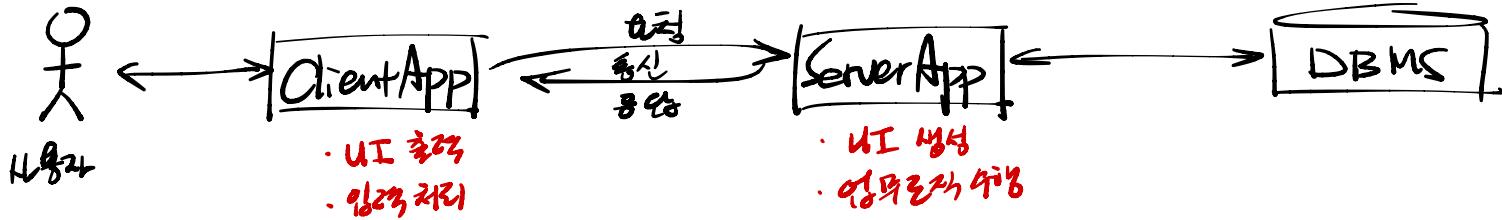
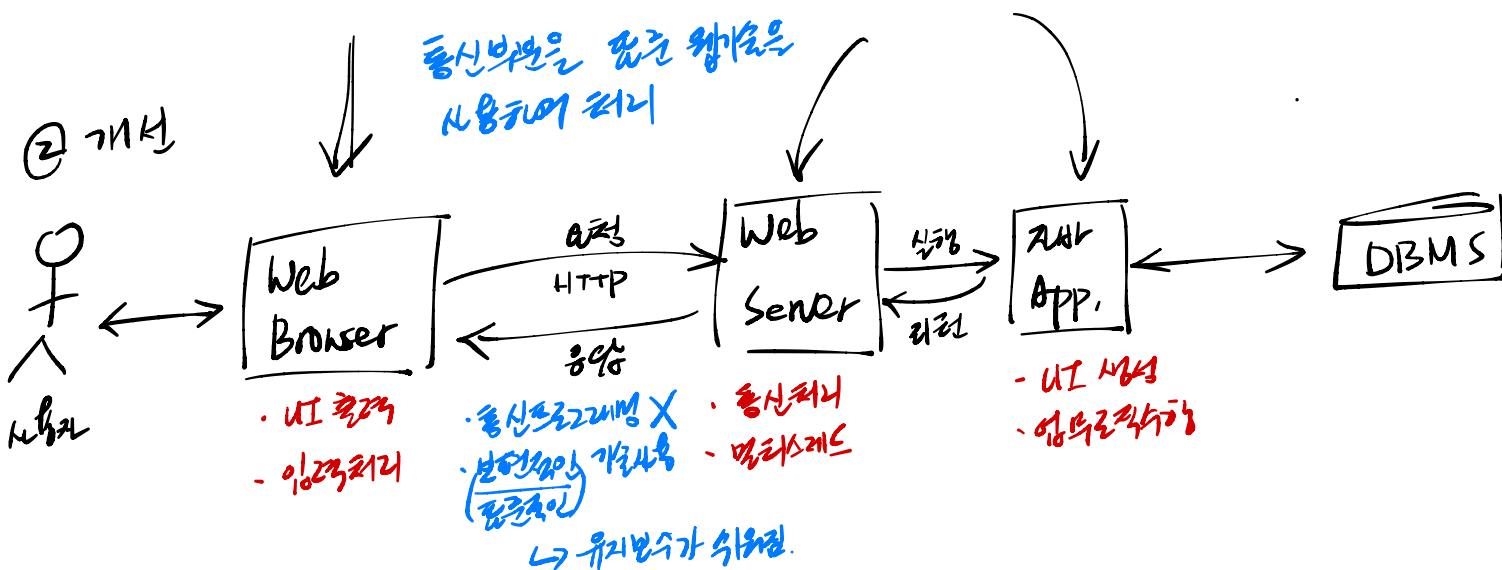


## 49. 웹애플리케이션 구조를 살펴보자

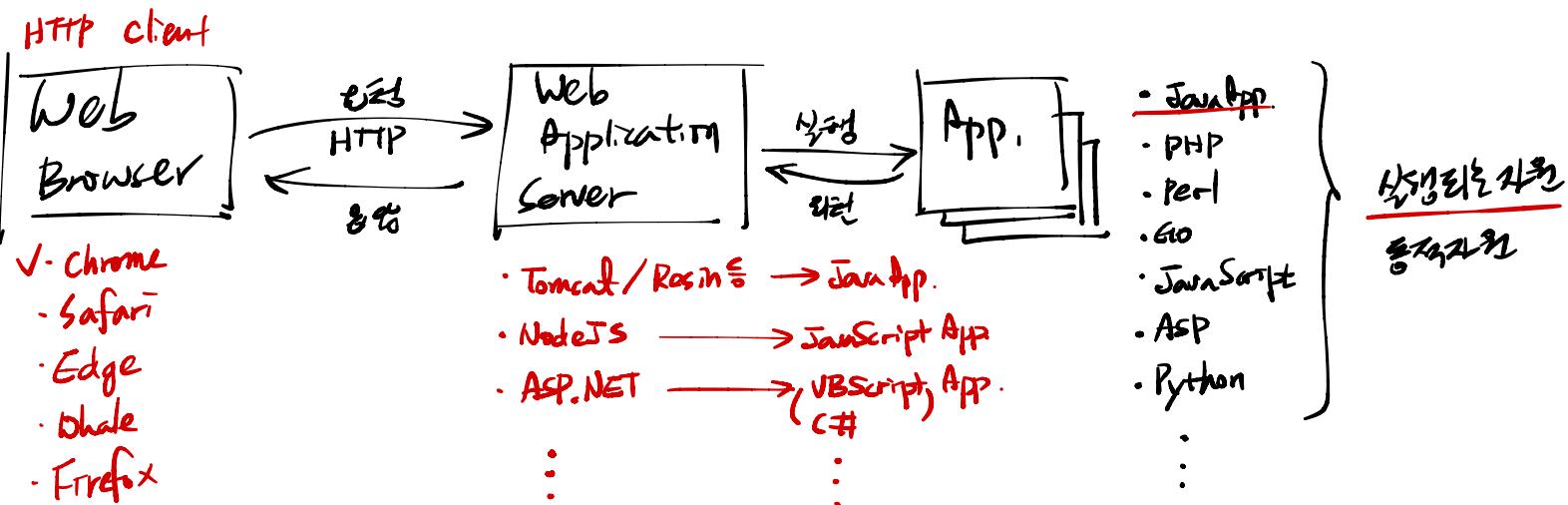
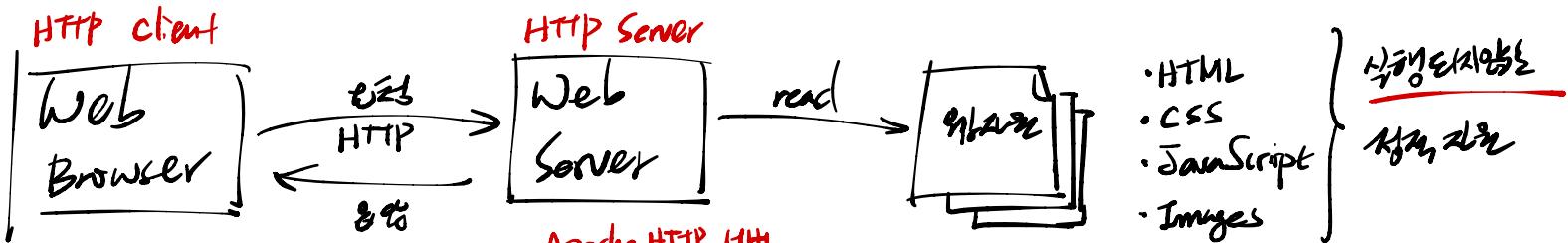
① 단계 → 전용 프로토콜을 사용하여 클라이언트/서버 통신



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

응답  
응답

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

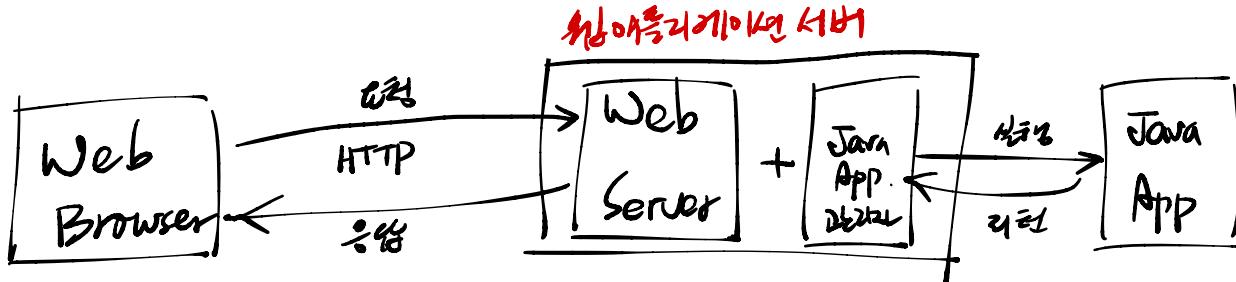
③ 흐름기 유형 애플리케이션 (인수/값 처리, 하위 시스템 등 입력/생성)

