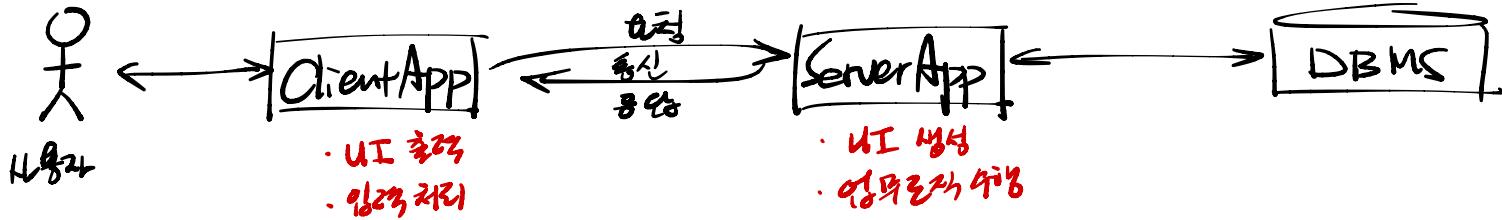
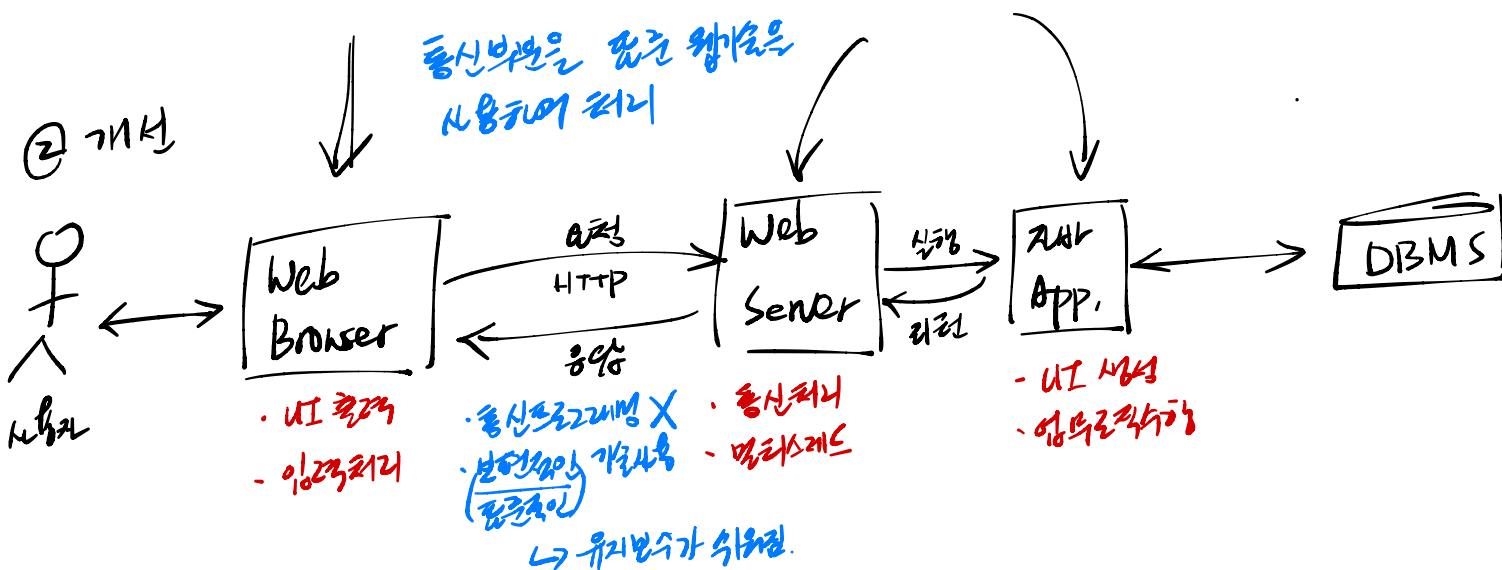


49. 웹애플리케이션 구조를 살펴보자

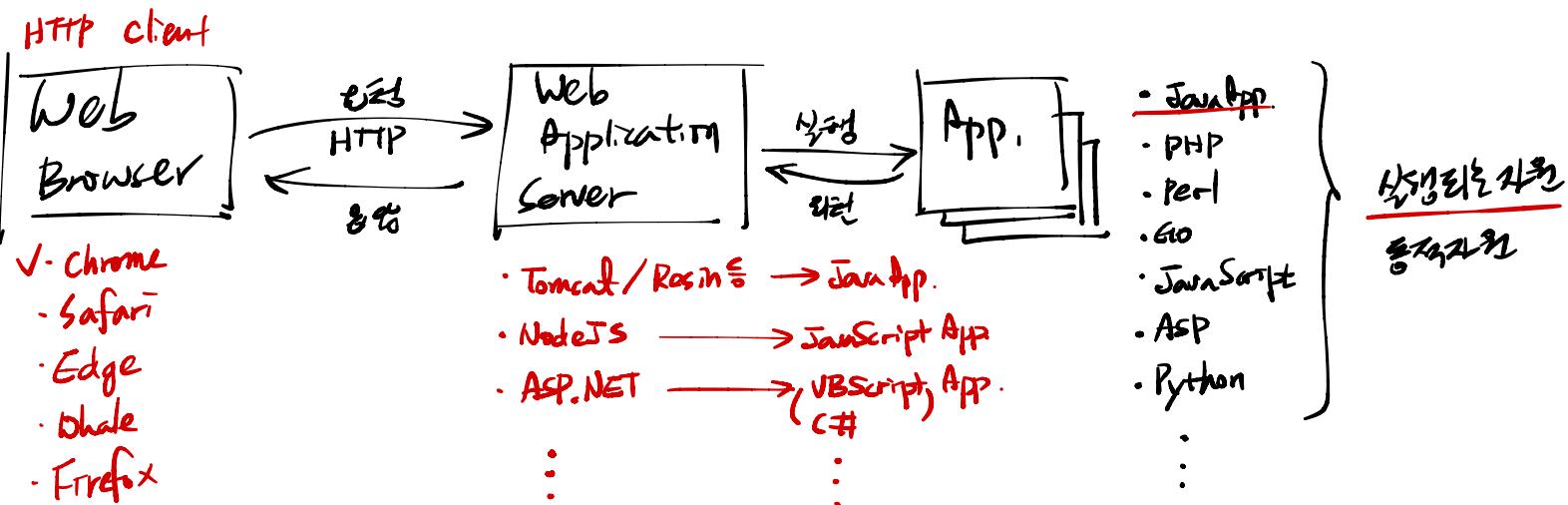
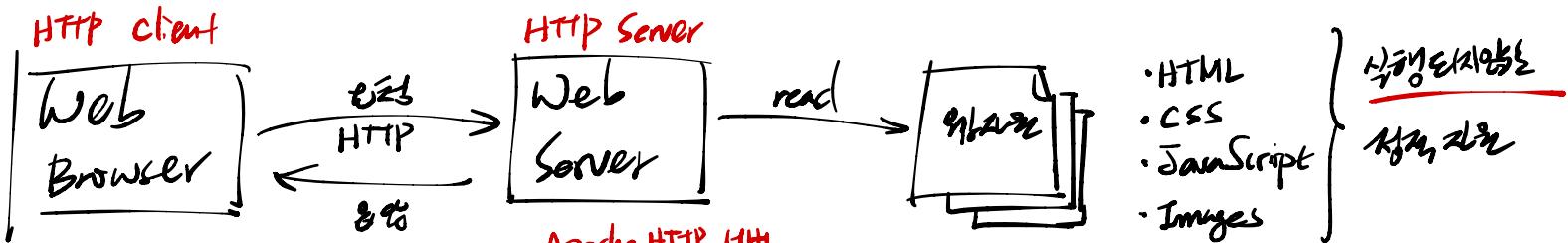
① 단계 → 전용 프로토콜을 사용하여 클라이언트/서버 통신



② 개발



* 웹 사이트의 구조와 작동 원리



* Web App. or CGI 프로그램

① 험장기 유형 애플리케이션 (제시판, 게시판, 쇼핑몰 등)

HTTP Client

- curl
- wget
- chrome
- Firefox

Web Browser

HTTP Server

Web Server

APP

Common Gateway Interface (CGI)
↳ 다른 APP. 활용 가능하게 하는 기술

요청
응답

CGI
결과

실행

C / C++

↑ CGI 프로그램 (CGI 기능이 추가된 풀체스팅)

② 풍선기 유형 애플리케이션 (쇼핑몰 등) → UI 브라우저 UI → 다양한 HTML 결과 → 문자열화하여 응답

Web Browser

HTTP Server

스크립트
언어처리기

스크립트
APP.

요청
응답

CGI
결과

.pl (PERL)
.php (PHP)
.asp (ASP)

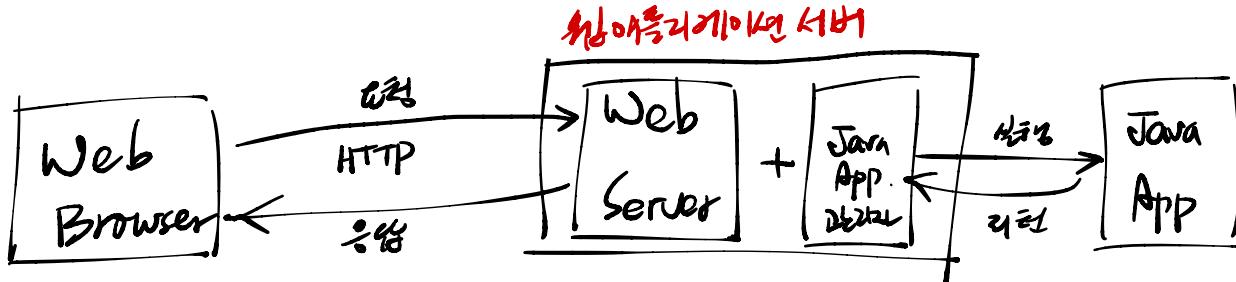
③ 흐름기 유형 애플리케이션 (인수/값 처리, 하위 시스템 등 입력/생성)

↳ 유형 애플리케이션 → OOP 프로그래밍 기법 도입
구현하기 쉽다

{ Java 언어
.NET
= }

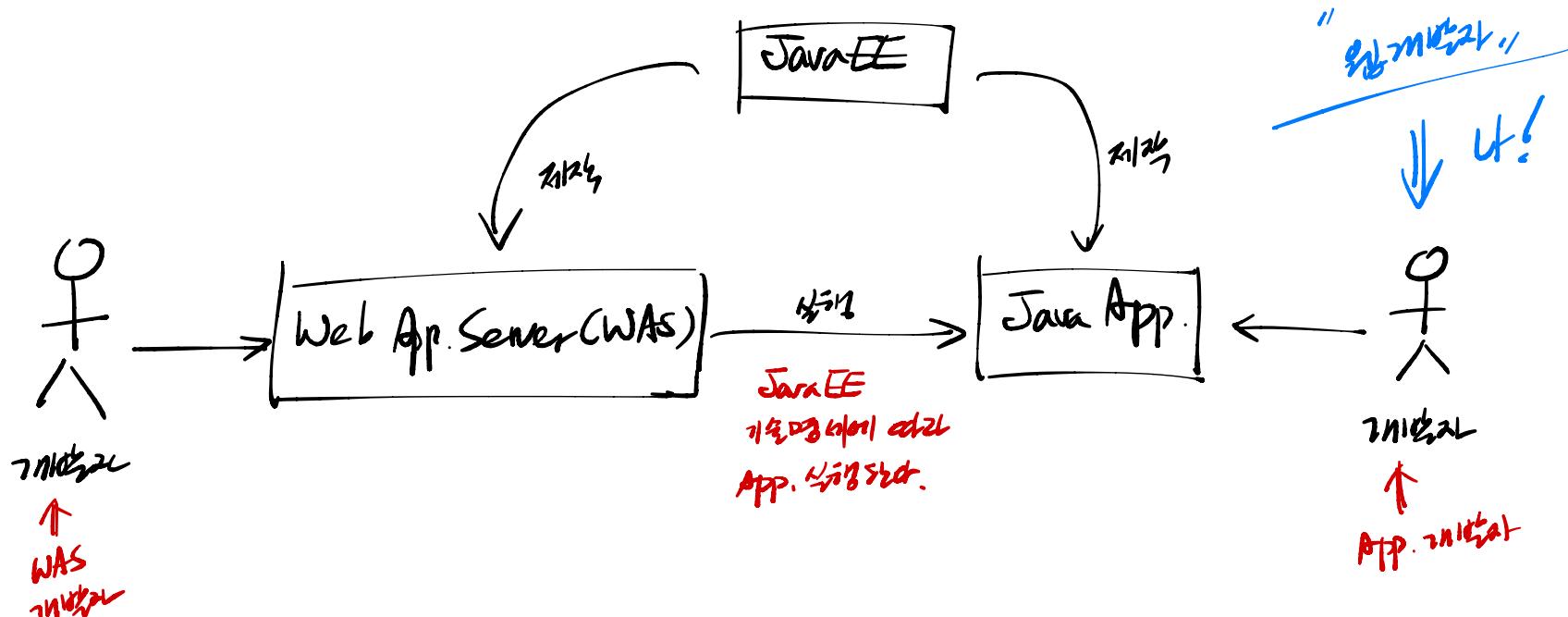
↑ C/C++ 보다 모듈화
단위별 파일,
시스템별 분리
설정이 어렵다
↳ 접근성이 X

* Java Web Application Architecture



- Tomcat / Resin / Jetty : ~~轻量级~~
 - Weblogic
 - Websphere
 - JBoss
 - BEA
 - IBM
 - JEUS
 - :

* Java Web Application ut JavaEE چیぞ咯!



* Java EE (Enterprise Edition) 기술 영역

↳ 기업용 App. 개발 기술 모음

↳ 데이터베이스, 공유자원 관리, 분산처리, 접근제한 등

애플리케이션

Web



Servlet/JSP

JSTL > EL

ESB

Web Service

Authentication
&
Deployment

- 웹 애플리케이션 개발 기술

← 핵심 기술!

