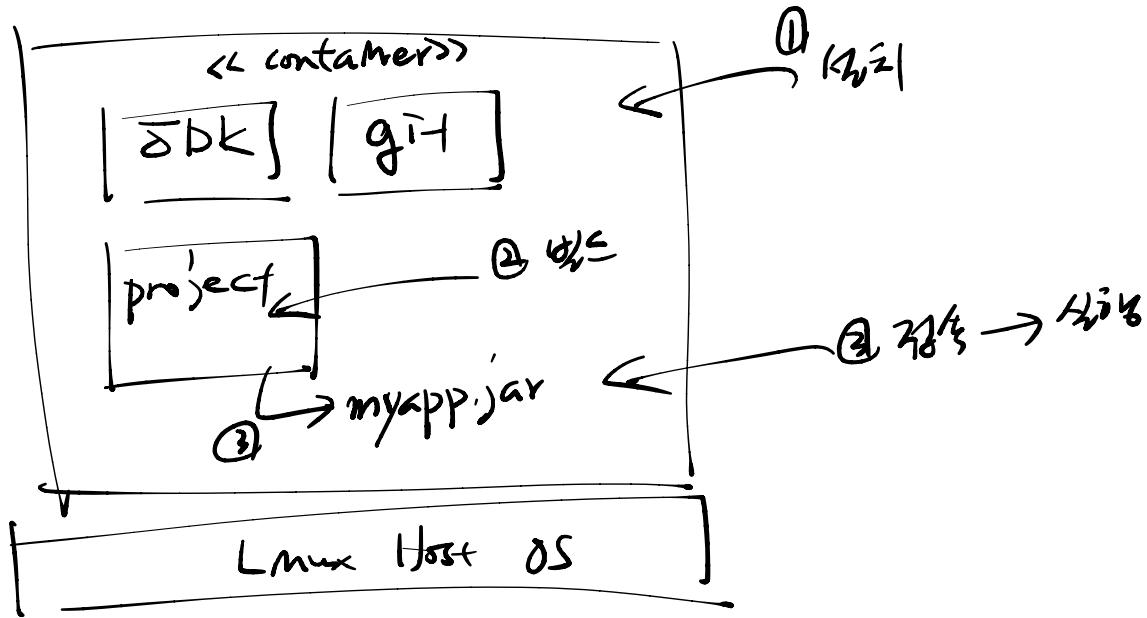
* Docker Volume - Docker의 Volume 풀이 되어

docker volume create --name myvolume



이거? docker가 뭘까
↳ docker 이미지를 실행할 때 쓰는 공간!