↳ 흐름 애플리케이션 → OOP 프로그래밍 기법 도입  
구현하기 쉽다

{ Java 언어  
.NET  
= }

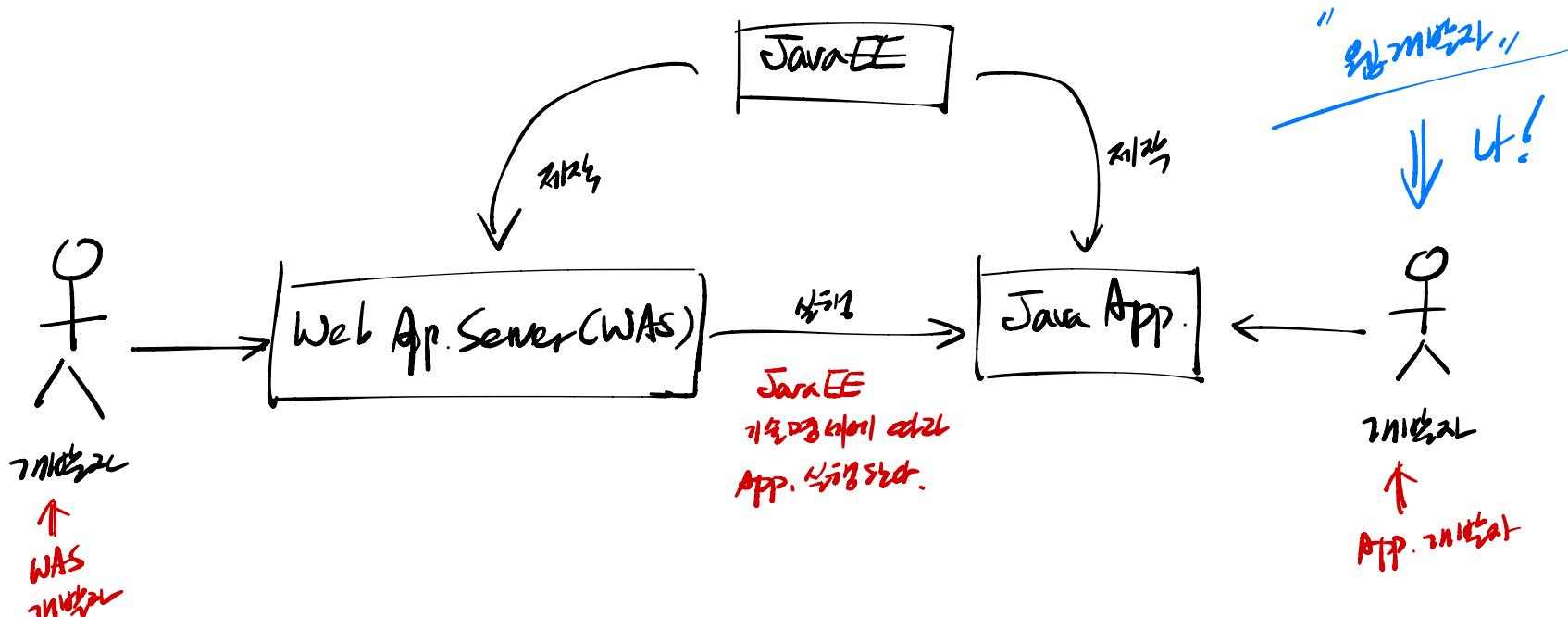
↑ C/C++ 보다 모듈화  
단위별 파일,  
시스템별 분리  
설정이 어렵다  
↳ 접근성이 X

## \* 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, 접근제어 등

애플리케이션

Web

<del>Servlet/JSP</del> JSTL & EL	<ul style="list-style-type: none"><li>웹 애플리케이션 개발 기술</li></ul> <p>→ 웹 API로 변환!</p> <p>⇒ REST API</p>
EJB	<ul style="list-style-type: none"><li>분산 처리 기술</li></ul>
Web Service	<ul style="list-style-type: none"><li>웹 기반 분산 처리 기술</li></ul>
Authentication & Deployment	<ul style="list-style-type: none"><li>인증 및 배포 기술</li></ul>

## \* Java EE 7/8/10/11 버전의 특징

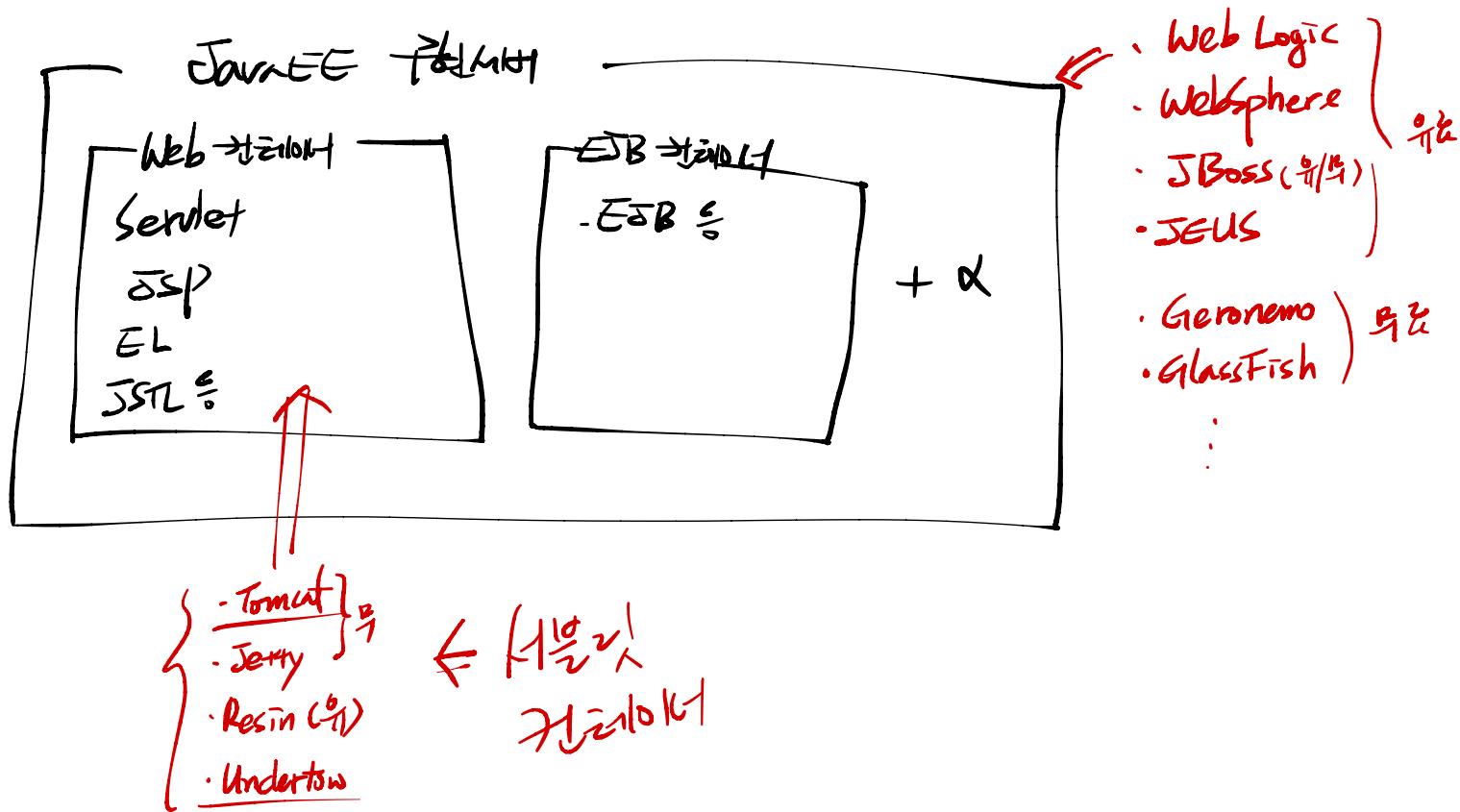
<del>JavaEE 버전</del> <del>제작년도</del>	Servlet	JSP	EL	EB
5	2.5	2.1		3.0
6	3.0	2.2	2.2	3.1
7	3.1	2.3	3.0	3.2
8	4.0	2.3	3.0	3.2

\* Java EE 1/2 버전부터 ESSENTIAL (Implements)