⇒ REST API

- 분산 처리 기술

- 서비스 기반으로 서비스 개발

- 인증 및 배포 기술

* Java EE 7/8/10/11 버전의 특징

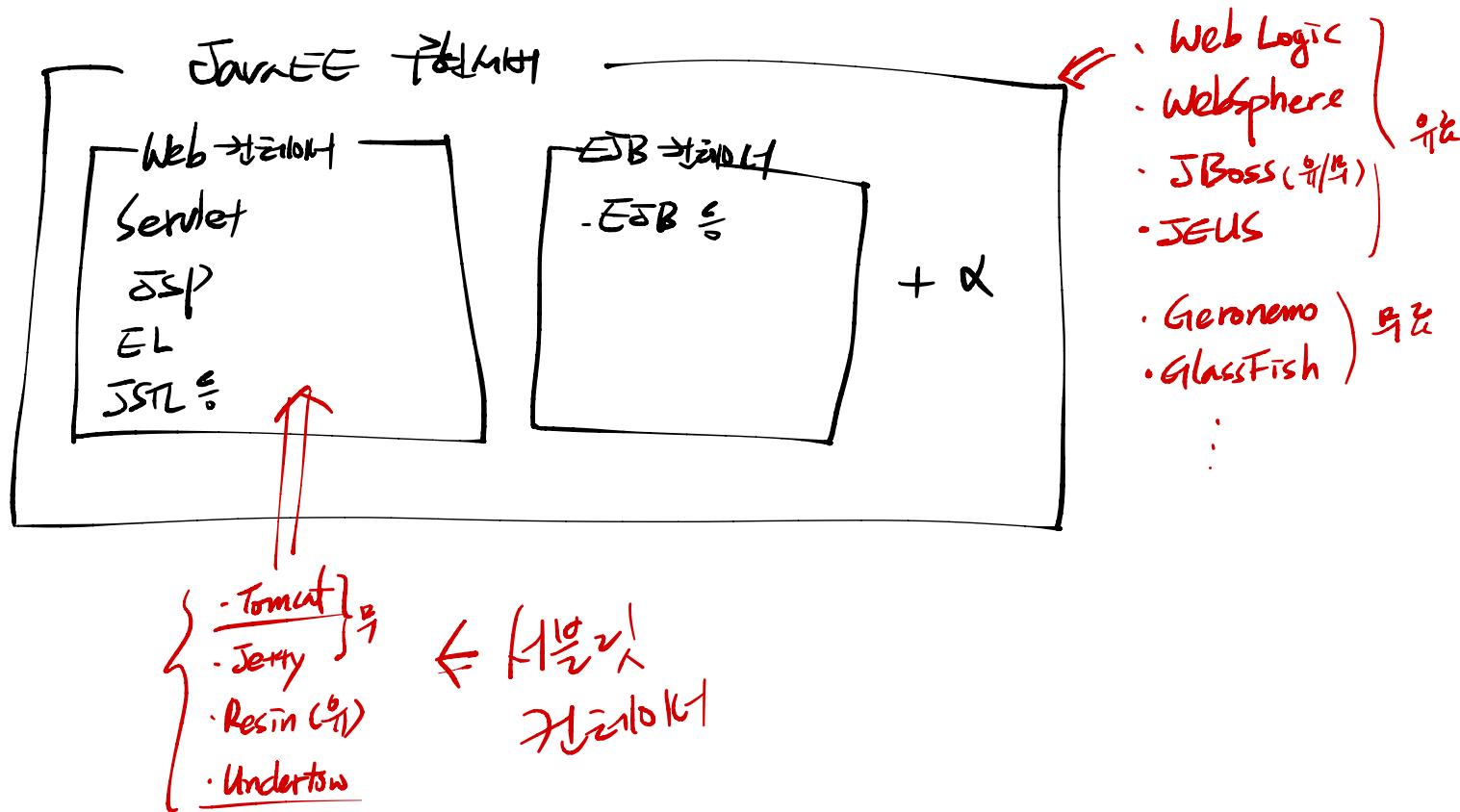
JavaEE 버전 제작년도	Servlet	JSP	EL	EB
5	2.5	2.1		3.0
6	3.0	2.2	2.2	3.1
7	3.1	2.3	3.0	3.2
8	4.0	2.3	3.0	3.2

* Java EE 1/2 버전부터 ESSENTIAL (Implements)

JavaEE 1/2	Web Logic	Web Sphere	JBoss	Tomcat
5	10.3	6.1, 7.0	6.0	6.0
6	11g, 12c(12.1.x)	7.0, 8.0	7.0	7.0
7	12c(12.1.3), 12.2.x	8.5, 9.0	8.0	8.0
8	12.2.1, 14.1.x	9.0.x	9.1, 8.2	8.5, 9.x

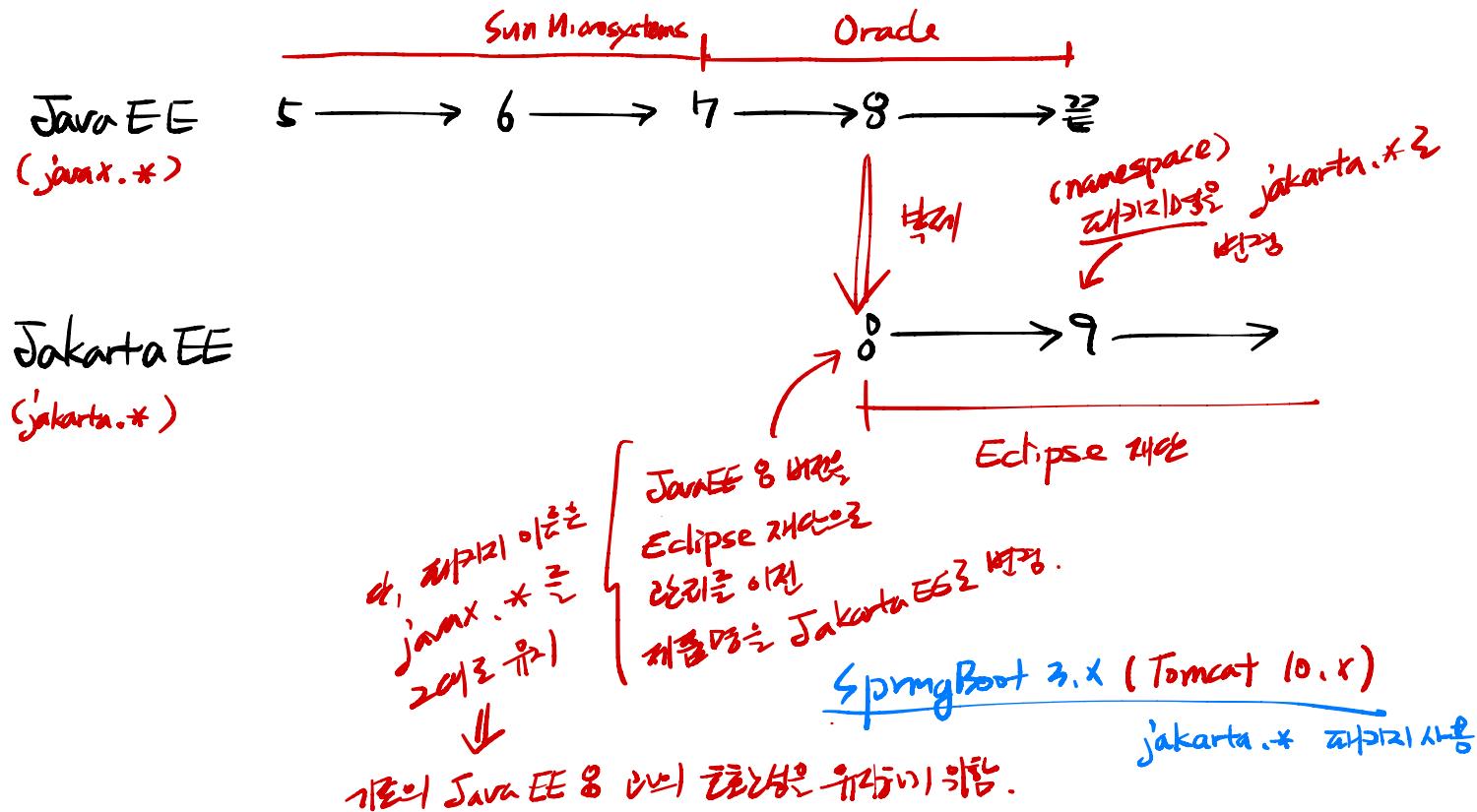
⇒ 2014년 10월 버전이 최신
2014년 Java EE 버전으로 업그레이드 할 것.

* JavaEE 편집기 사용하기

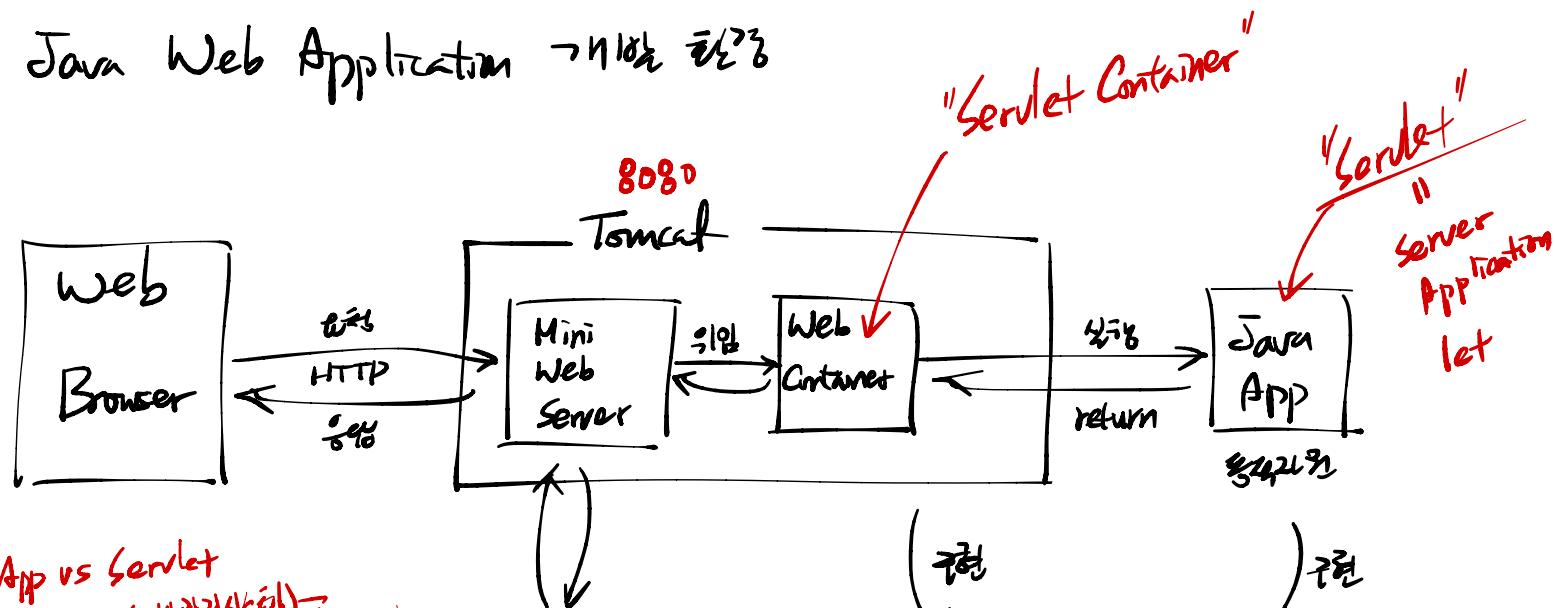


* JavaEE & JakartaEE

SpringBoot 2.x.x (Tomcat 9.x)

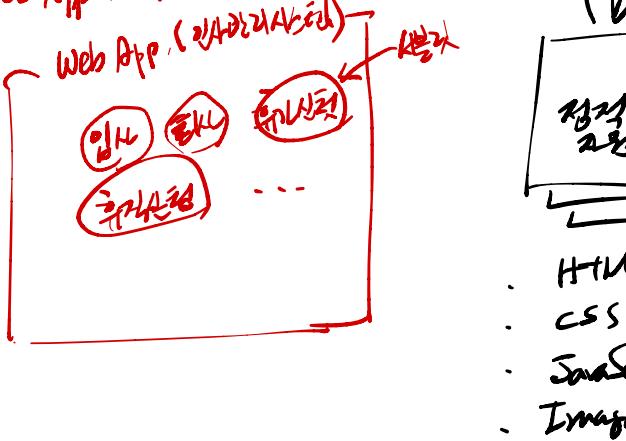


* Java Web Application → چیزی که



* Web App vs Servlet

Web API (HTML API) → REST API



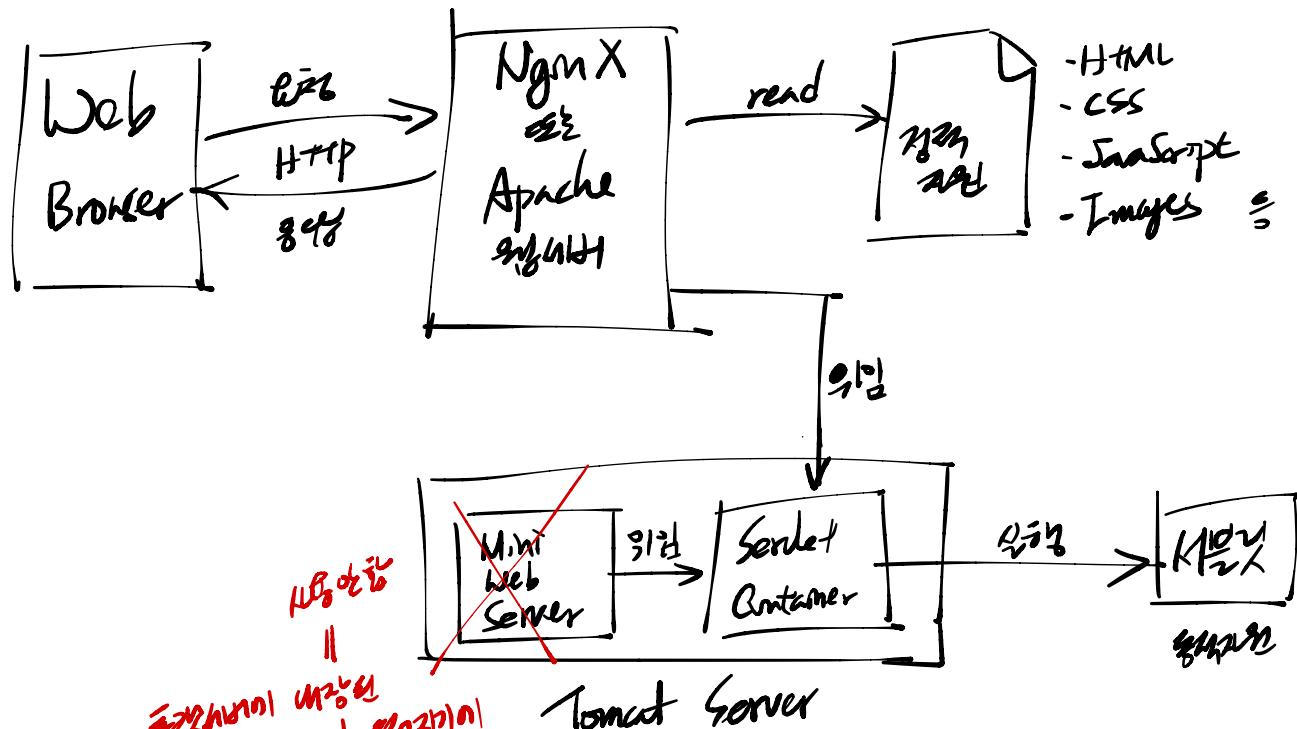
- HTML
- CSS
- JavaScript
- Images

JavaEE

~| MVC モデル |~

- Servlet / JSP -

* Java Web Application 운영 원리



External module
application layer
middle layer
presentation layer
business logic
data layer.