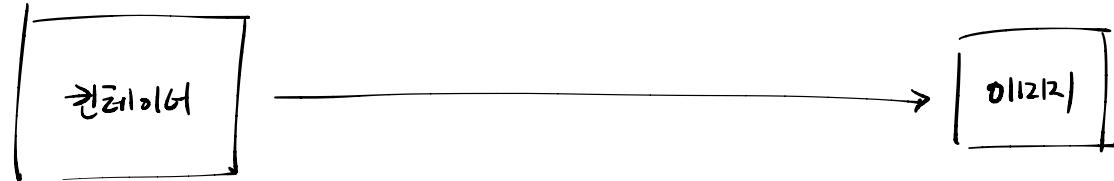
* SDK와 git이 있는 배치 container 만들기



* 도커 이미지 만들기
↳ 컨테이너 설치 파일

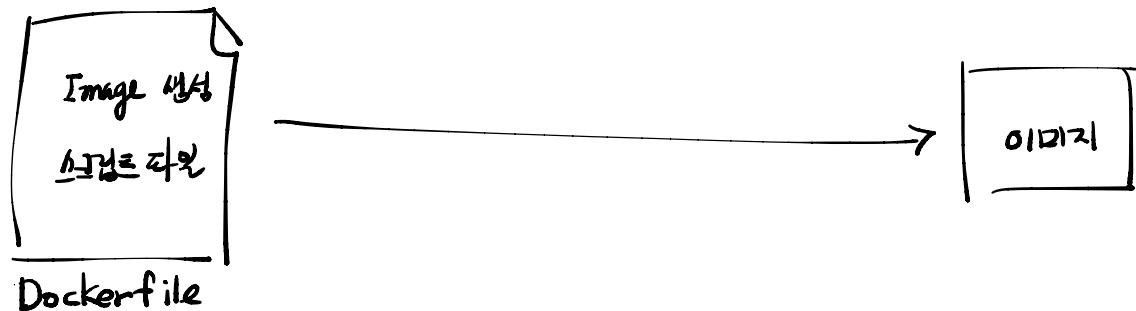
① docker commit 옵션 컨테이너명 REPOSITORY:TAG

a) sudo docker commit -a "제작자" -m "설명" 컨테이너명 이미지명:태그명

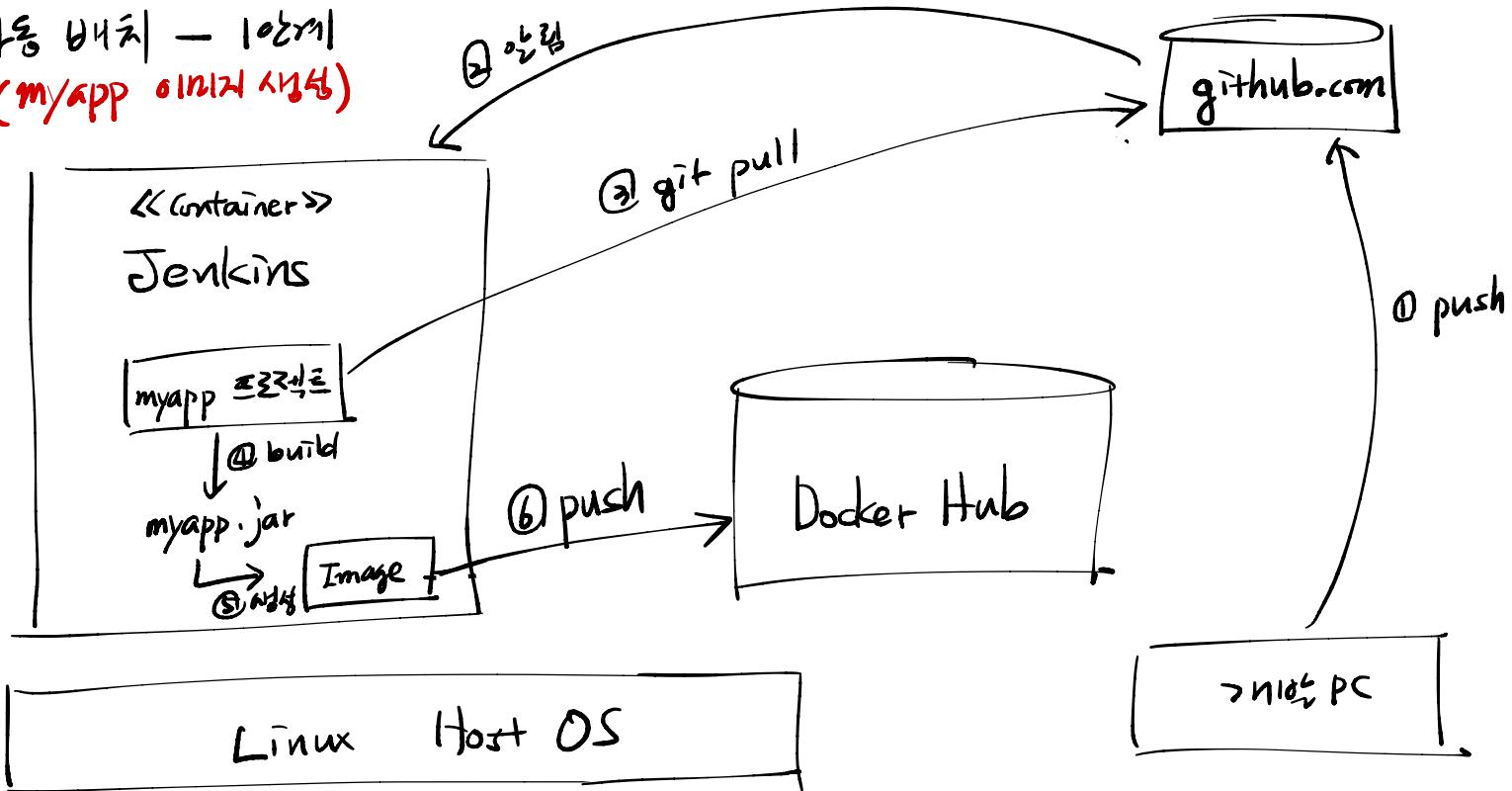


② docker build -t eomjinyoung/bitcamp:myapp .