<del>JavaEE 1/2</del>	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

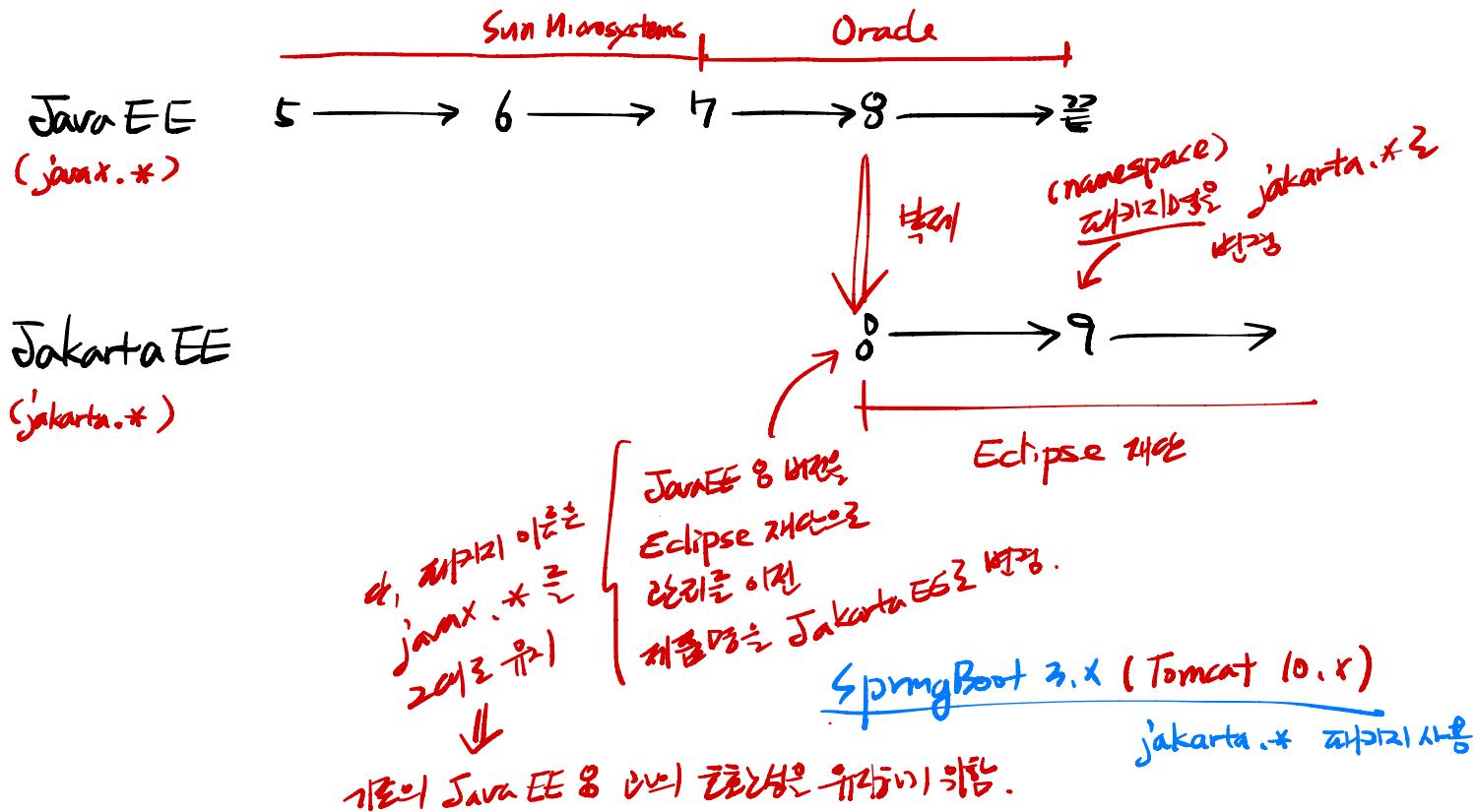
⇒ 2014년 10월 버전이 최신  
2014년 Java EE 버전으로 업그레이드 할 것.

## \* JavaEE 페미터 서브 분류

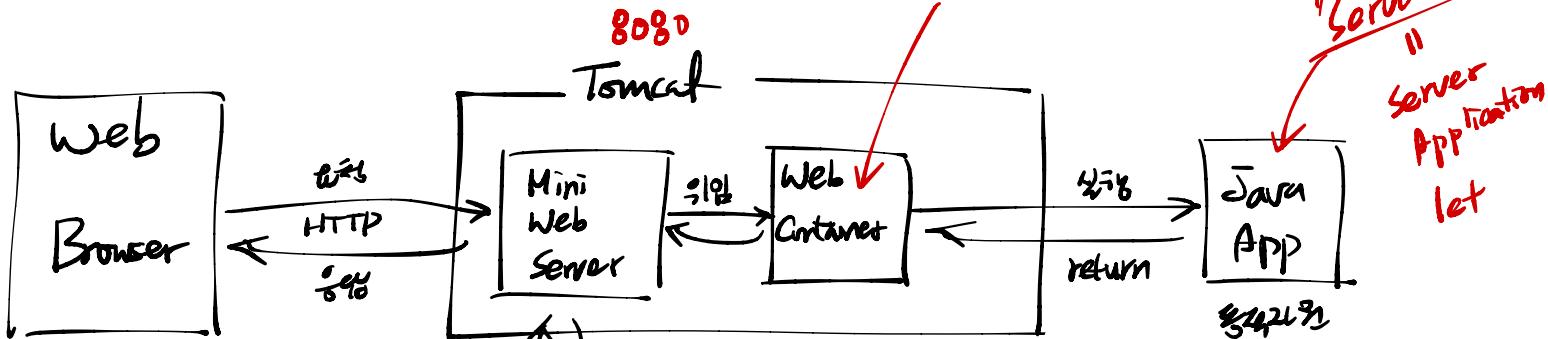


## \* JavaEE & JakartaEE

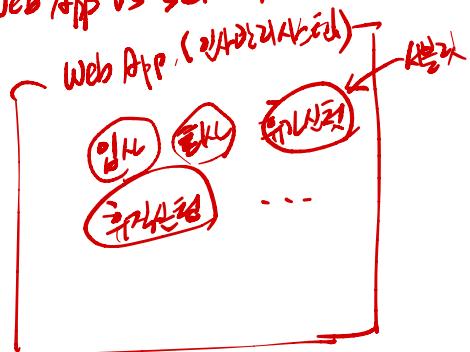
SpringBoot 2.x.x (Tomcat 9.x)



\* Java Web Application → 웹 어플리케이션



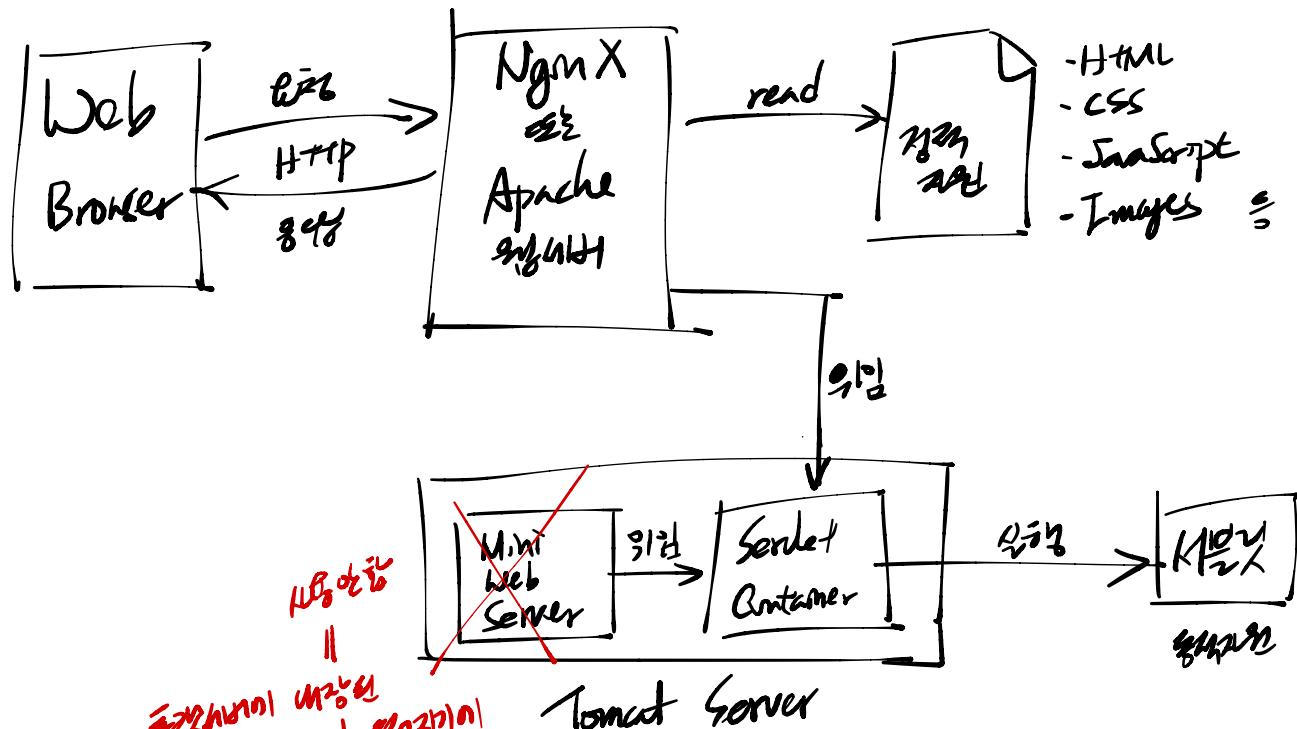
\* Web App vs Servlet



- HTML
- CSS
- JavaScript
- Images

Java EE  
- JSP, Servlet, ...  
— 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 사용하는 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)

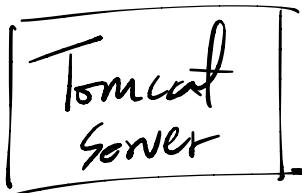
停止文件 => shutdown.bat (Windows)  
shutdown.sh (Mac, Linux)

\* Tomcat 끝내는 것

① 시작하기



startup.bat



Development (개발)

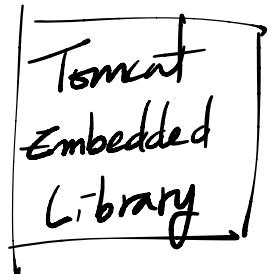
↓ .war

Operation (운영)

② 구현하기



call



운영

Spring Boot



Development + Operation = DevOps

## \* Embedded Tomcat 끝까지 써보기

new Tomcat()