* Tomcat Server

CATALINA_HOME/

- bin/ ← 허버 실행과 관련된 파일
- conf/ ← 허버 설정 파일
- lib/ ← 허버 라이브러리 파일
- logs/ ← 허버 실행 로그 파일
- temp/ ← 허버 실행 중에 사용되는 파일을 두는 폴더
- webapps/ ← 웹 어플리케이션 폴더를 두다.
- work/ ← JSP를 Servlet 처리하기로 변환한 파일을 두는 폴더

启动文件 => startup.bat (Windows)
startup.sh (Mac, Linux)

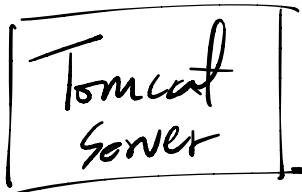
停止文件 => shutdown.bat (Windows)
shutdown.sh (Mac, Linux)

* Tomcat 끝내는 것

① 시작하기



startup.bat



Development (개발)

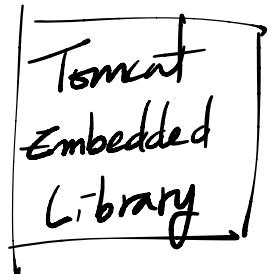
↓ .war

Operation (운영)

② 구현하기



call



운영

Spring Boot

Development + Operation = DevOps

* Embedded Tomcat 끝까지 써보기

new Tomcat()

- port = 8888
- baseDir = temp
- URLEncoder = UTF-8

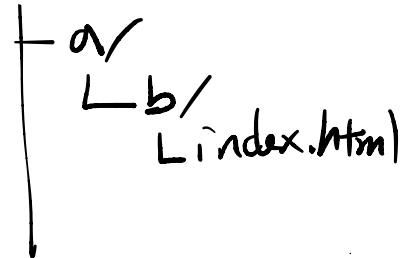
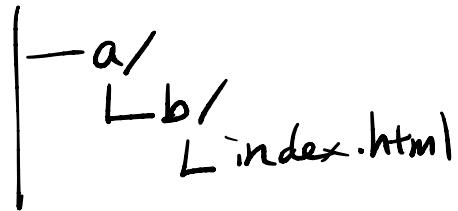
- 웹 어플리케이션 경로 \Rightarrow /
 - 정적 자원 경로 \rightarrow src/main/webapp/ \leftarrow .html, .css, .js, .gif 등
 - 동적 자원 경로 \rightarrow bin/main \leftarrow .class, .properties, .xml 등

웹 자원 주소: http://localhost:8888 ↗

- ① 정적 파일 경로인 /index.html 을 찾는다.
- ② 동적 파일 경로인 /index.html 3 번째가 실행된 내용을 찾는다.

* აბსოლუტური

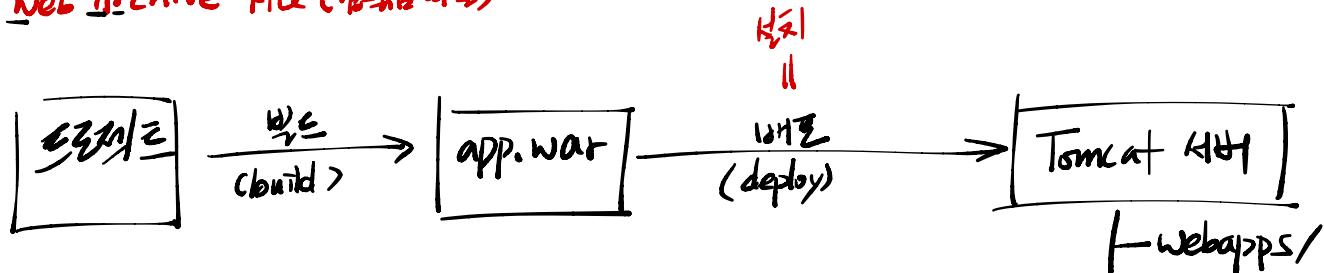
http://localhost:8888/ → src/main/webapp/



* 웹 어플리케이션 프로젝트 배포

① .war 파일 배포

Web Archive File (웹아카이브)



② 코드 + API 라이브러리
(라이브러리)
배포 = springboot 앤드



* Maven 파일은 쉽게 편리한 확장자를 가짐

구조도



.html, .css, .js, .gif 등 이미지 파일

WEB-INF

web.xml ← 딜레이션 데스크립터 (Deployment Descriptor File; DD File)

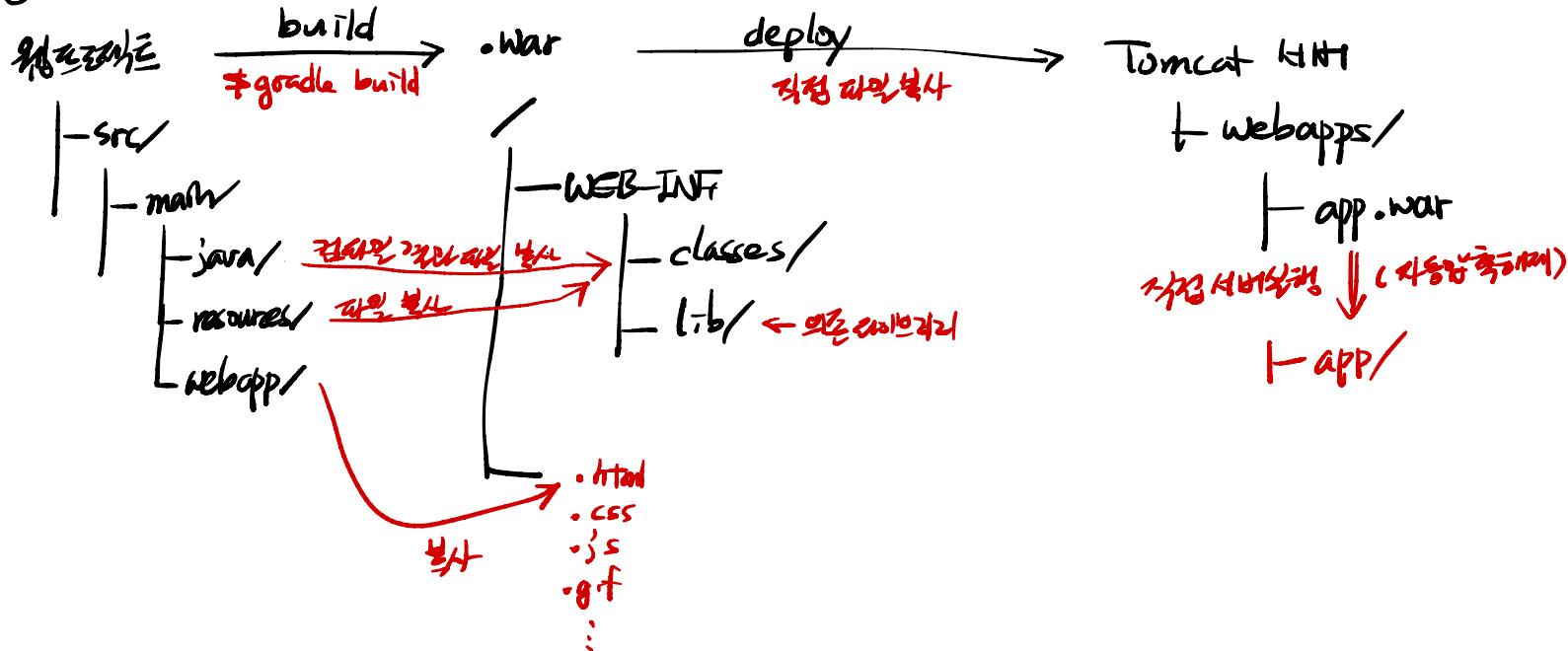
classes/ ← .class, .properties, .xml

lib/ ← .jar

.xml, .properties 등 설정 파일

* 웹프로젝트 배치

① 죽기보다 서둘러 미치는 ← 유품에 둘러싸여 있던 가방을 발견한 후에 배치하는 경우



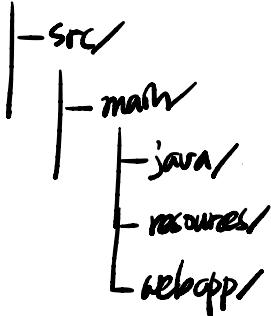
* 웹프로젝트 파일

② Eclipse 웹 파일 <서버 설정을 적용하는 동안 서버를 파일에

생성되는

Eclipse 임시 파일 폴더

이경스 위치:/.metadata/.plugins/org.eclipse.wst.server.core/



tmp0/

- conf/ ← tomcat 설정 /xml, 폴더는 복사됨

- logs/ ← 실행 기록 파일

- temp/ ← 일정 주기로 자동으로 삭제되는 폴더

- webapps/ ← 새롭게 만든

- work/ ← JSP 변환 파일 생성되는 폴더

- wtpwebapps/ ← 웹프로젝트 파일 폴더

생성되는/

- WEB-INF
 |
 + classes/
 |
 + lib/

생성되는

* 웹프로젝트 ID IntelliJ

② 웹프로젝트 + WAS 라이브리 <= SpringBoot 환경



* 풀不尽 서버로 진화하여 등장

① 웹서버 → ② 애플리케이션 서버

Server App

- 파일통신구현
- 서비스레이팅 구현
- 전송통신망
- 웹서버 구현

Server App

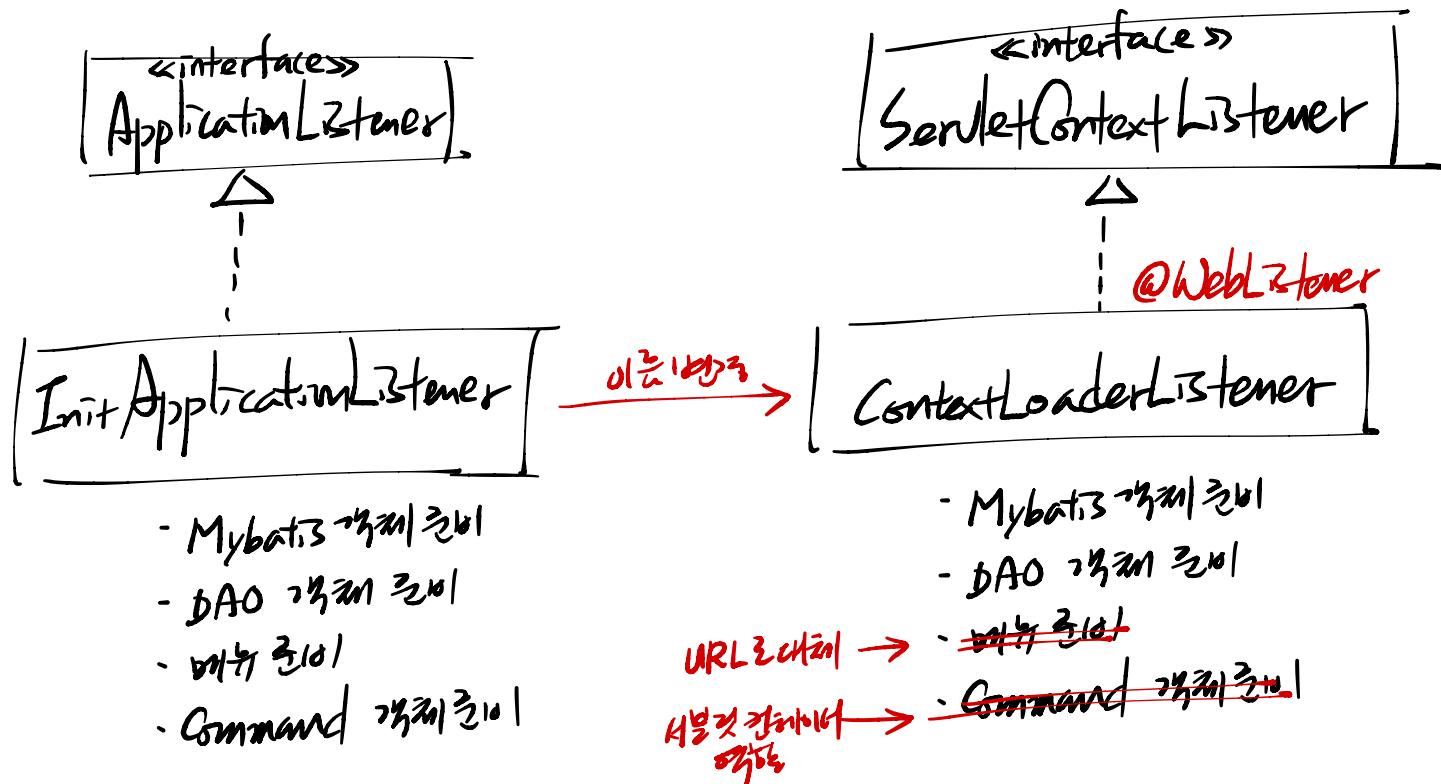
- ~~파일통신구현~~
- ~~서비스레이팅 구현~~
- ~~전송통신망~~
- ~~웹서버 구현~~

Web

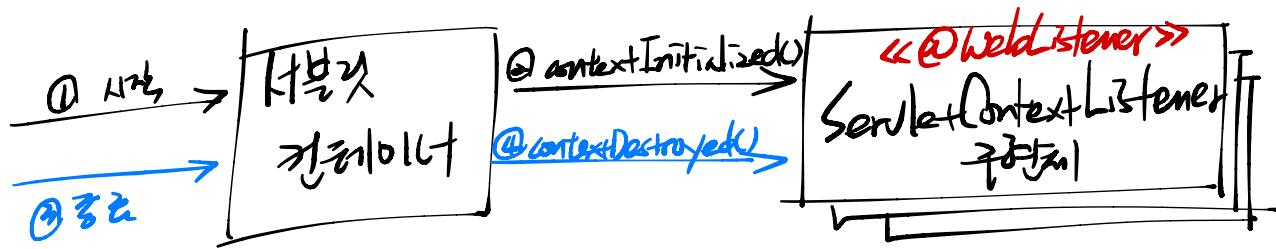
Tomcat

- HTTP 프로토콜을 처리하는 웹브라우저 구현
- 서비스레이팅 구현
- 전송통신망 구현
- 웹서버 기능으로 구현
- 웹사이트의 URL을 기능 구현