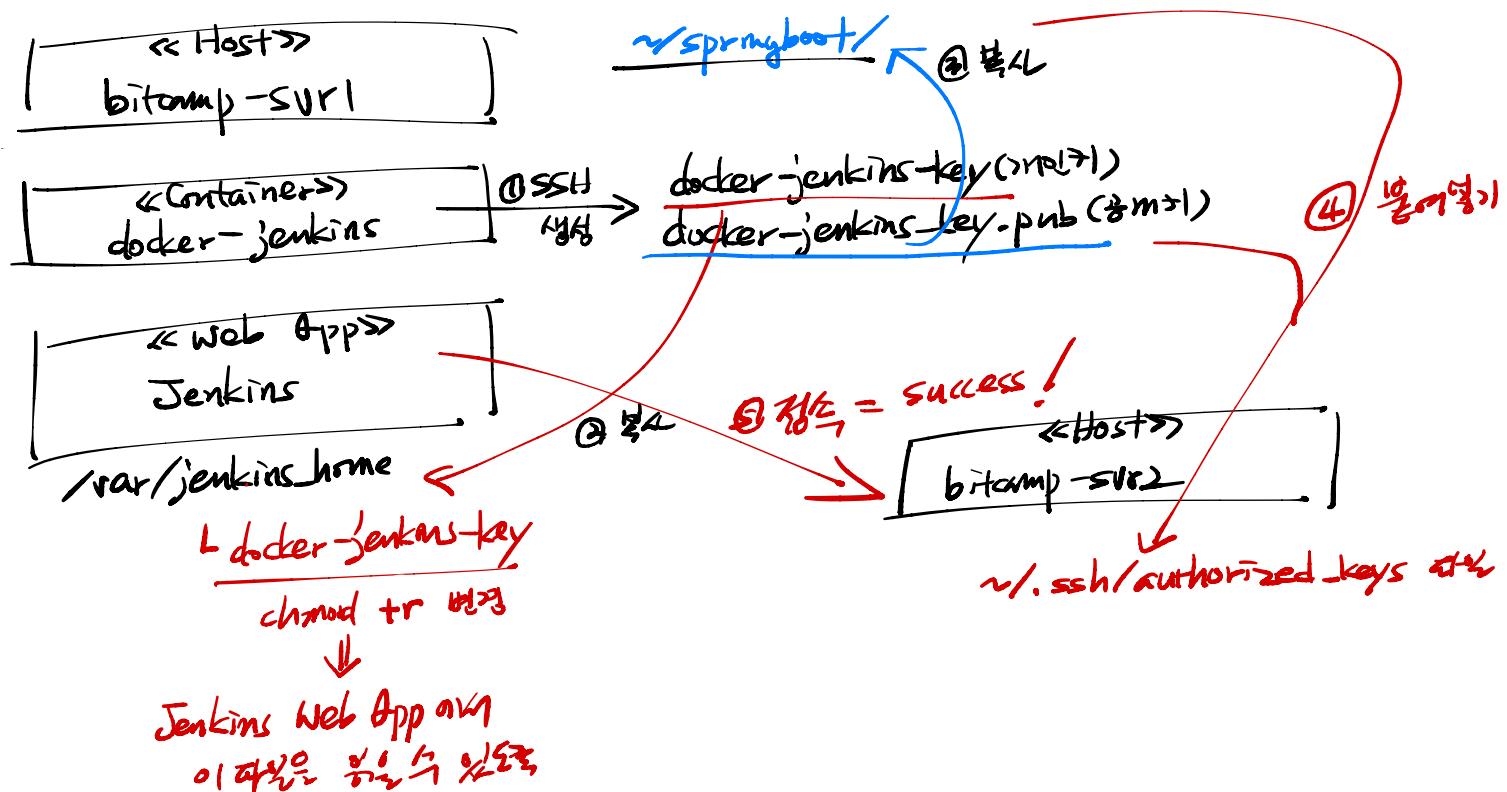
도커빌트파이썬 이미지명 태그명