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

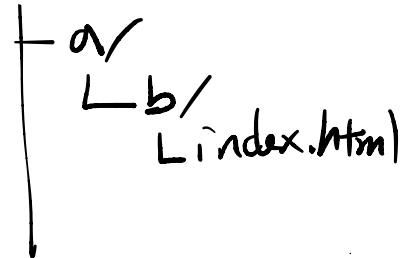
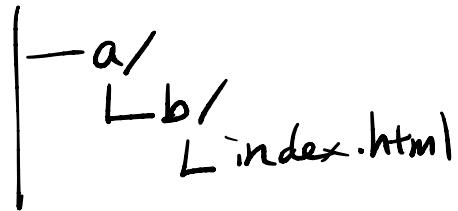
- 웹 어플리케이션 경로  $\Rightarrow$  /
  - 정적 자원 경로  $\rightarrow$  src/main/webapp/  $\leftarrow$  .html, .css, .js, .gif 등
  - 동적 자원 경로  $\rightarrow$  bin/main  $\leftarrow$  .class, .properties, .xml 등

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

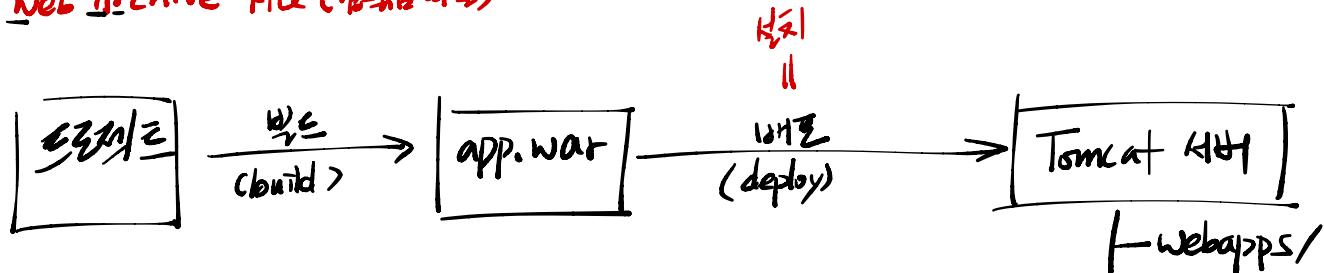
http://localhost:8888/ → src/main/webapp/



\* 웹 어플리케이션 프로젝트 배포

① .war 파일 배포

Web Archive File (웹아카이브)



② 라이브러리 + API 라이브러리  
(라이브러리)  
= springboot 앱



\* Maven 파일은 쉽게 편리한 확장자를 가짐

구조도



.html, .css, .js, .gif 등 이미지 파일

WEB-INF

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

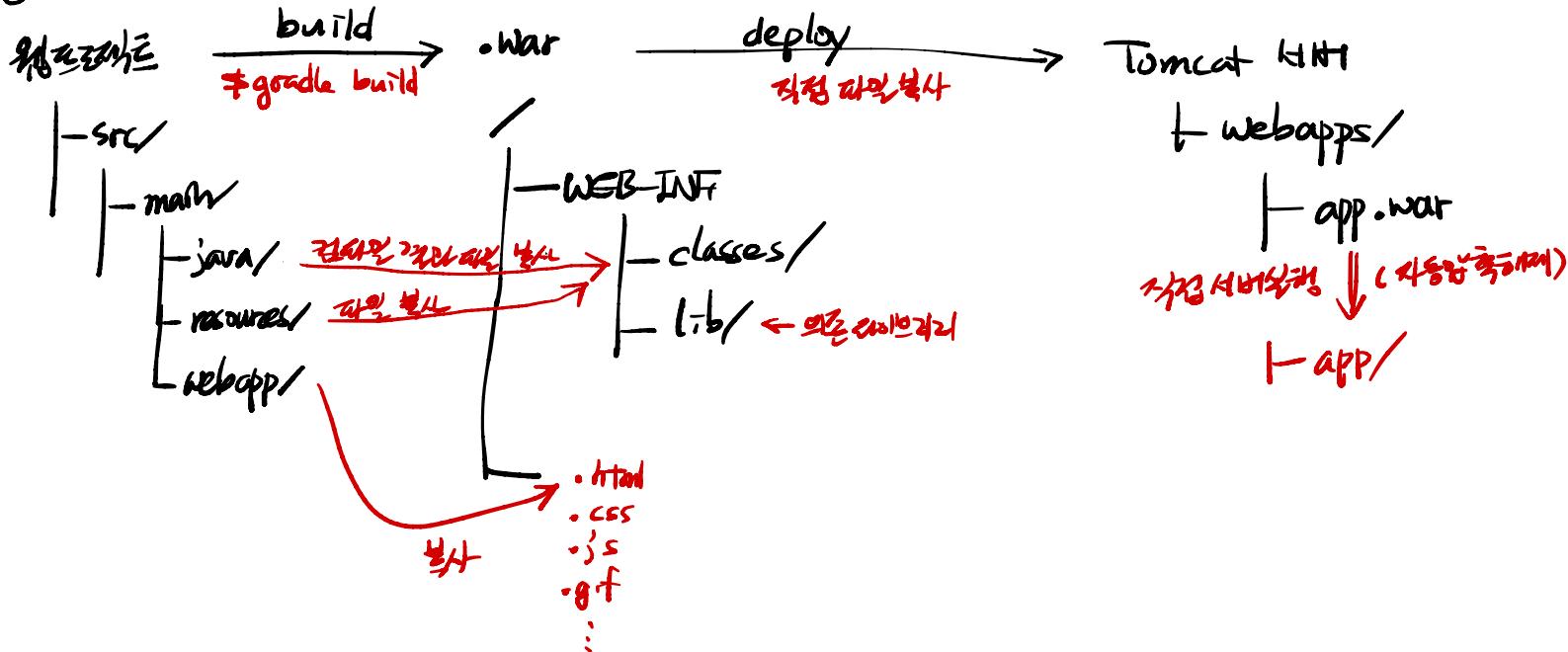
classes/ ← .class, .properties, .xml

lib/ ← .jar

.xml, .properties ← 설정파일

\* 웹프로젝트 디렉토리

① 디렉토리 구조  $\leftarrow$  웹애플리케이션 기본을 갖춘 후에 버전으로 1842



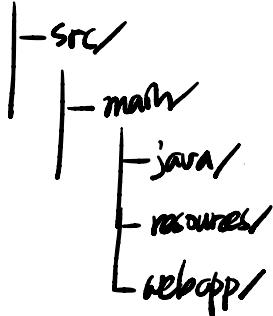
\* 웹프로젝트 파일

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 / conf, 폴더는 복사됨

- logs/ ← 실행 기록 파일

- temp/ ← 일시적 사용되는 파일을 두는 곳

- webapps/ ← 사용 애플리케이션

- work/ ← JSP 변환 파일 두는 곳

- 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

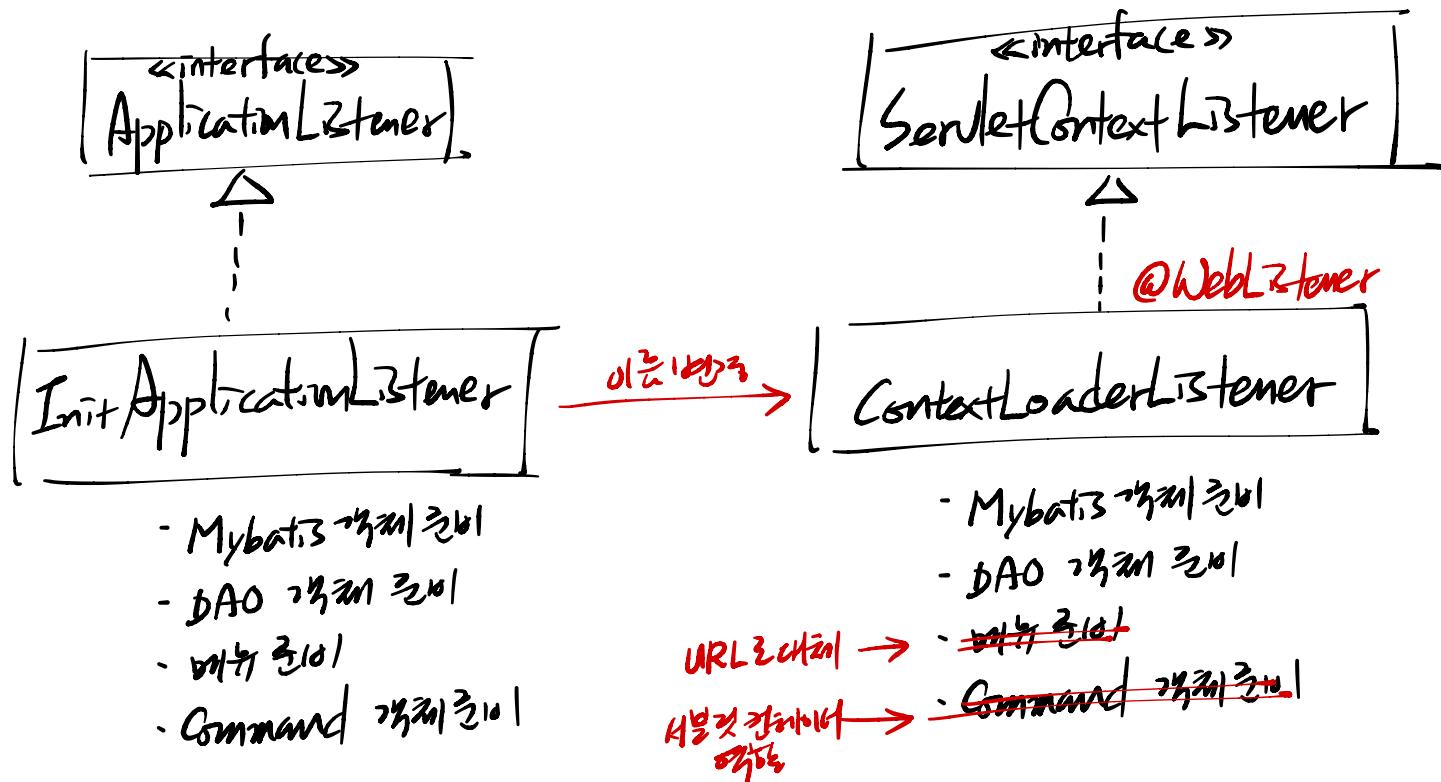
- ~~파일통신구현~~
- ~~서비스레이팅 구현~~
- ~~전송통신망~~
- ~~웹서버 구현~~