* 리스너 만들기



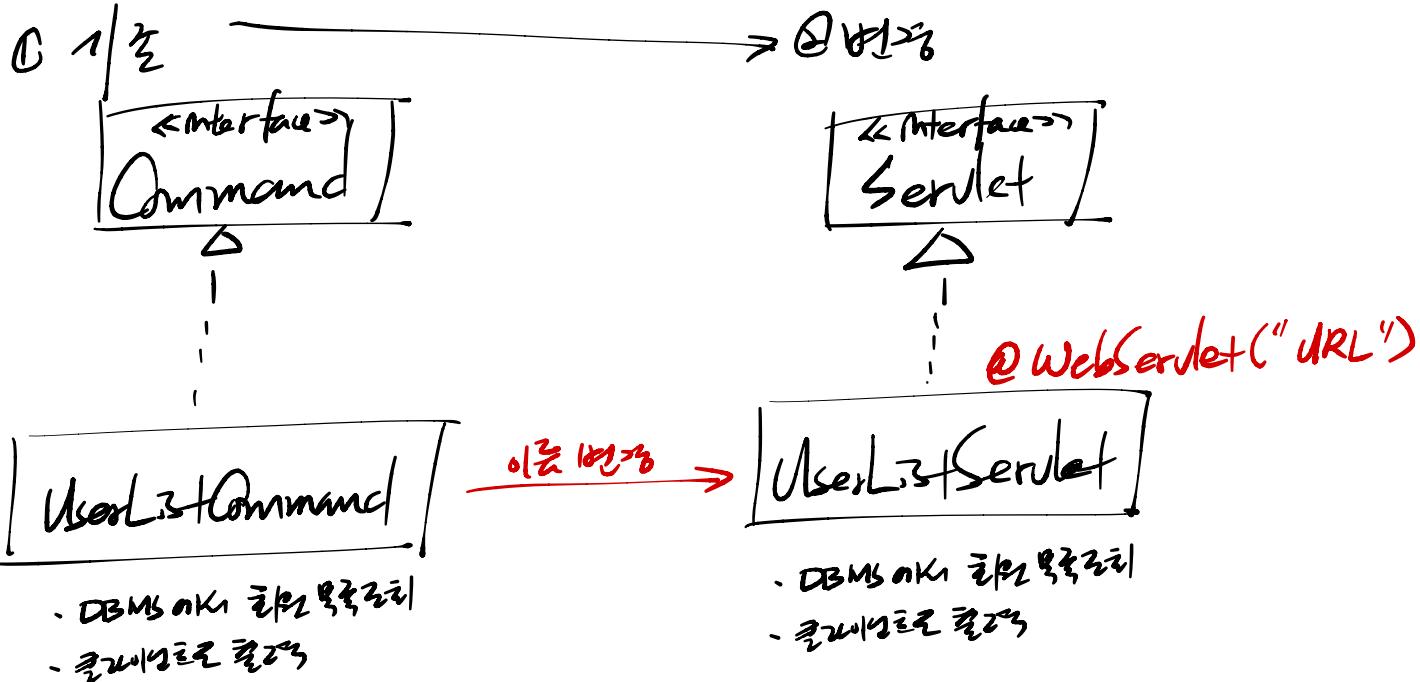
* ServletContext Listener



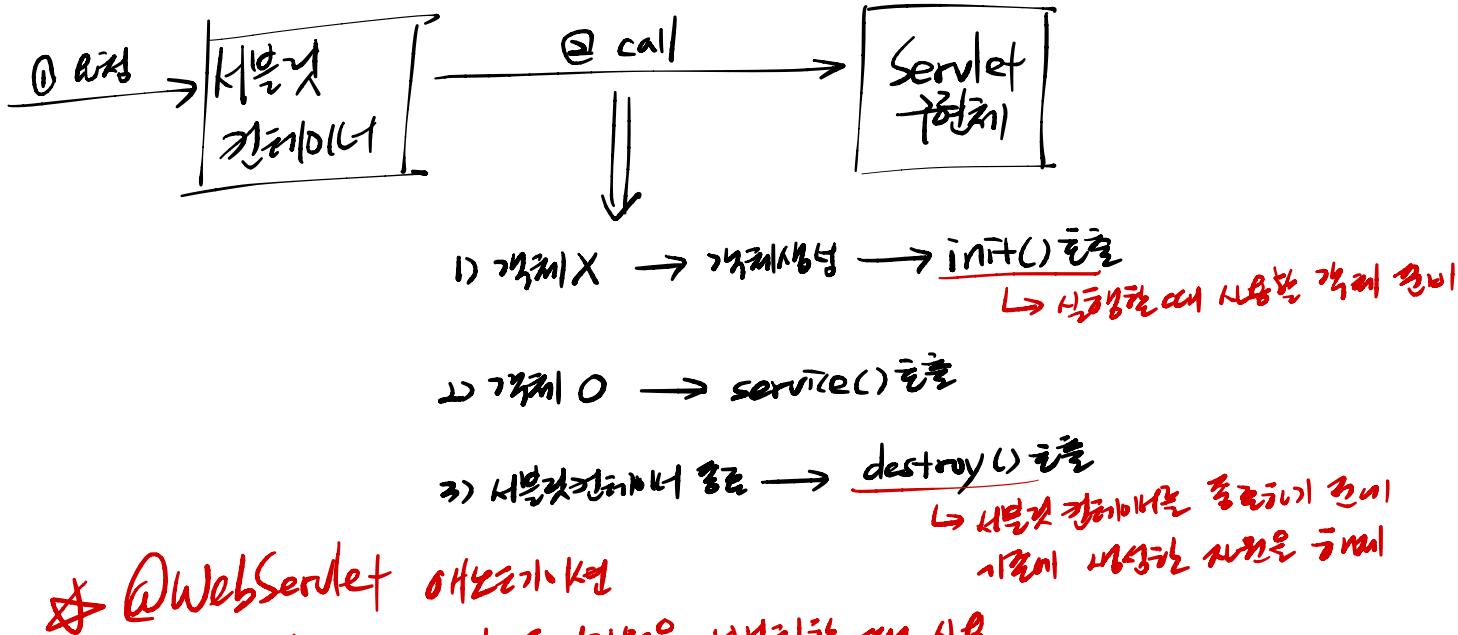
* **@WebListener** effect하기

- 라이브러리에 수동적 필터를 넣기
수동적 필터는 서버가 요청을 처리하는 과정에서 자동으로 처리되는 특성

* 허용 가능한 방법



* Servlet 구조



* WebServlet의 특징

- 서블릿작동이 끝나면 종료되는 대체로 특징은 유지된다.
- 클라이언트에게 응답을 때 그 내용을 처리하는 코드를 작성한다.
- URL을 이용하여 코드를 처리하는 처리를 적용한다.