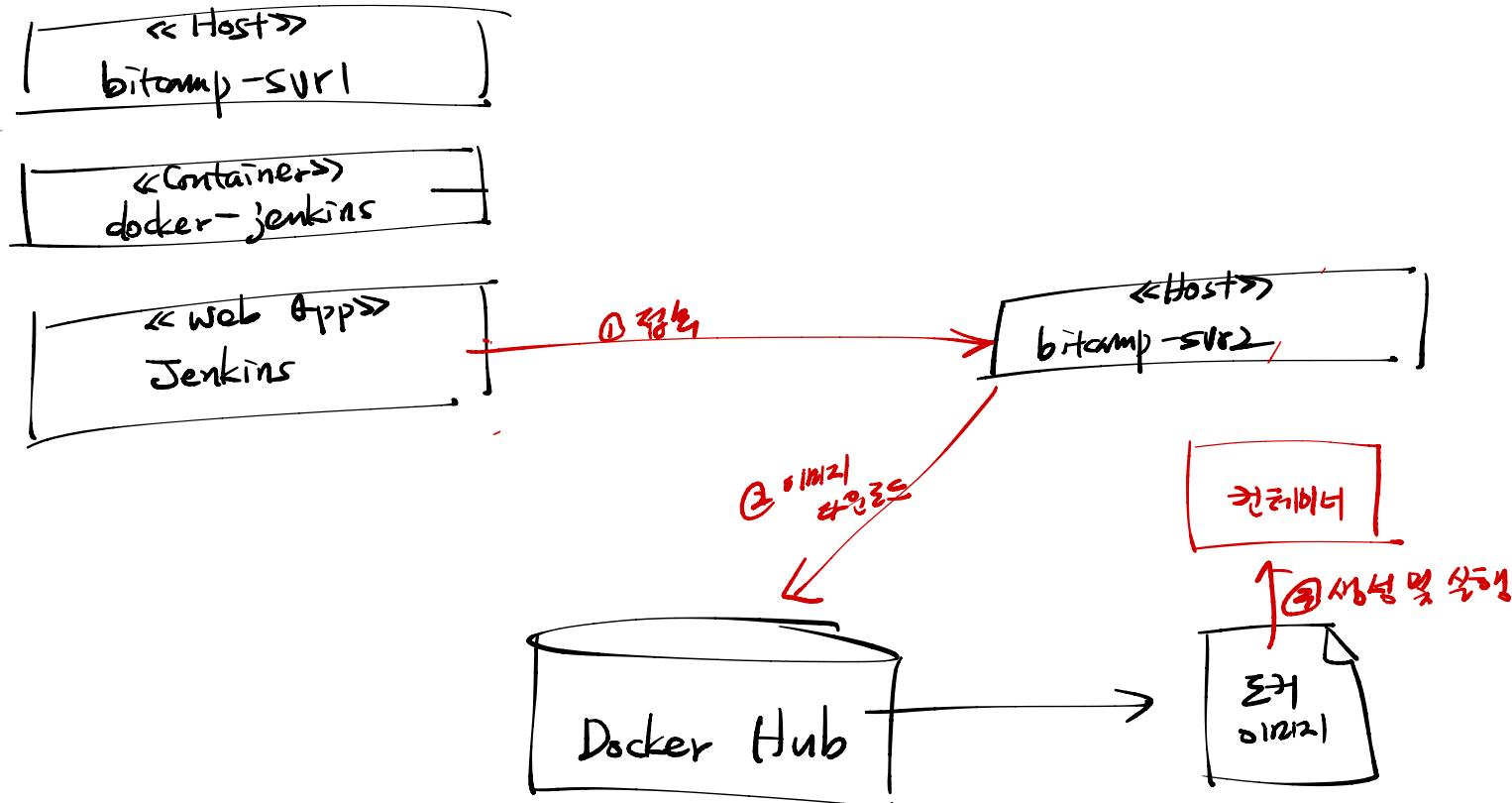
* 자동 배포 - CI/CD
(my/app 이미지 사용)