Web

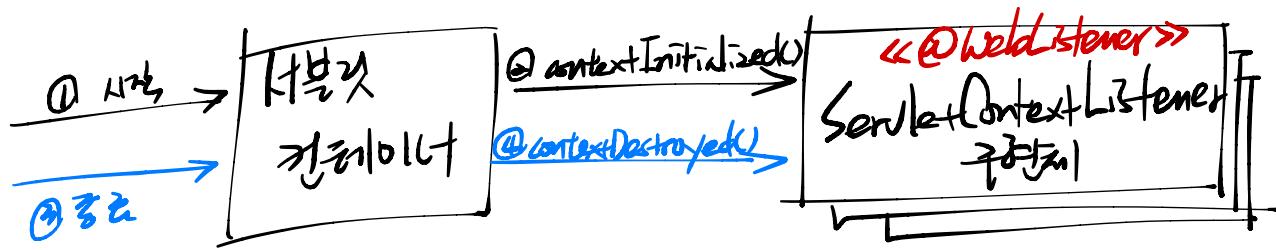
Tomcat

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

\* 리스너 만들기



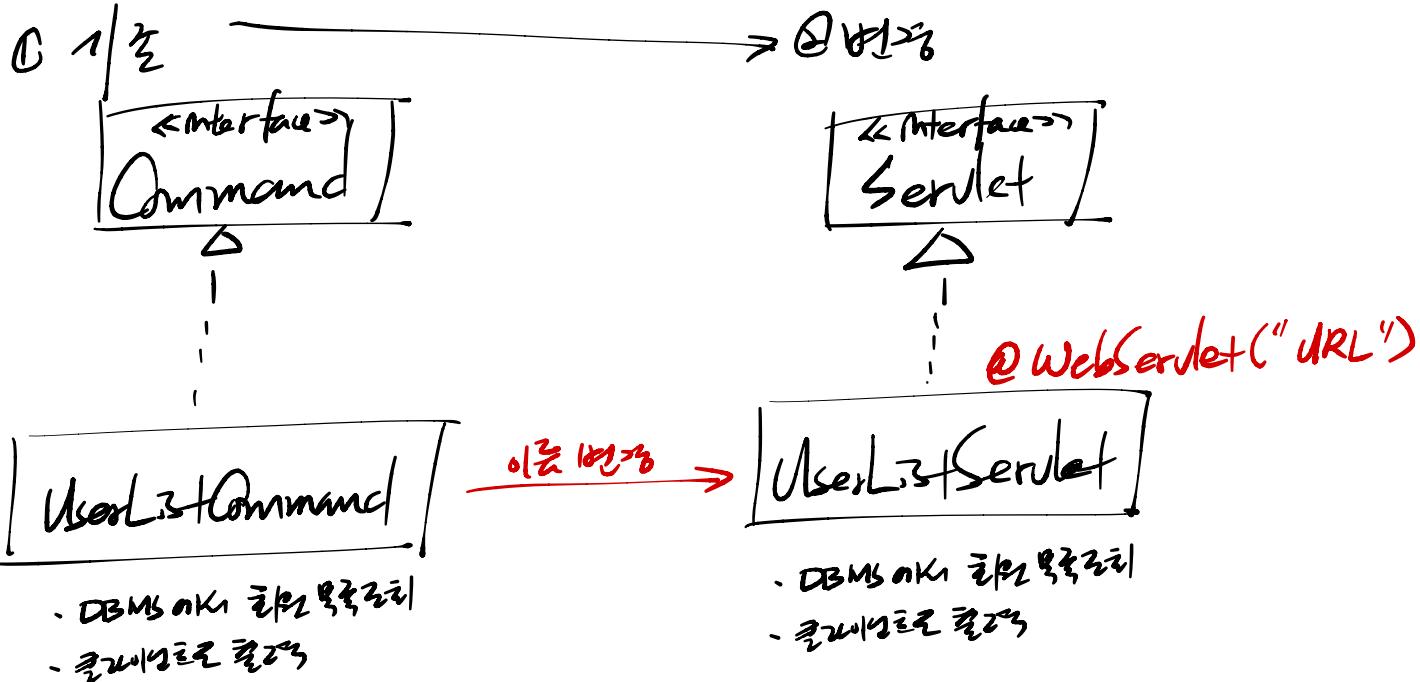
## \* ServletContext Listener



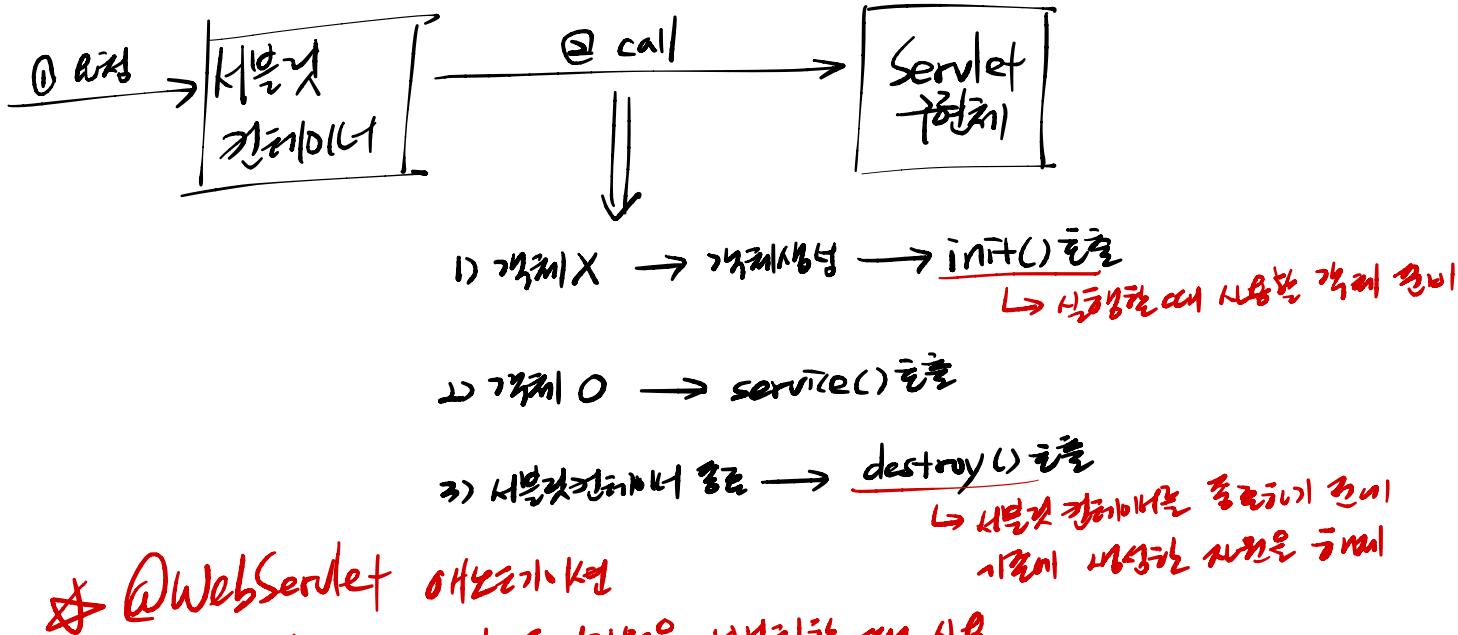
\* **@WebListener** effect하기

- 라이브 컨테이너는 서블릿 컨테이너를 통해  