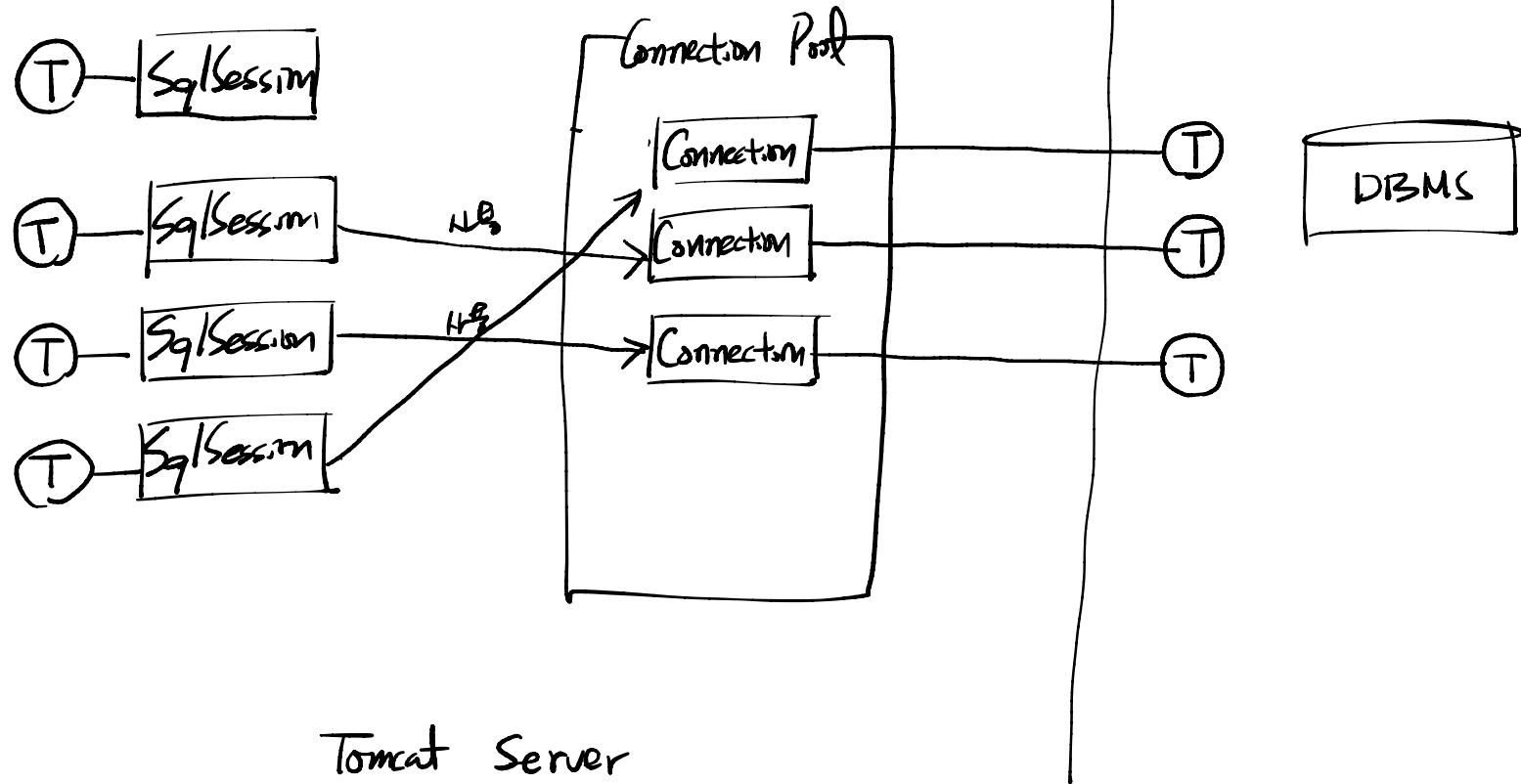
* HTML 문서에서 다른 자원을 요청하는 경로 주소

<u href="/user/view?no=11"> user11 </u>

HTML reference

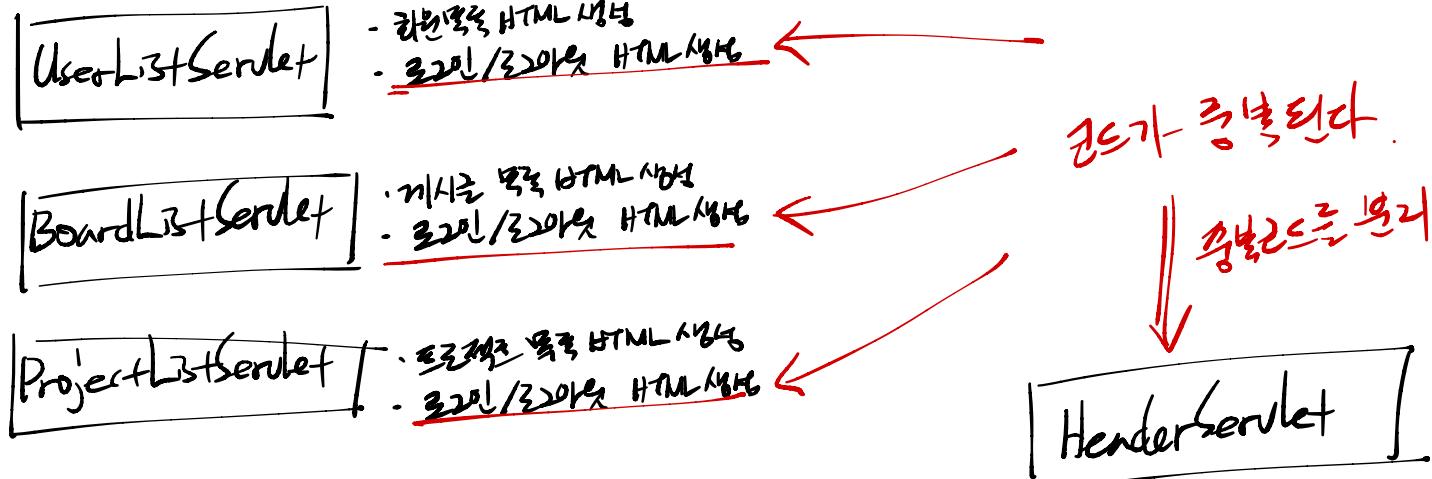
→ 템플릿 엔진
→ 컨트롤러
→ 서비스
→ 데이터베이스
→ 응답

* DBMS et SqlSession



* 서블릿 include → 다른 서블릿의 실행 결과를 포함시키는 방법

① 예제



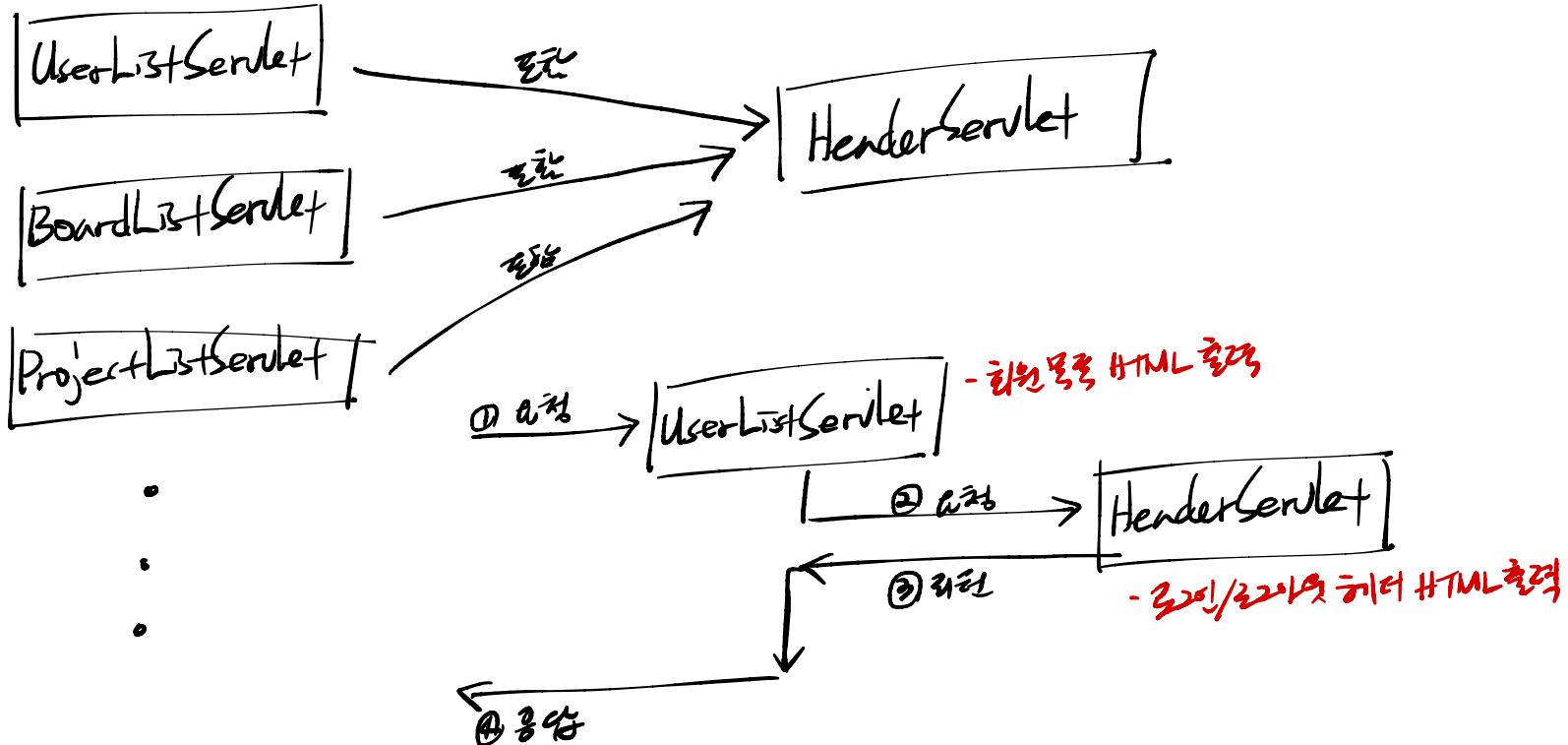
•

•

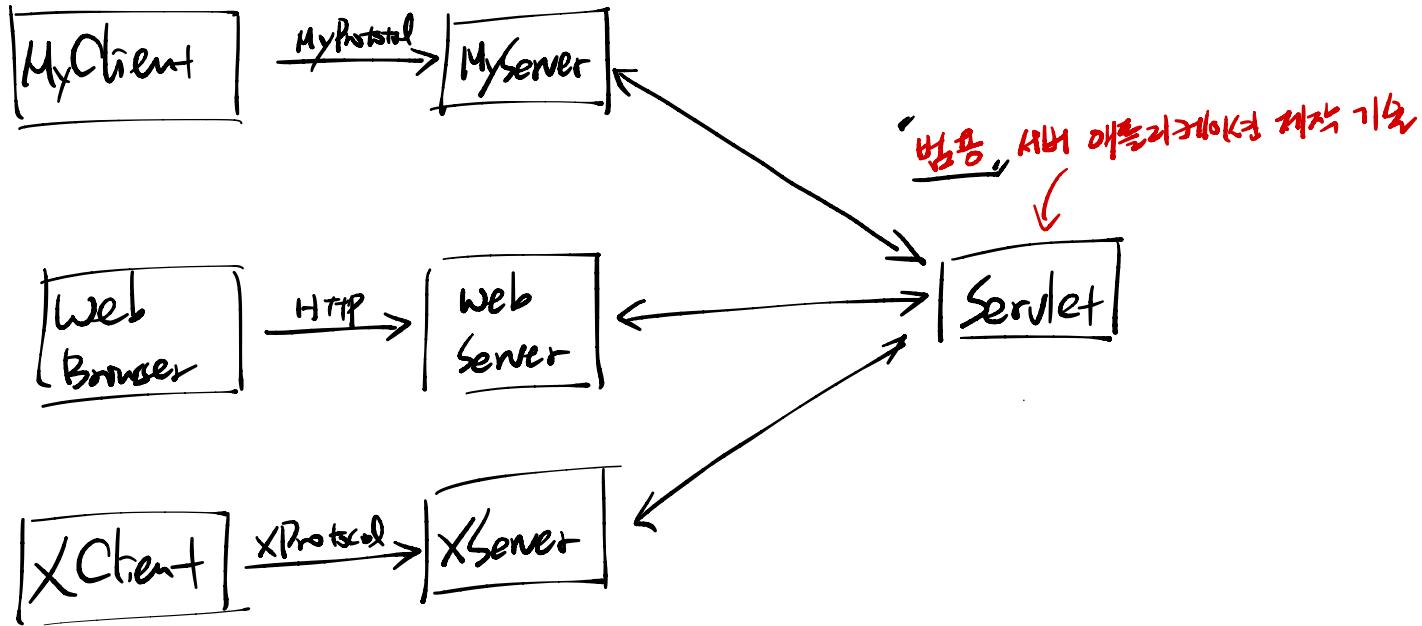
•

* 서블릿 include → 다른 서블릿의 실행 결과를 포함시키는 방법

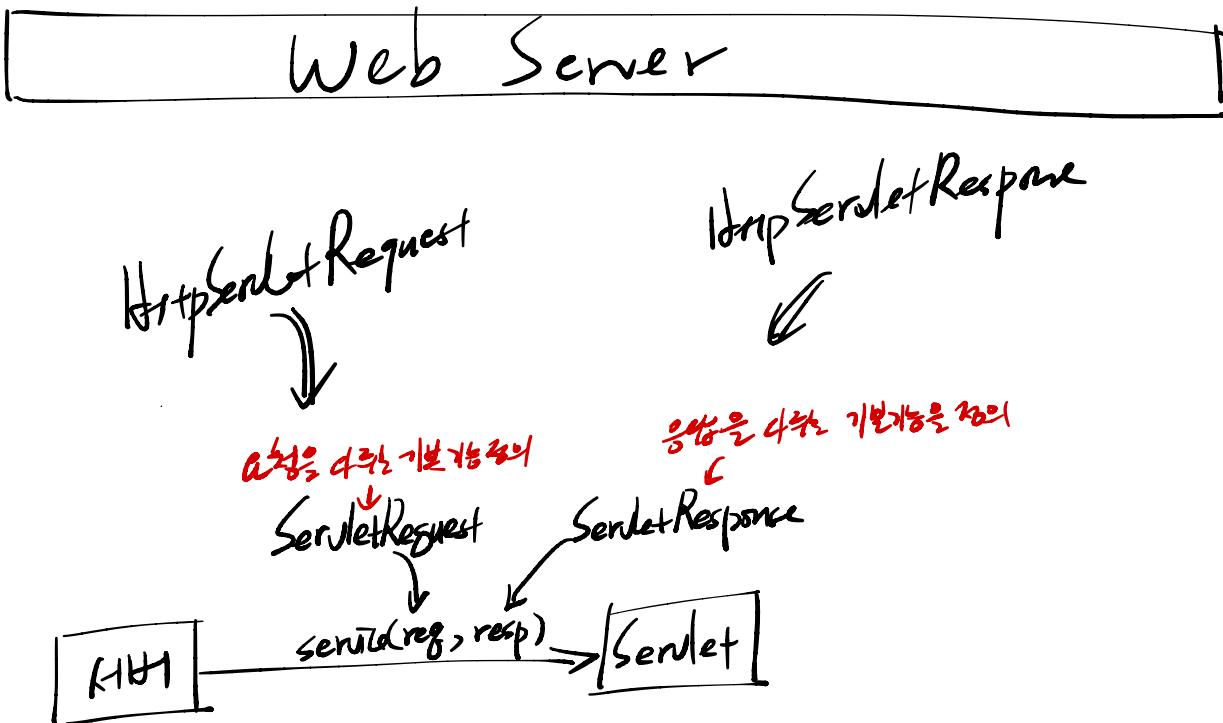
② 구조



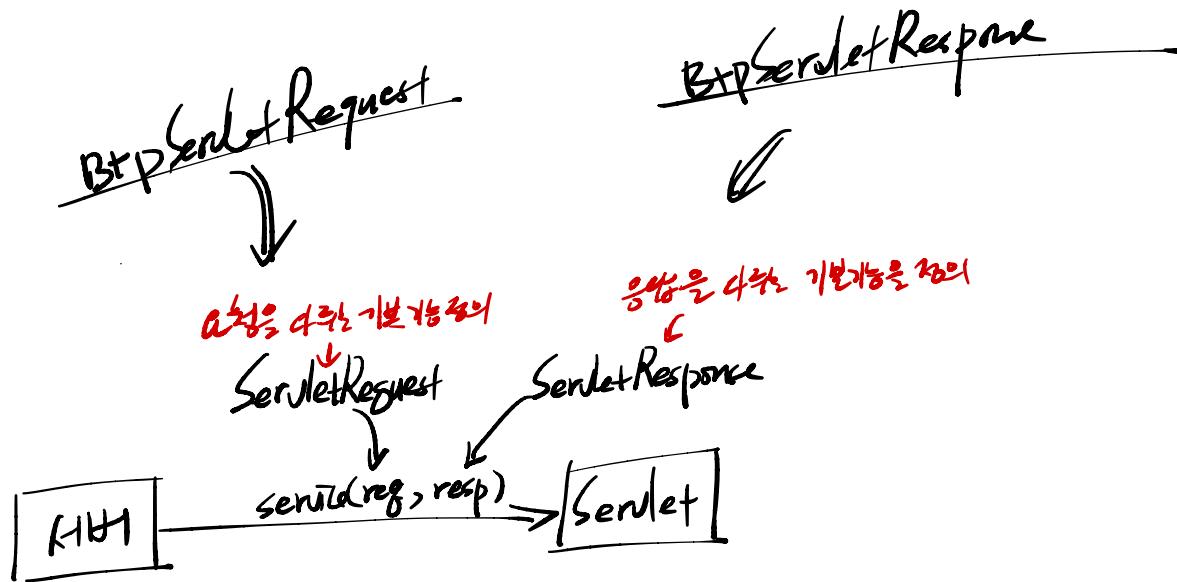
* Servlet چیزی چی



* ServletRequest et ServletResponse

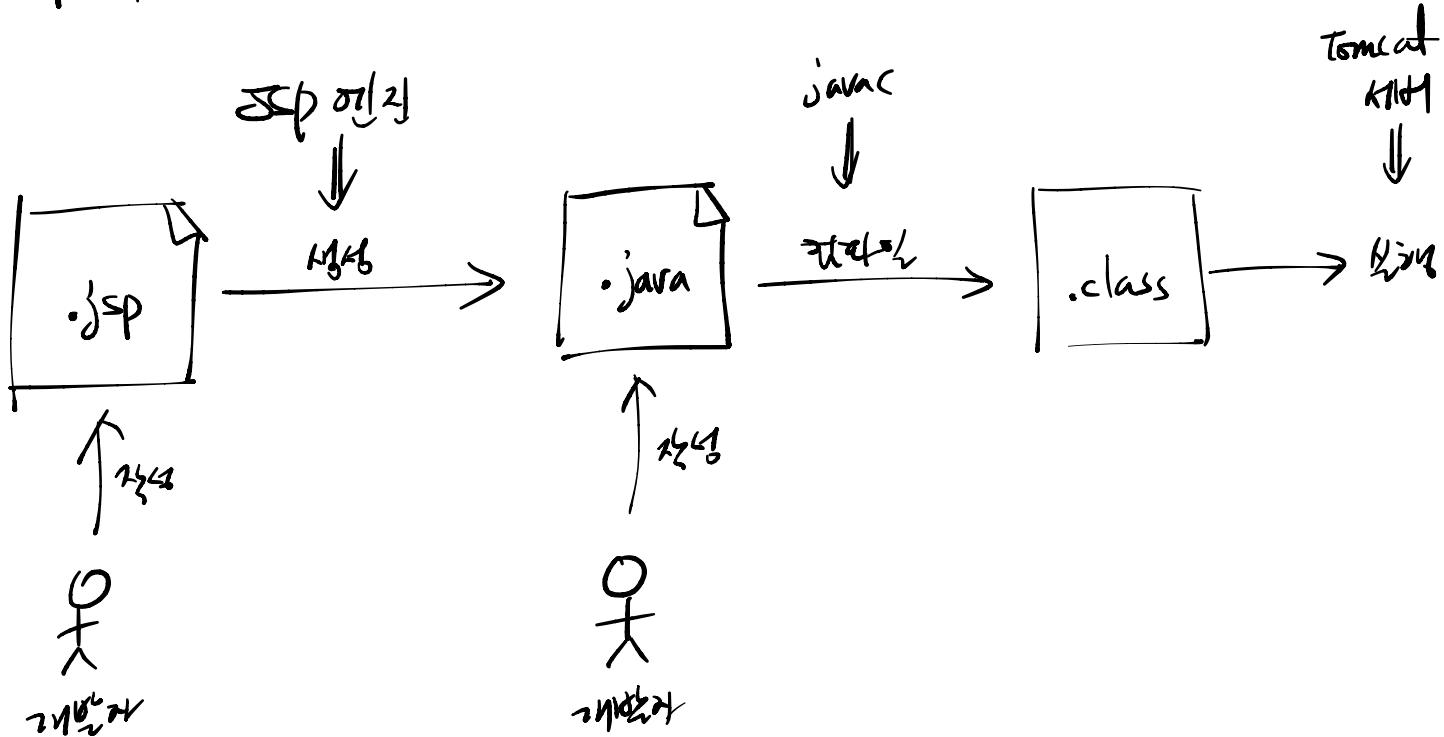


Bitcamp Server

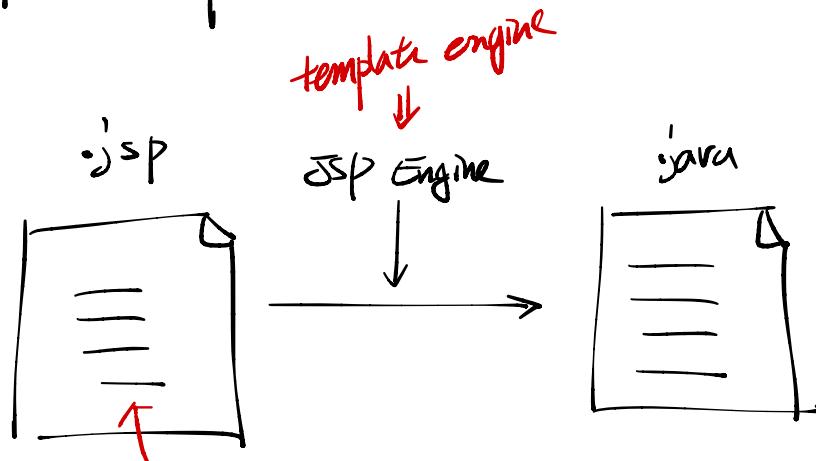


50. UI 흐름을 자동 생성하기 — JSP (Java Server Page)

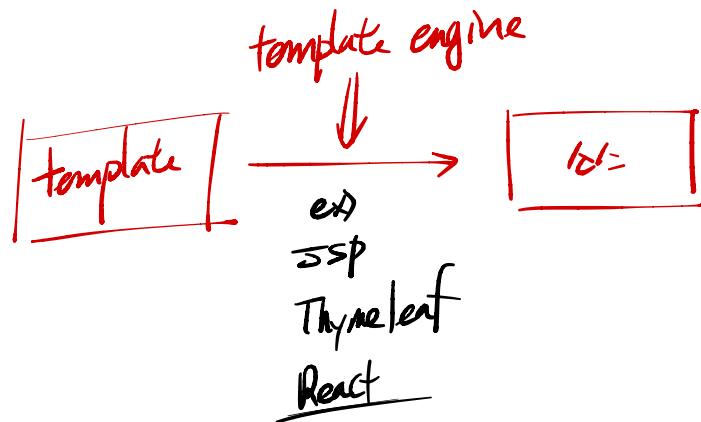
* JSP 구동 원리



* JSP - Template 엔진

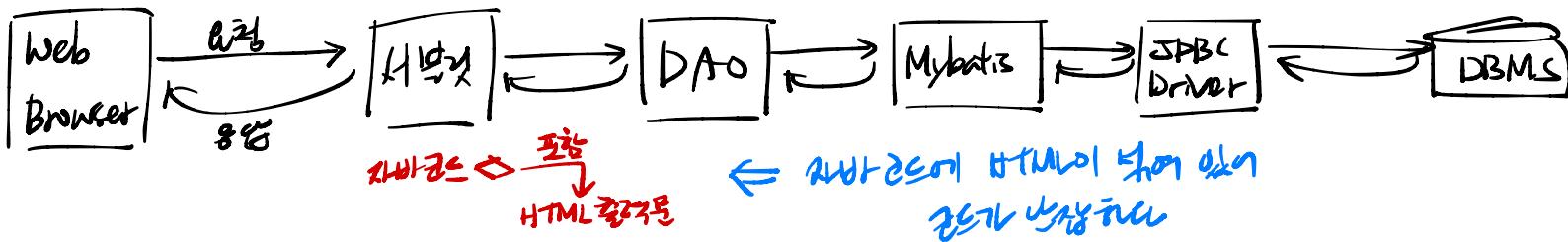


자바 코드를 만들고자 하는 것
자바 코드를 만들자
자바 코드를 만들자
"자바 코드를 만들자"
"자바 코드를 만들자"