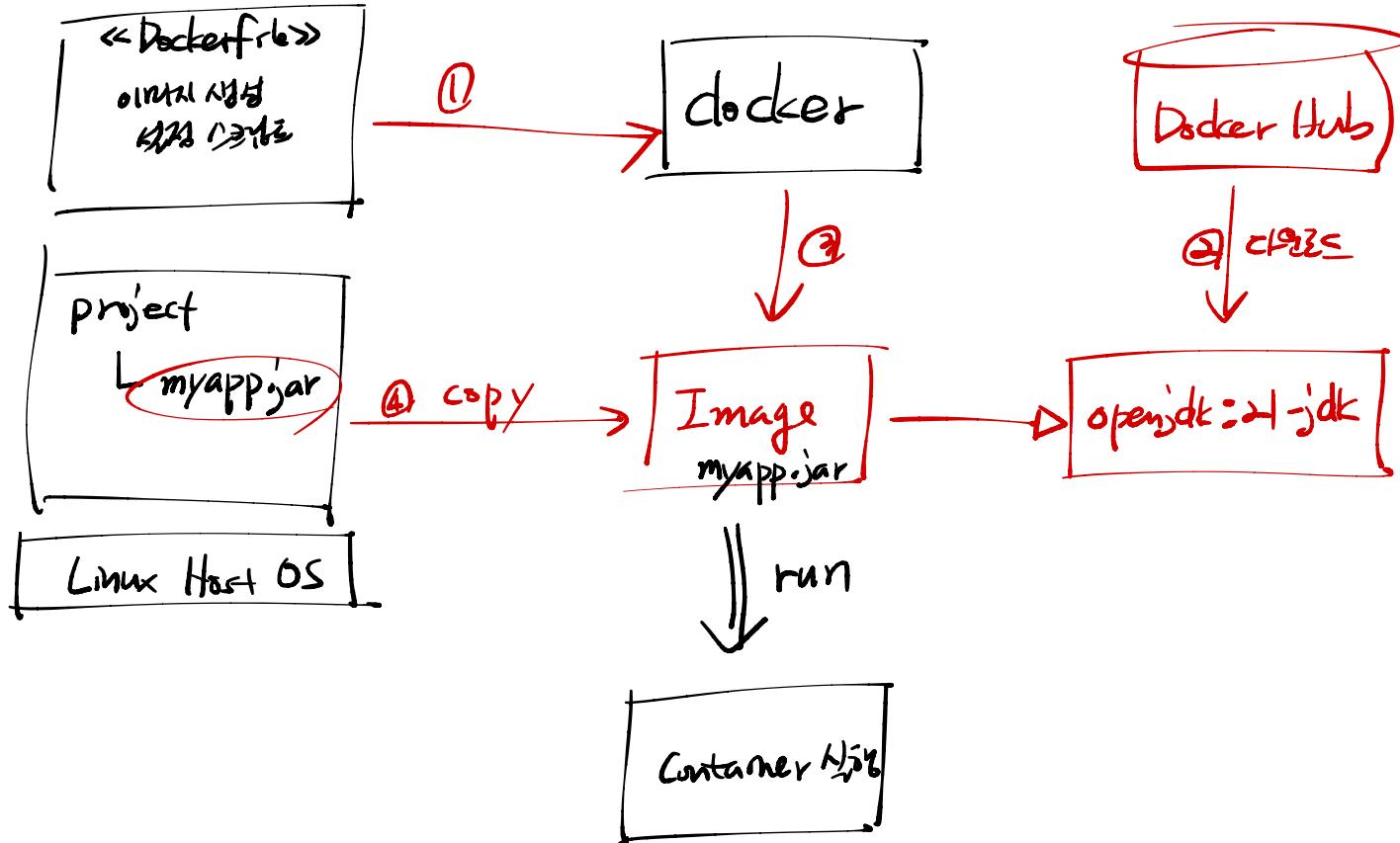
* 자동 배치 - ZOERM (1회차 실습 11.28)



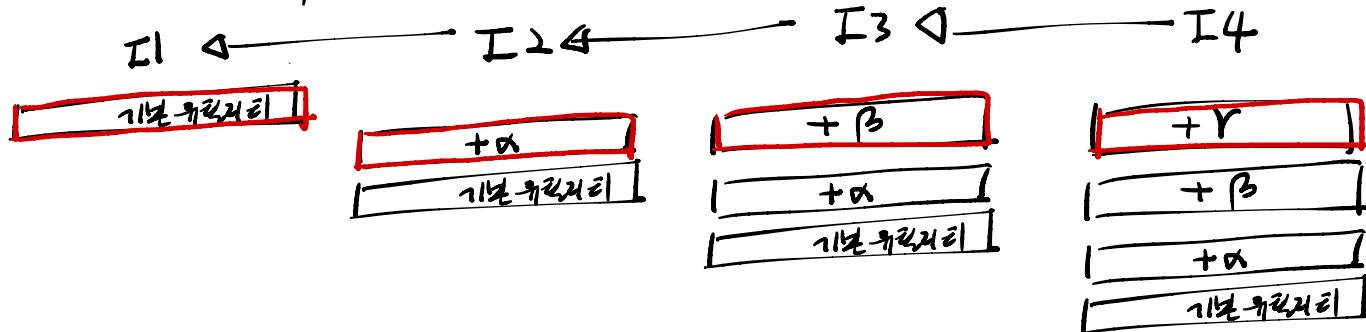
* 자동 배치 - 3단계(1단계)



* Dockerfile သို့ မြင်ဆဲ Container မှုပေါ်



→ Docker 01[12] +?



Layered Image 구조

↓ 커리어저 쌓임

I1 차이

I2 차이

I3 차이

I4 차이



Kiwi