라이브 컨테이너는 서블릿 컨테이너를 통해

\* 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

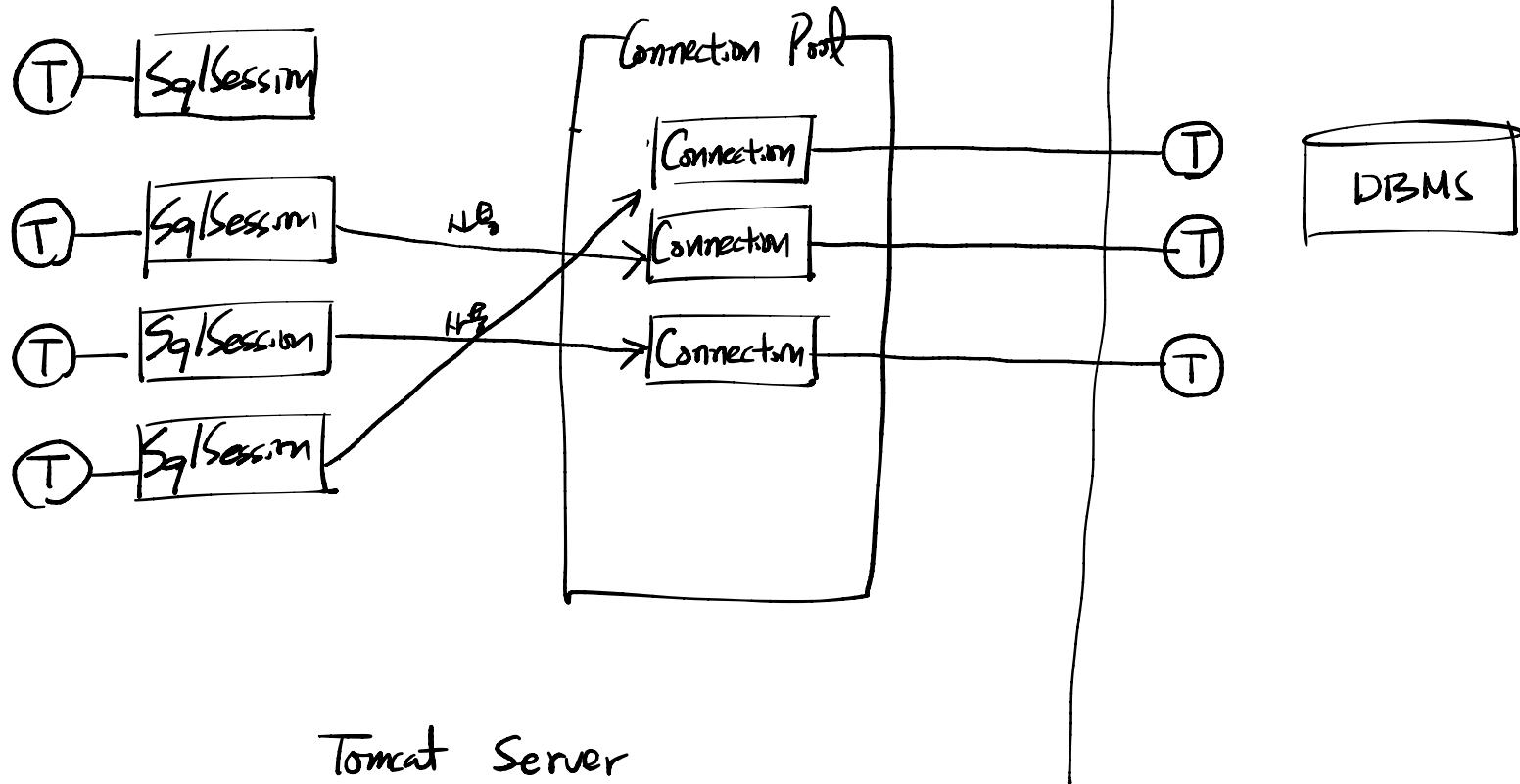
<u href="/user/view?no=11"> user11 </u>

HTML reference

→ 템플릿 엔진은  
→ 각각의  
→ 사용자  
→ 경로를  
→ 찾는다.

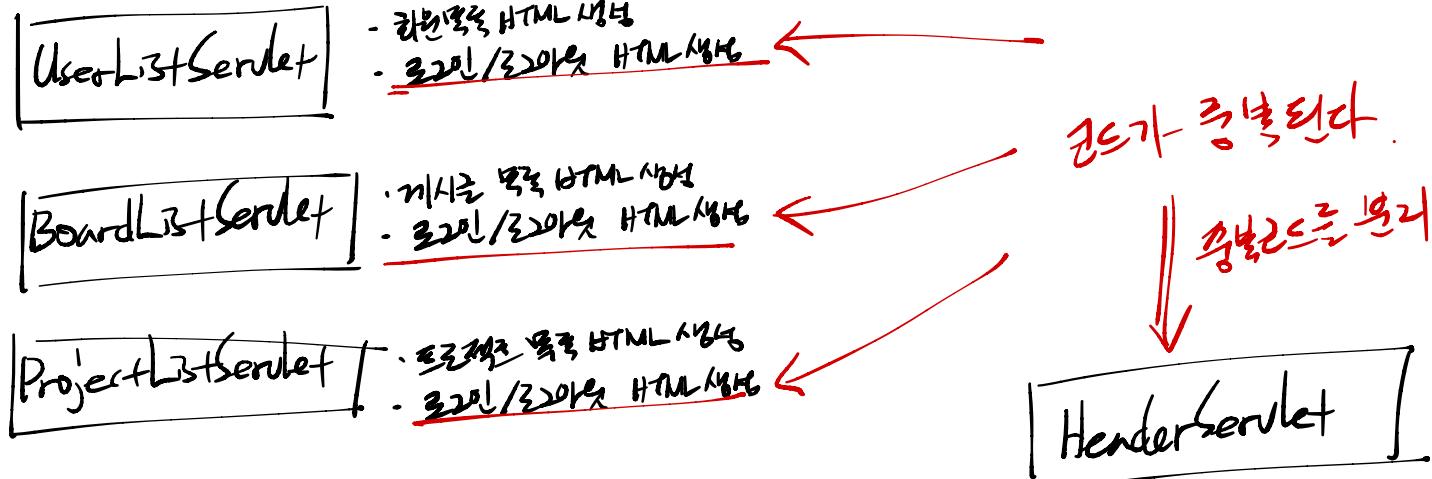
→ 템플릿 엔진은  
→ 각각의  
→ 사용자  
→ 경로를  
→ 찾는다.