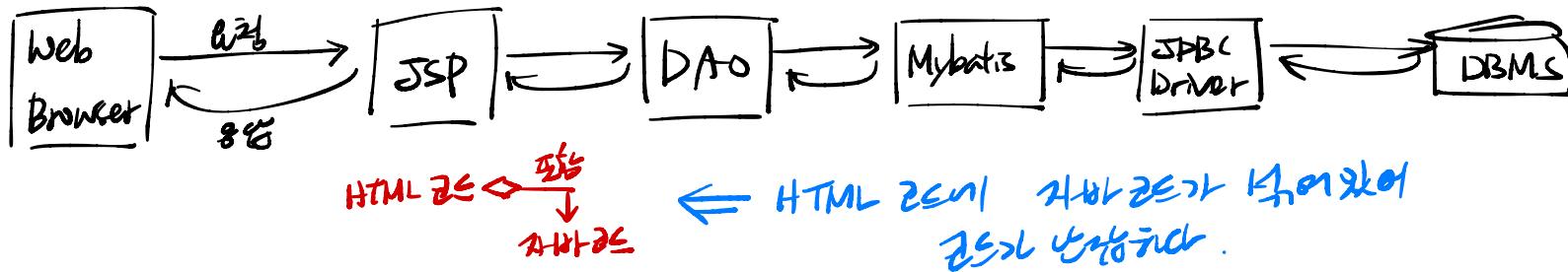


* Servlet 와 JSP 애플리케이션 아키텍처

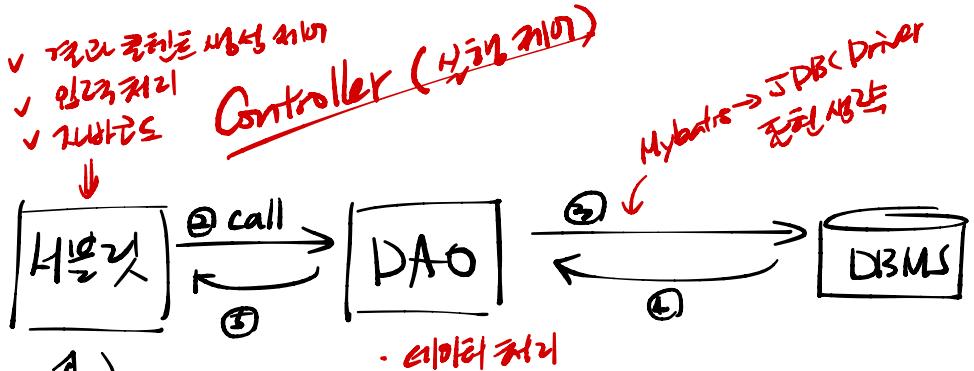
① 서블릿 애플리케이션 아키텍처 - 이전부분



② JSP 애플리케이션 아키텍처 = MVC | 확장



* MVC 2 아키텍처



View (UI 모듈)

(다양한 화면에 출력되는 기능)

UI 모듈
내용을 처리하는 기능

UI 편집자.

JSP

⑩ HTML 내용을 출력하는

UI 편집

View (UI 편집 모듈)

Mybatis → JDBC Driver
정적 접근

Controller (수행 모듈)

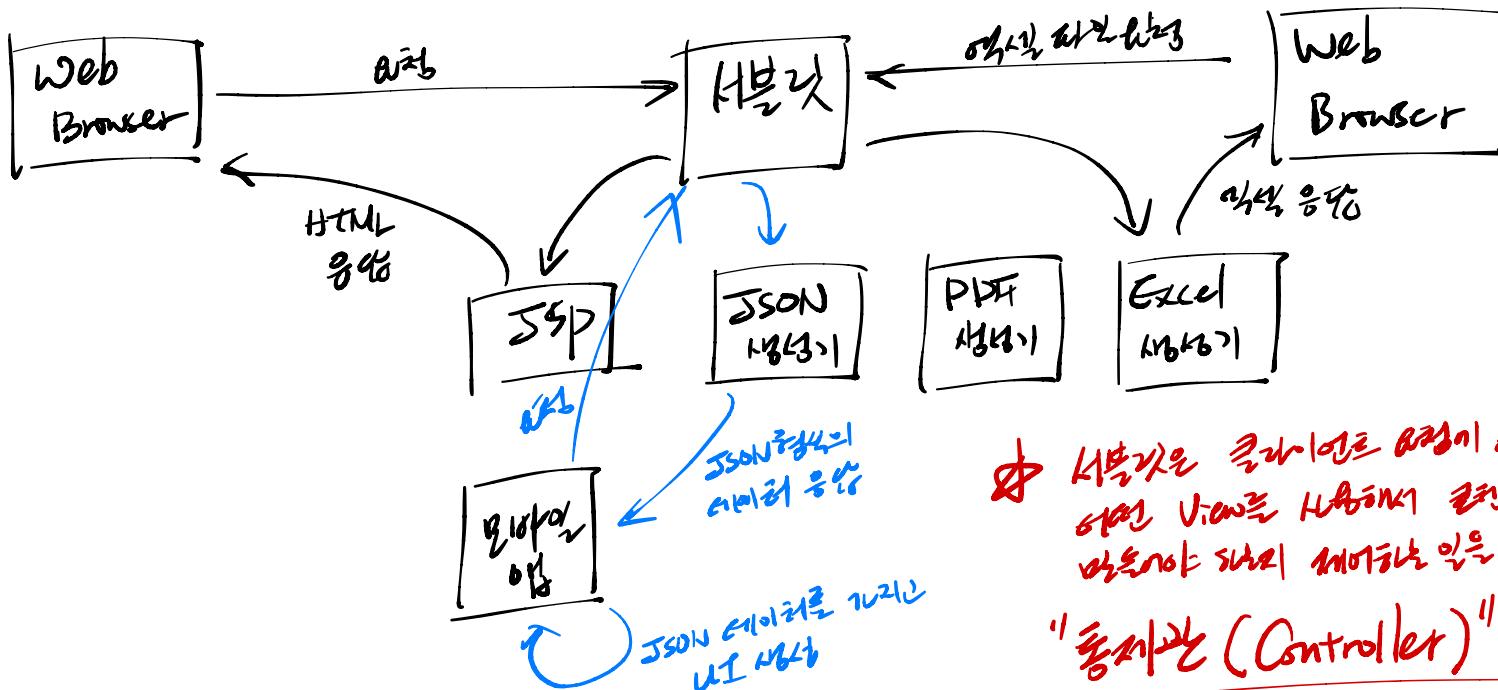
• 데이터 처리

Model (DB 접근 모듈)

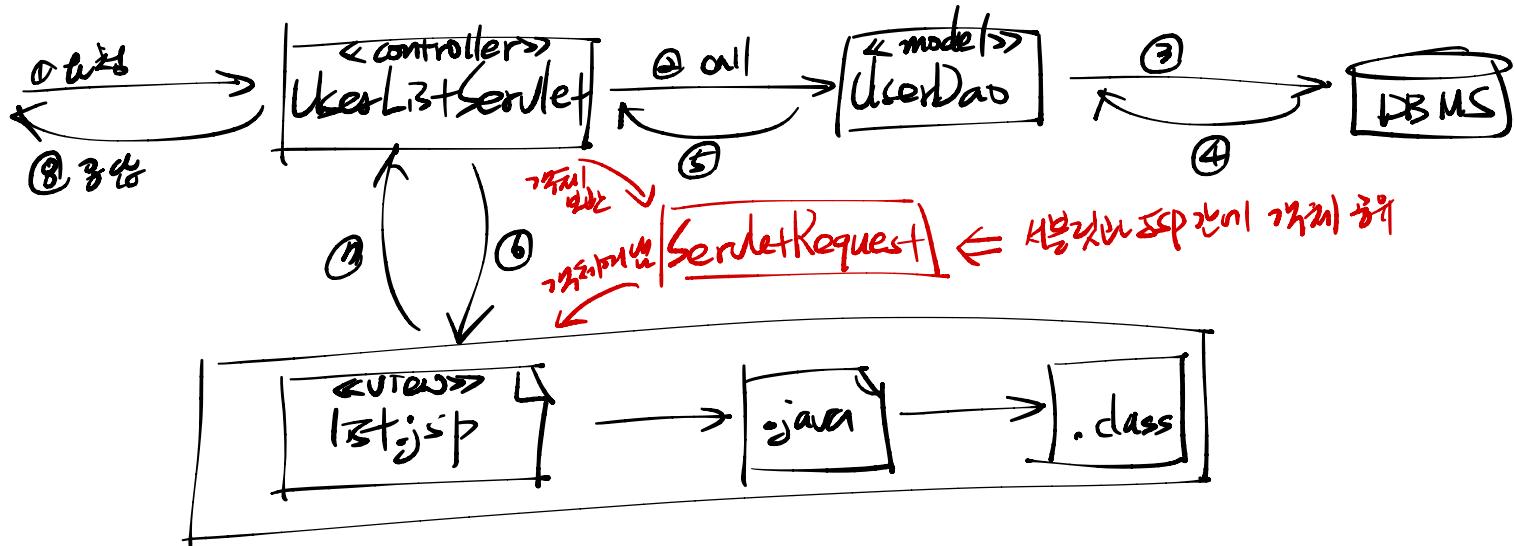
* 서블릿에서 웹면 흐름을 통제할 때의 예는 아래

- ① 코드가 쓰여있지 않은 유지보수하기 힘들다
- ② 결과 표현코드를 통제하기 힘들다.

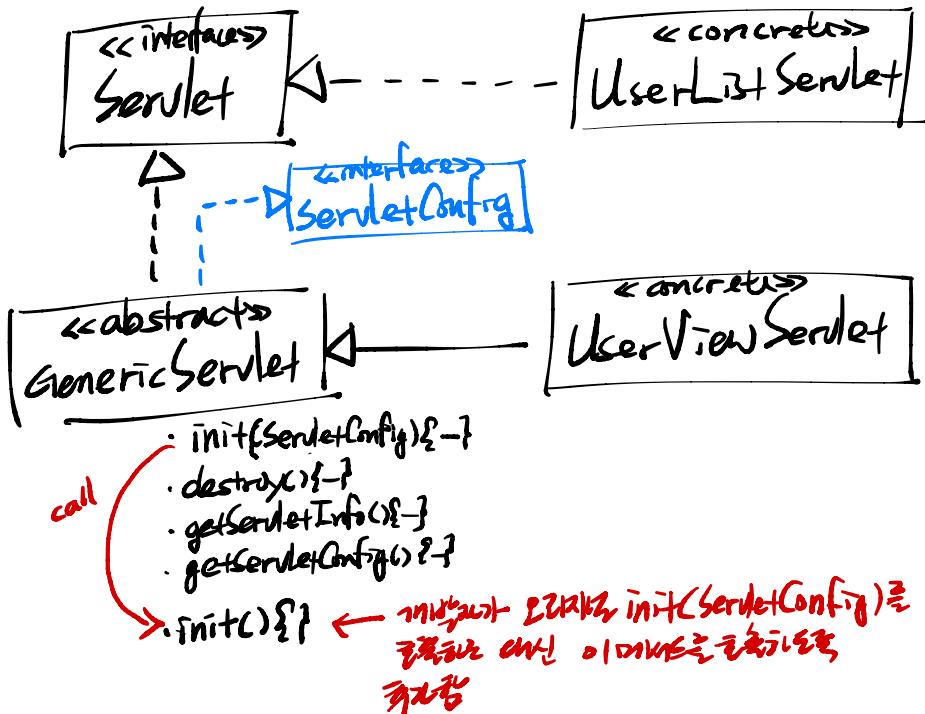
결과적으로 디자이너가 통제해 어렵다. 정부에 있는 경우 관리하기 어렵다.



* MVC2 ဆို ဘွဲ့တော်များ



* 亂世のアーキテクチャ



- init(){ } *
- service() *
- destroy(){ }
- getServletInfo(){ }
- getServletConfig(){ }

service() ↗

56. DAO는 로그를 뿐만 아니라

* 확장

- DAO가 내용을 넣기 위해 데이터가 있는 풀과 데이터 준비
- JSP가 "
- 풀과 관련된 풀과 처리
- 데이터를 넣기

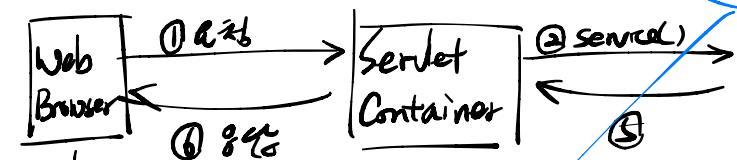
③ insert()
insertMembers() <--> Model

④ insert()

ProjectDao

DBMS

• 데이터 영역성을 처리



⑤ 데이터 처리
⑥ 결과

⑦ 데이터 전송
forward

error.jsp

• 오류페이지 사용

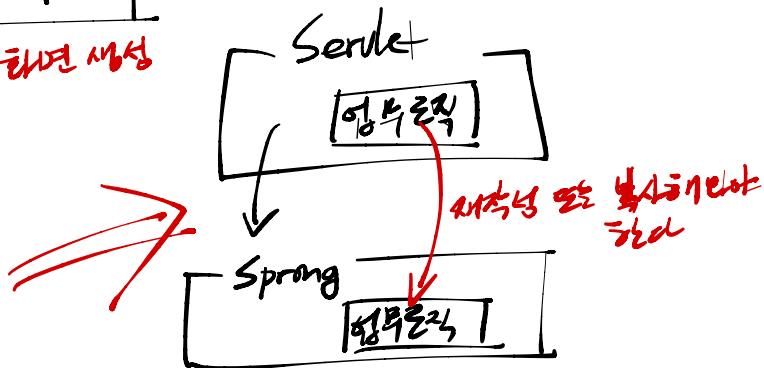
영역간 풀과의 관계에 따른 데이터
Servlet 가용성이 유지된다

↓
Servlet 가용성이 유지된다.

↓
Servlet은 다른 기술로 고쳐야되며
이유로는 오작동이나 안전성이 있다

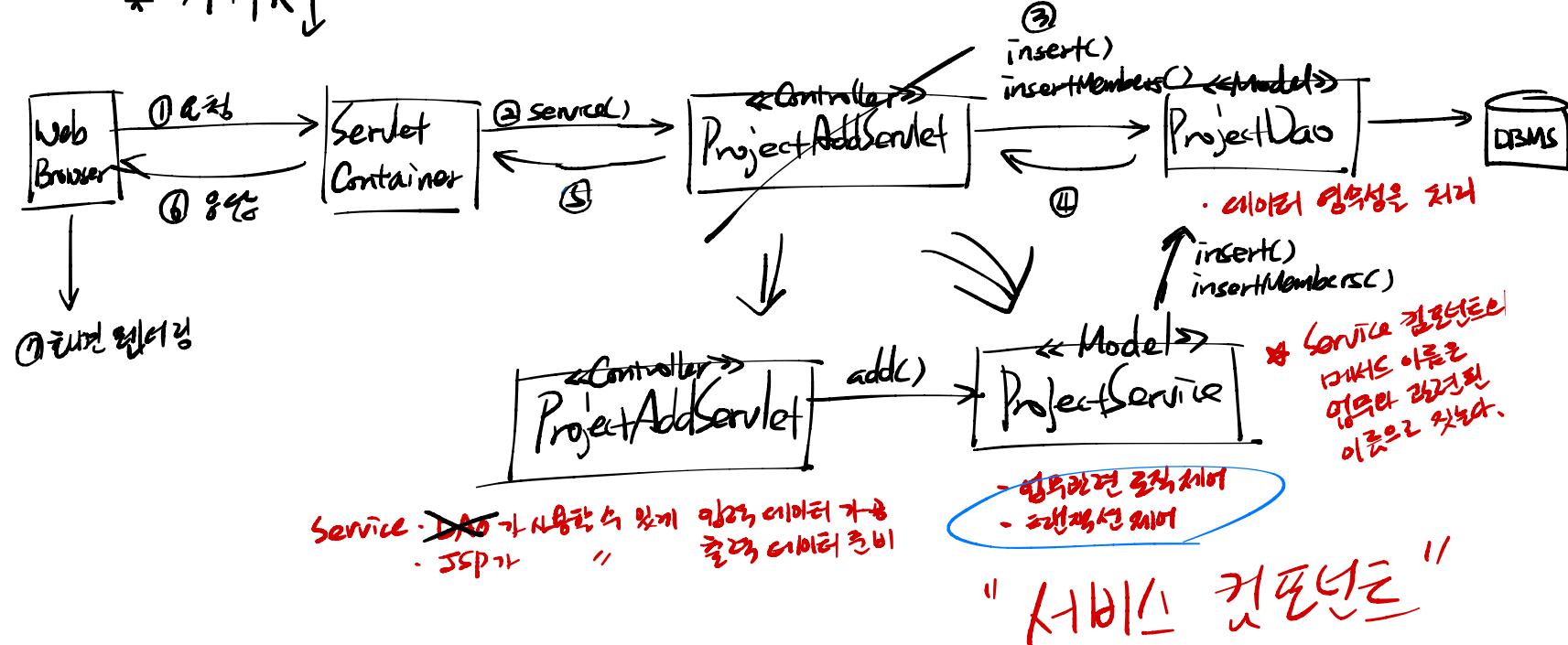
자바Beans
EJB

유지보수
변경하기



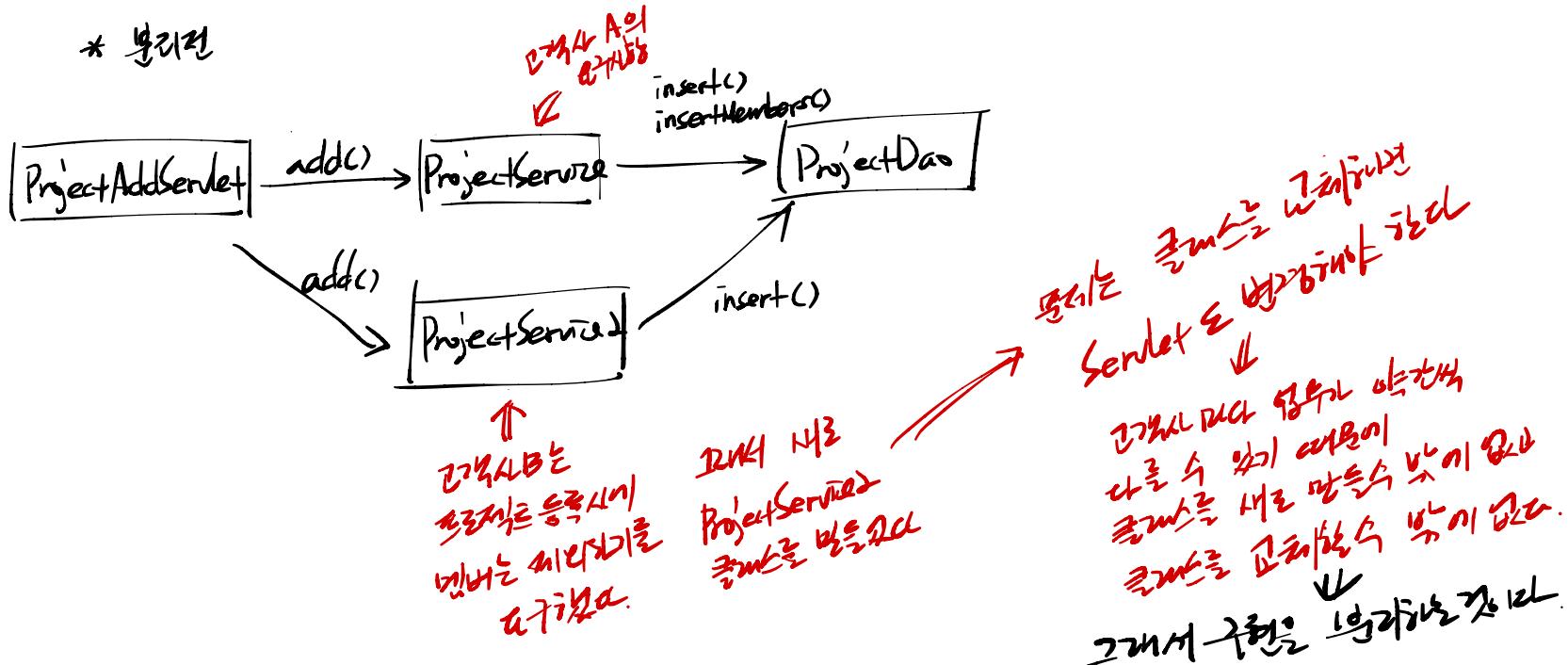
56. 웹 서비스로의 분리화하기 - 서비스 컨포넌트 등장

* 개수↓



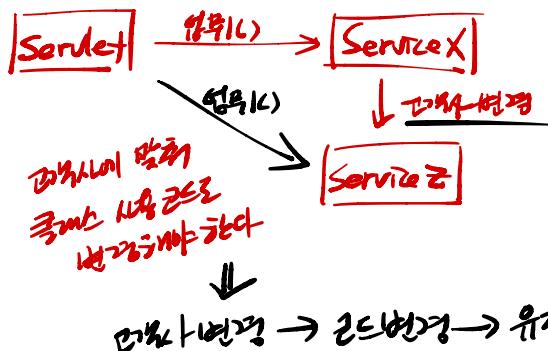
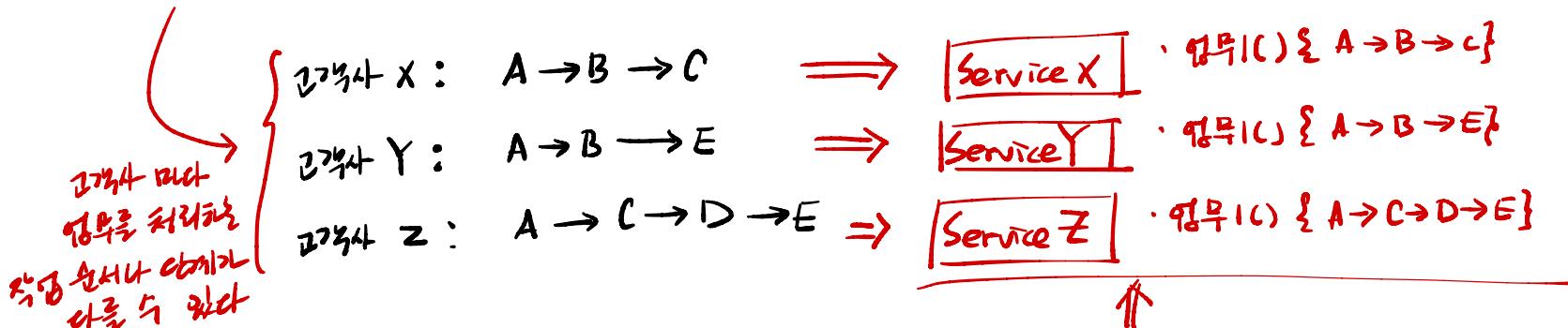
56. 웹了很久 프로젝트 - 인터페이스와 구현을 분리하기

* 분리점

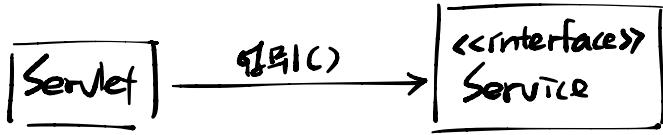


* 고객사와 업무

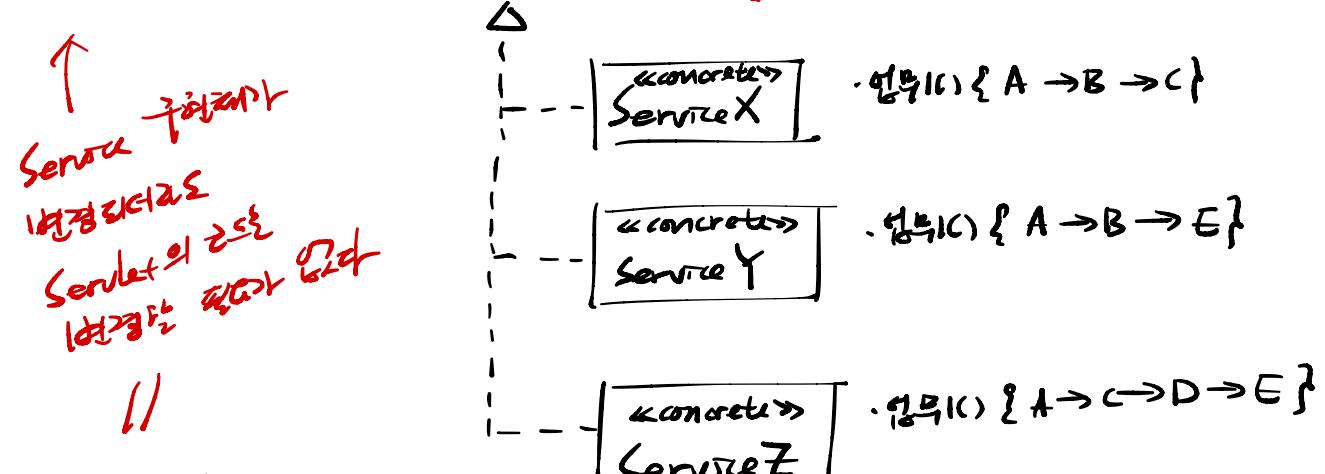
업무1: A작업 \rightarrow B작업 \rightarrow C작업 \rightarrow D작업 \rightarrow E작업 (단순한 업무흐름)



고객사 업무에 맞춰 동작하는
클래스를 따로 만들거나 하는데



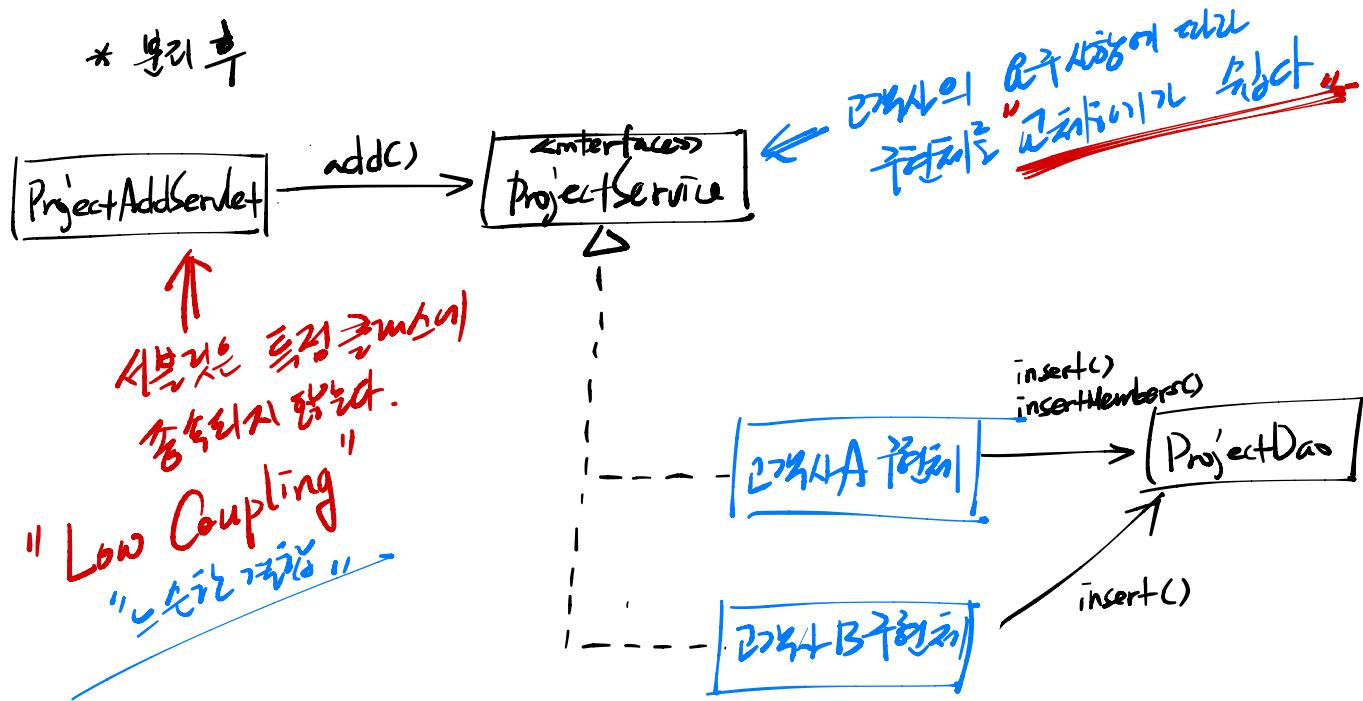
제작자에게
제공되는
제작자
제작자



“제작자에 제작자는
 → 제작자”는 제작자에 제작자!