## \* DBMS et SqlSession



\* 서블릿 include → 다른 서블릿의 실행 결과를 포함시키는 방법

## ① 예제



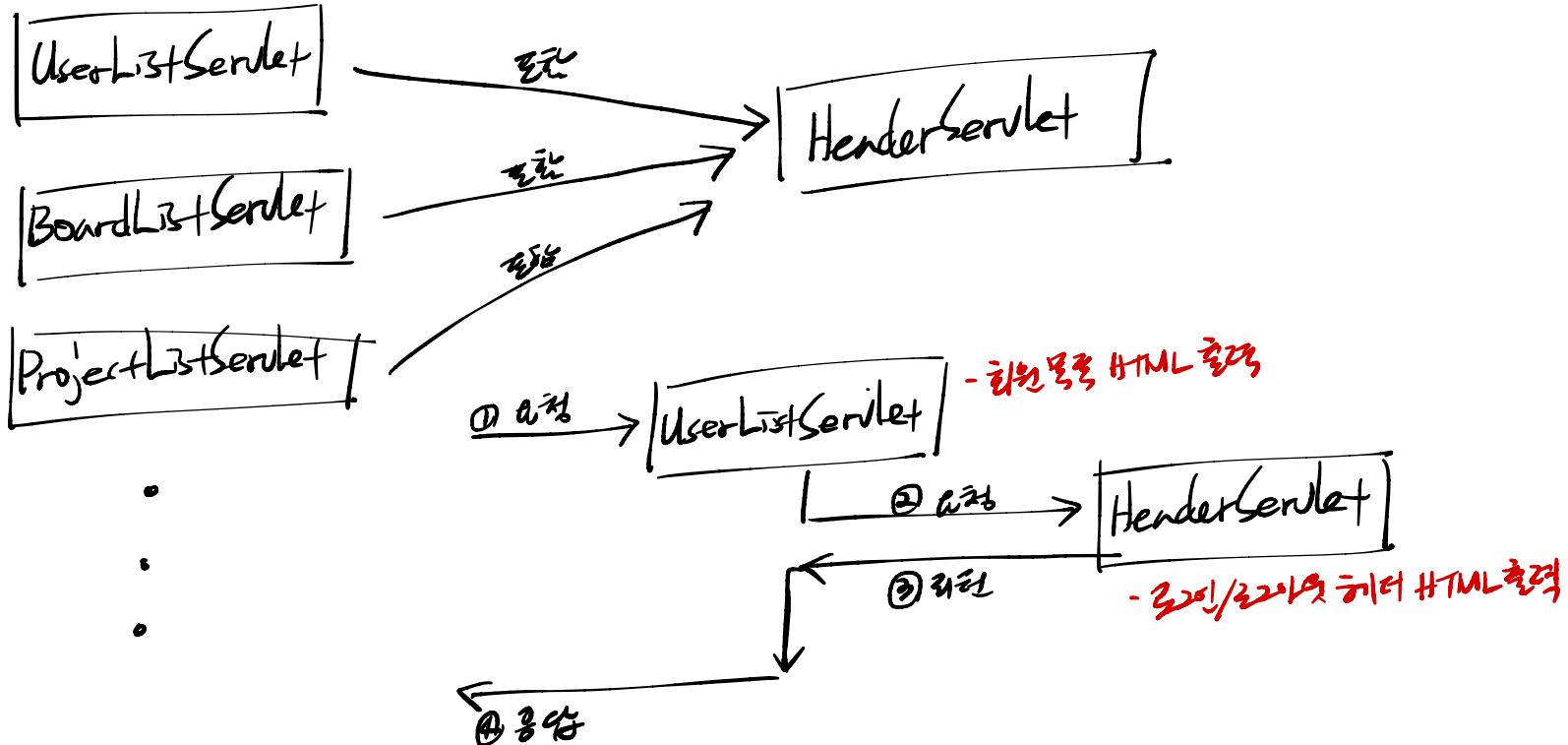
•

•

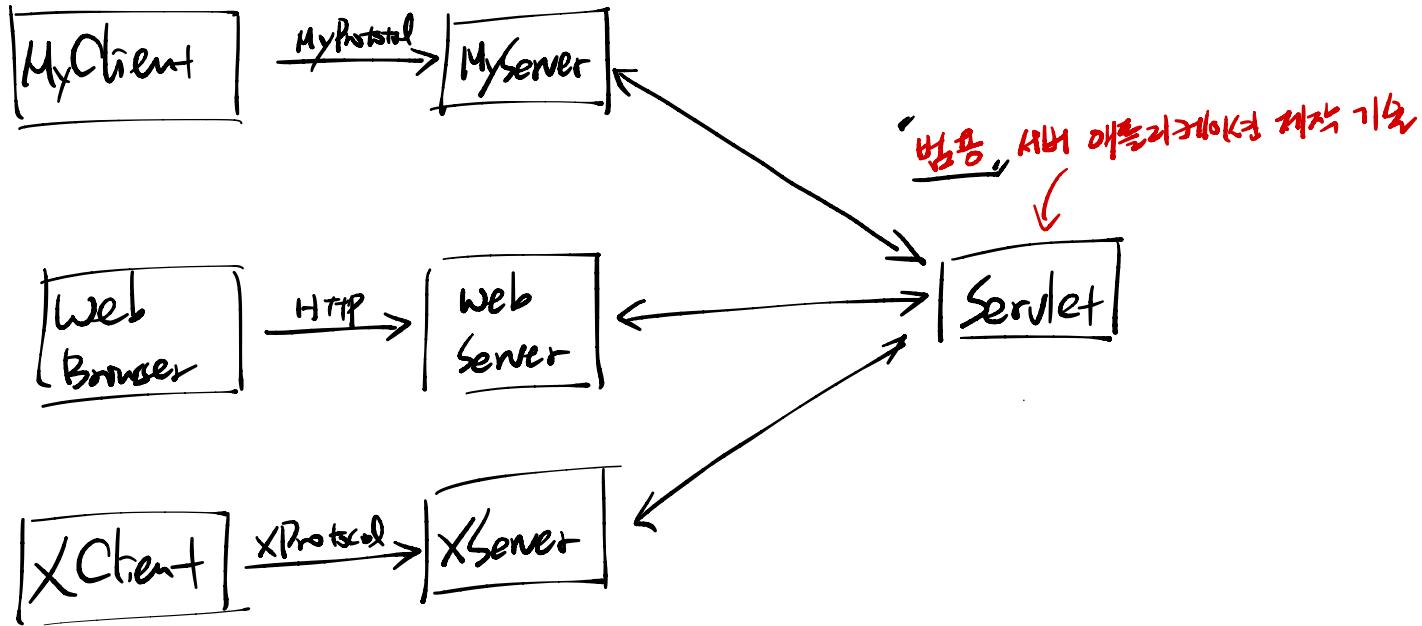
•

\* 서블릿 include → 다른 서블릿의 실행 결과를 포함시키는 방법

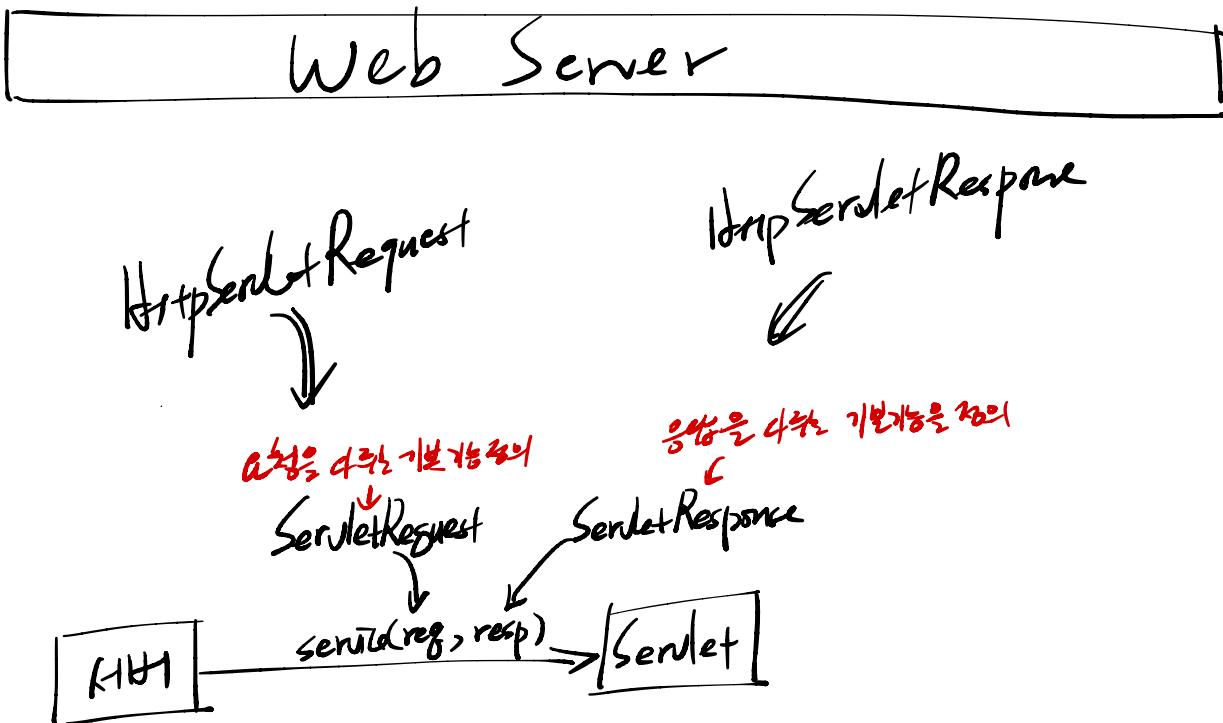
② 구조



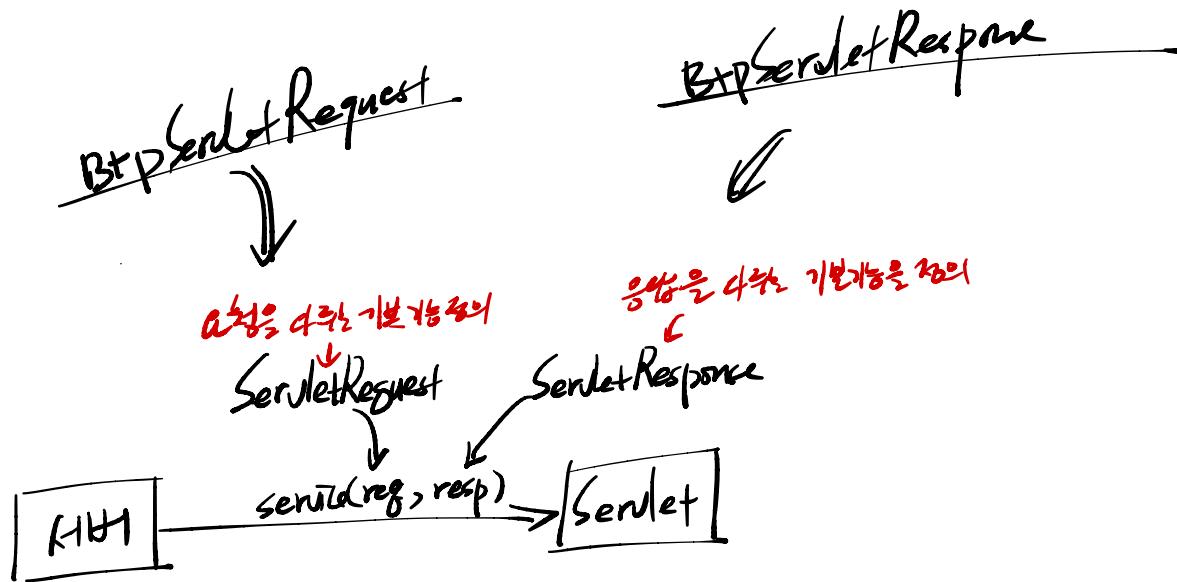
## \* Servlet چیزی چی



## \* ServletRequest et ServletResponse

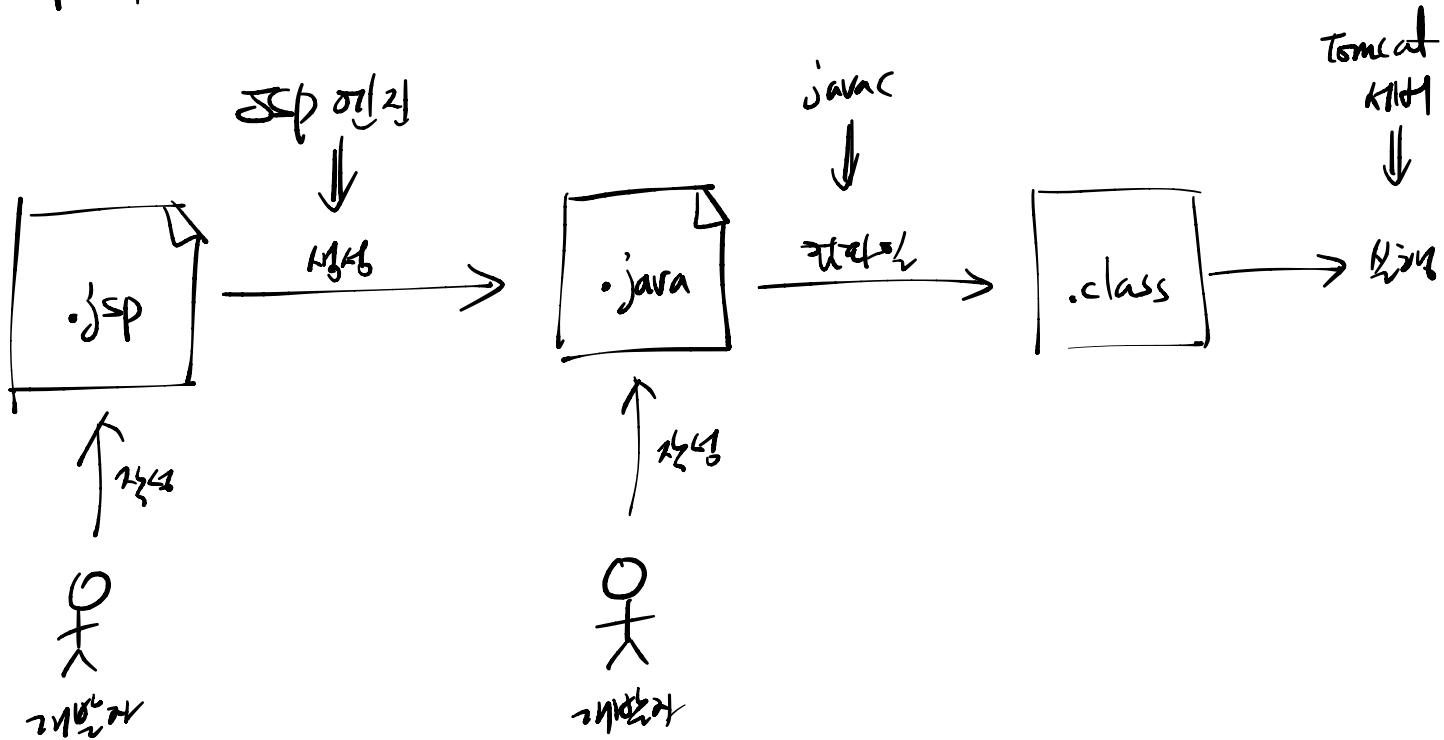


# Bitcamp Server

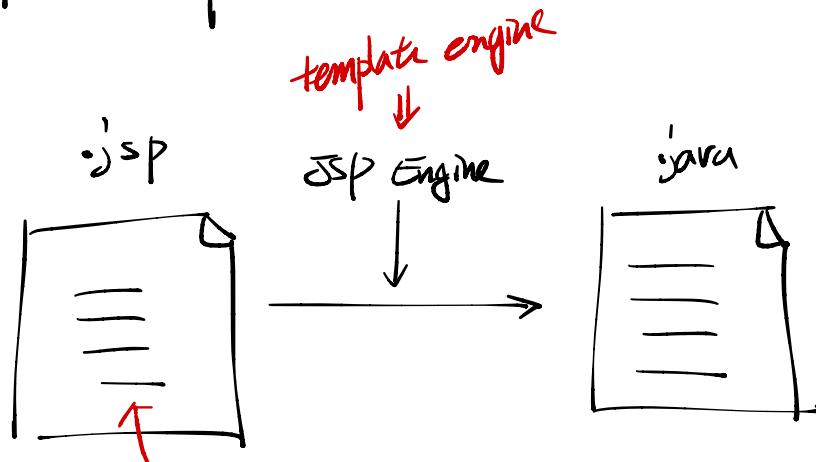