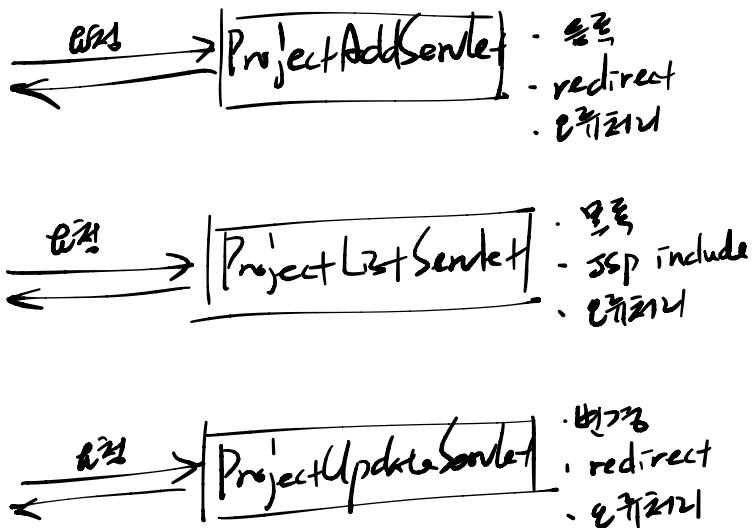
56. 헤리티지 풀다운하기 - 인터페이스와 구현을 분리하기

* 분리 후

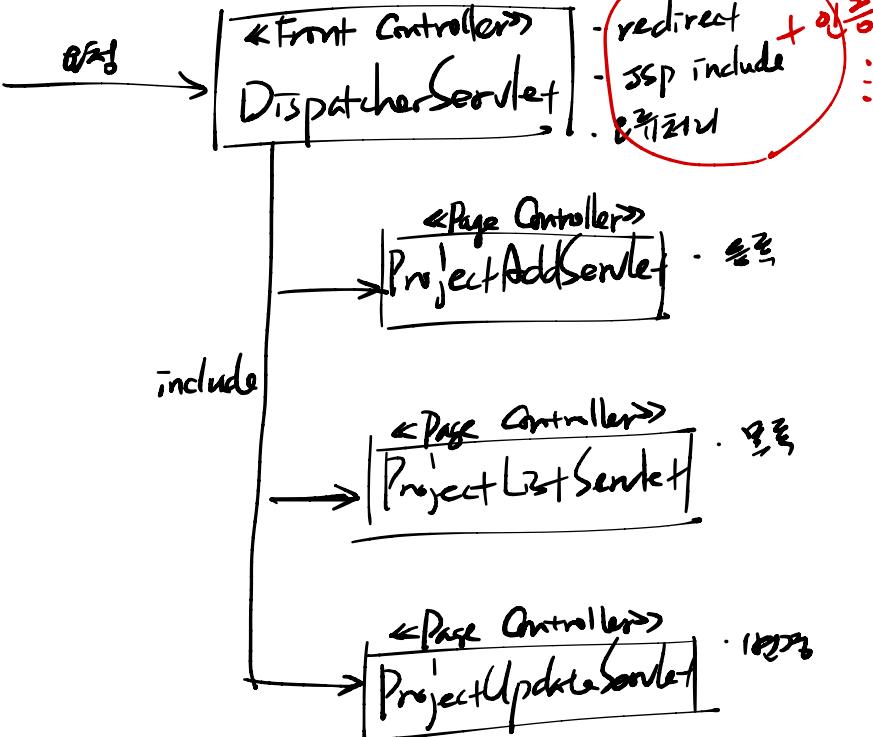


57. Front Controller \Controllers သိမ်

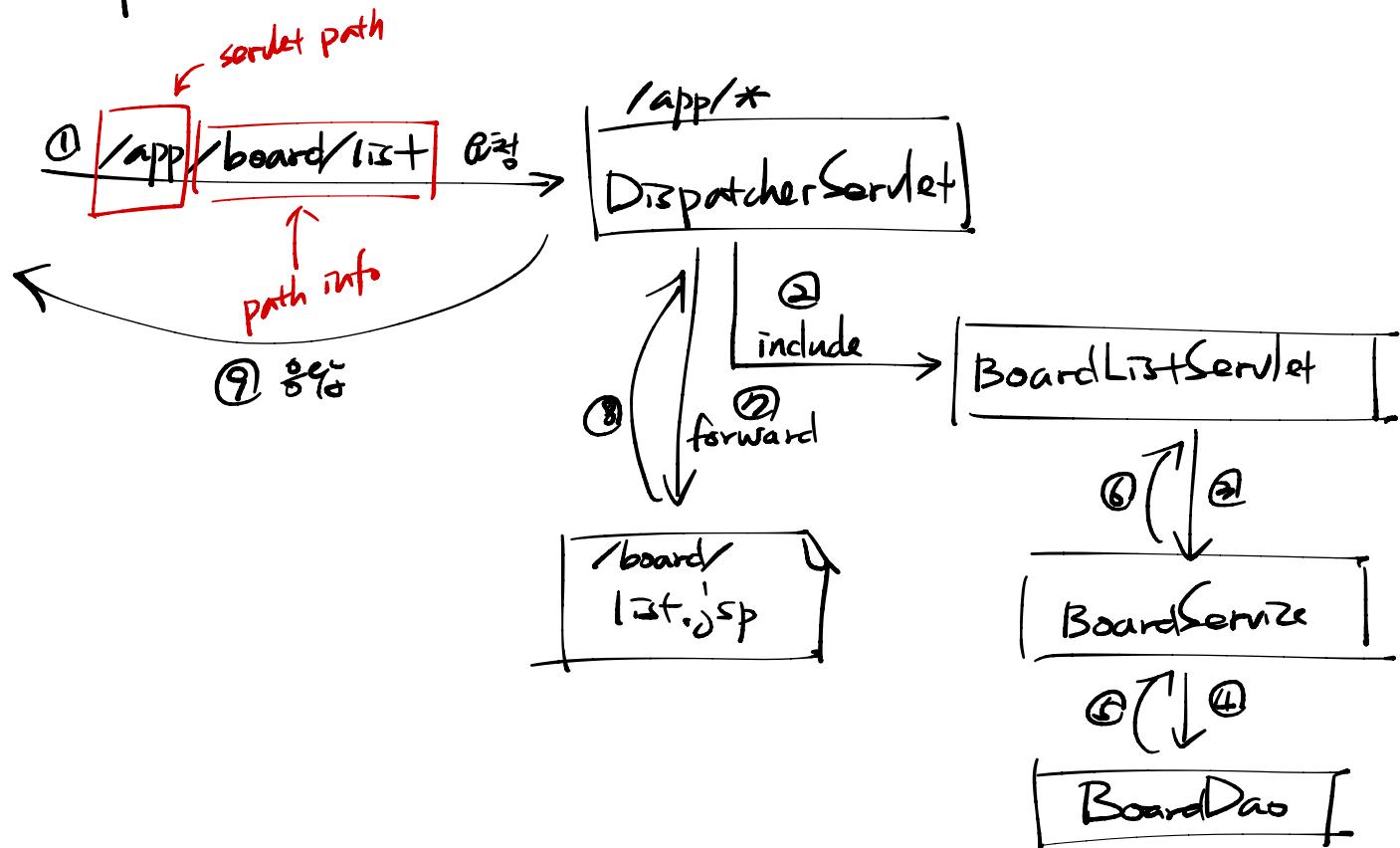
* အကျဉ်းချုပ်



* အကျဉ်းချုပ်

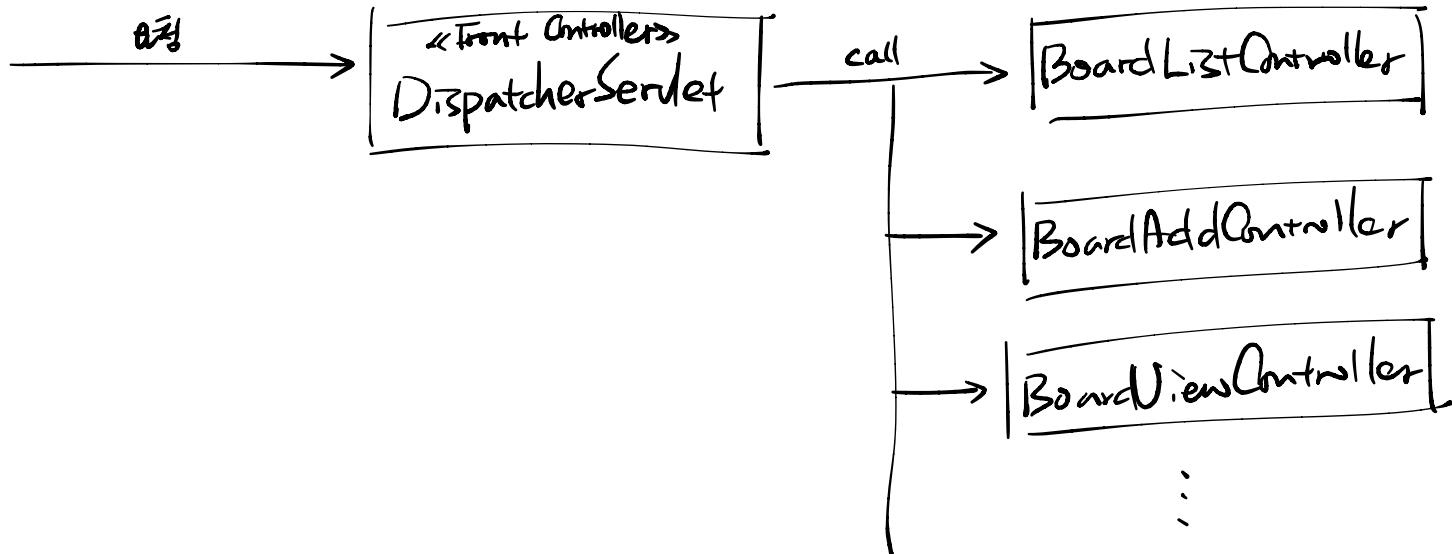


* DispatcherServlet



58. ~~제이비디~~ → ~~간단한~~ $\frac{1}{2}$ POJO 를 바꾸기

Plain Old Java Object ← 일부 기능을 쓰지 않고, 새롭게 기능을 추가하는
수정하기 쉬운 것



58, $\text{HTTP}(2) \rightarrow \text{HTTP}(\frac{1}{2})$ POJO 3 바꾸기

($\frac{1}{2}$)

@WebServlet("/user/list")

```
class UserListServlet  
    extends HttpServlet {
```

UserService userService;

void init() {

}

void doGet(HttpServletRequest request)

=

}

}

⇒

controller map nl
→ $\text{HTTP}(\frac{1}{2})$ 3번 째 짜리

POJO

```
class UserListController {
```

UserService userService;

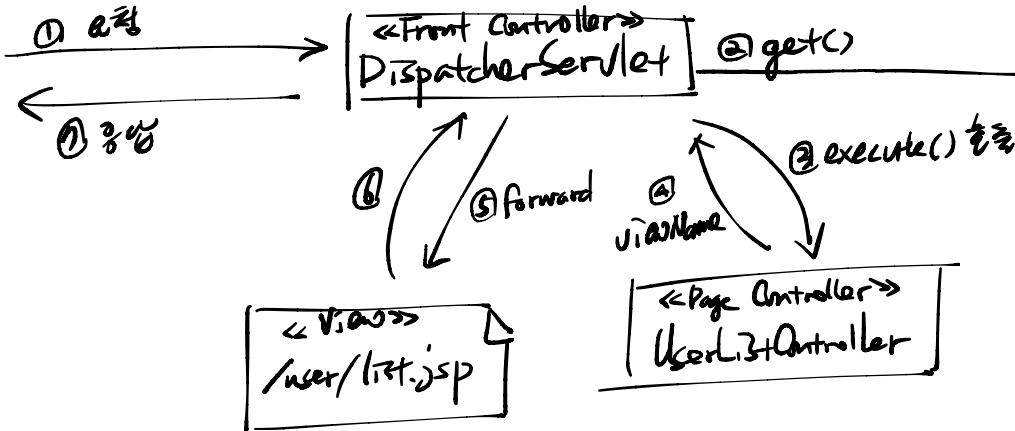
HttpServlet UserListController (UserService us) {
 userService = us;
}

void execute(HttpServletRequest request)

=

}

58. ~~제10(2) 칸드풀러~~ POJO 3의 구조 - 쪽지보내기



controllerMap : Map<String, Object>

key	value
"/user/list"	UserListController

58. 제이지 컨트롤러² POJO로 바꾸기 — CRUD 기능 추출하기

* 현황

* 개선

UserAddController

UserListController

UserViewController

UserUpdateController

UserDeleteController

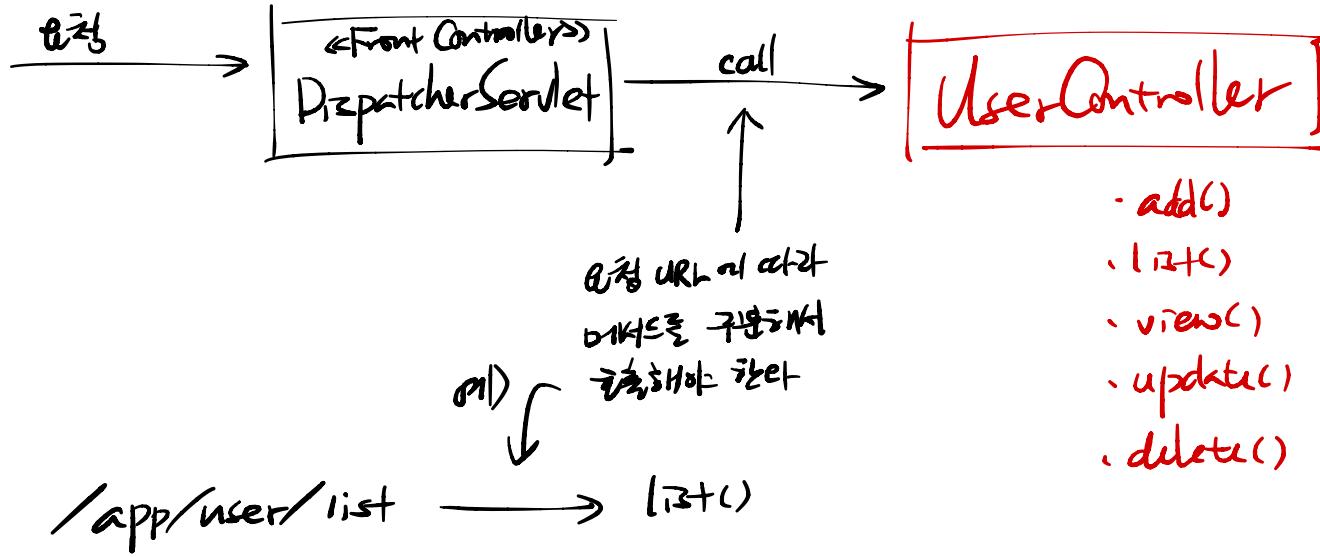


UserController

CRUD 작업
기능이 분리되었음
모듈화
정형화된 코드
구조화된 코드
제작 효율화

- add()
- list()
- view()
- update()
- delete()

58. 데비지 컨트롤러² POJO로 바꾸기 - CRUD 기능 허용하기 \Rightarrow 요청에 따라
서비스 주입은 가능해?



그렇지? 서비스 주입은 가능해!
서비스 주입은 가능해!

58. 제이비디 컨트롤러 POJO 바꾸기 - CRUD 기능 추가하기 + 어노테이션 활용
+ Reflection API 활용