## 50. UI 흐름을 자동 생성하기 — JSP (Java Server Page)

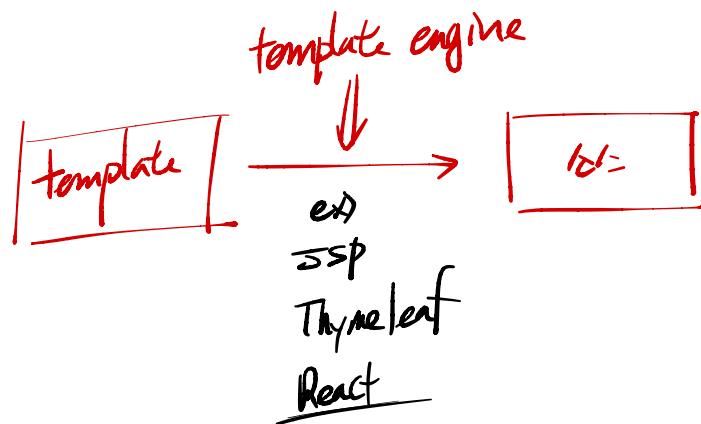
\* JSP 구동 원리



## \* JSP - Template 엔진

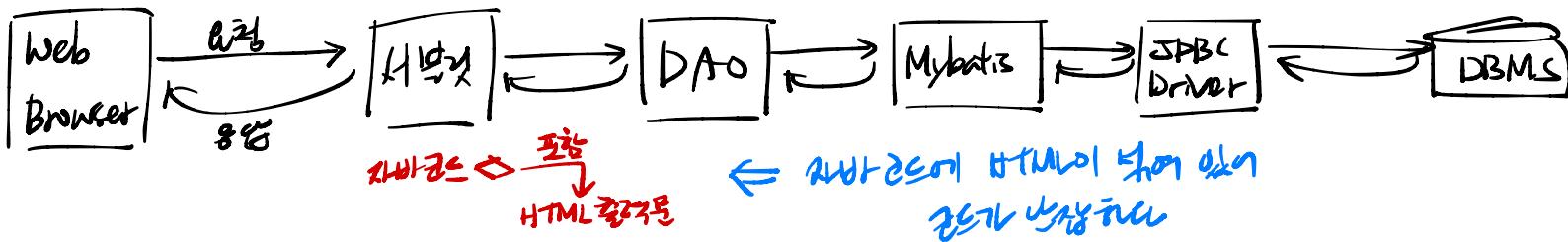


자바 코드를 만들거나  
서버를 실행하거나  
자바 코드를 출력하는  
"특수한"(template)"

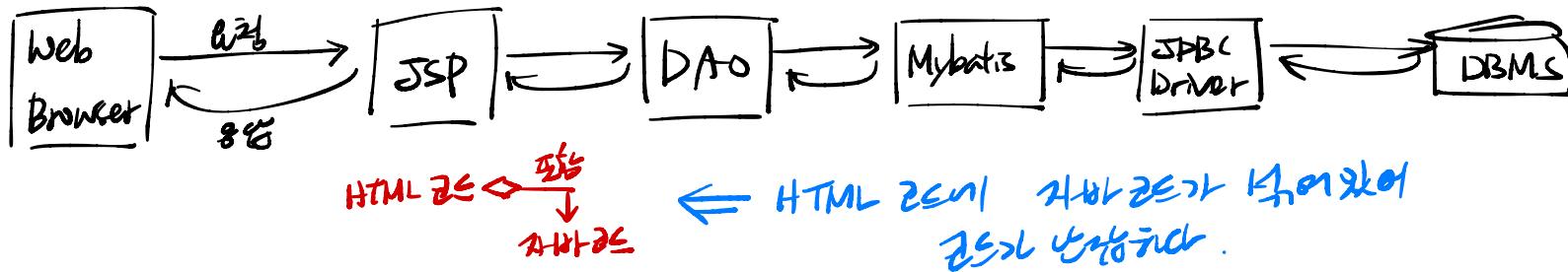


\* Servlet 와 JSP 애플리케이션 아키텍처

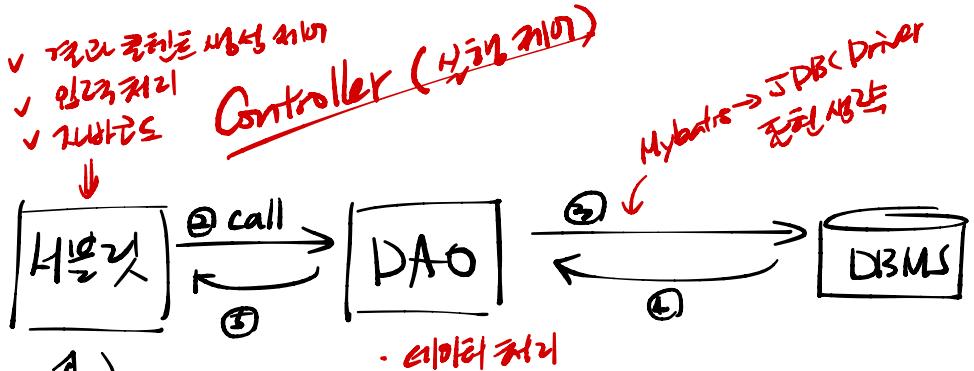
① 서블릿 애플리케이션 아키텍처 - 이전부분



② JSP 애플리케이션 아키텍처 = MVC | 확장

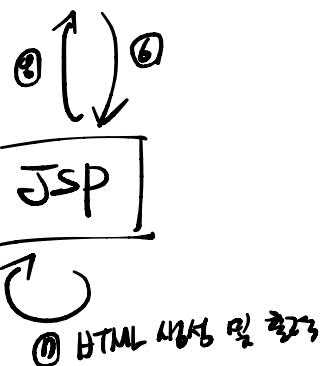


## \* MVC 2 아키텍처



View (UI) (회원 모듈)  
(다양한 화면에 출력되는 기능)

UI는 서버에게서  
데이터를 받아온다.

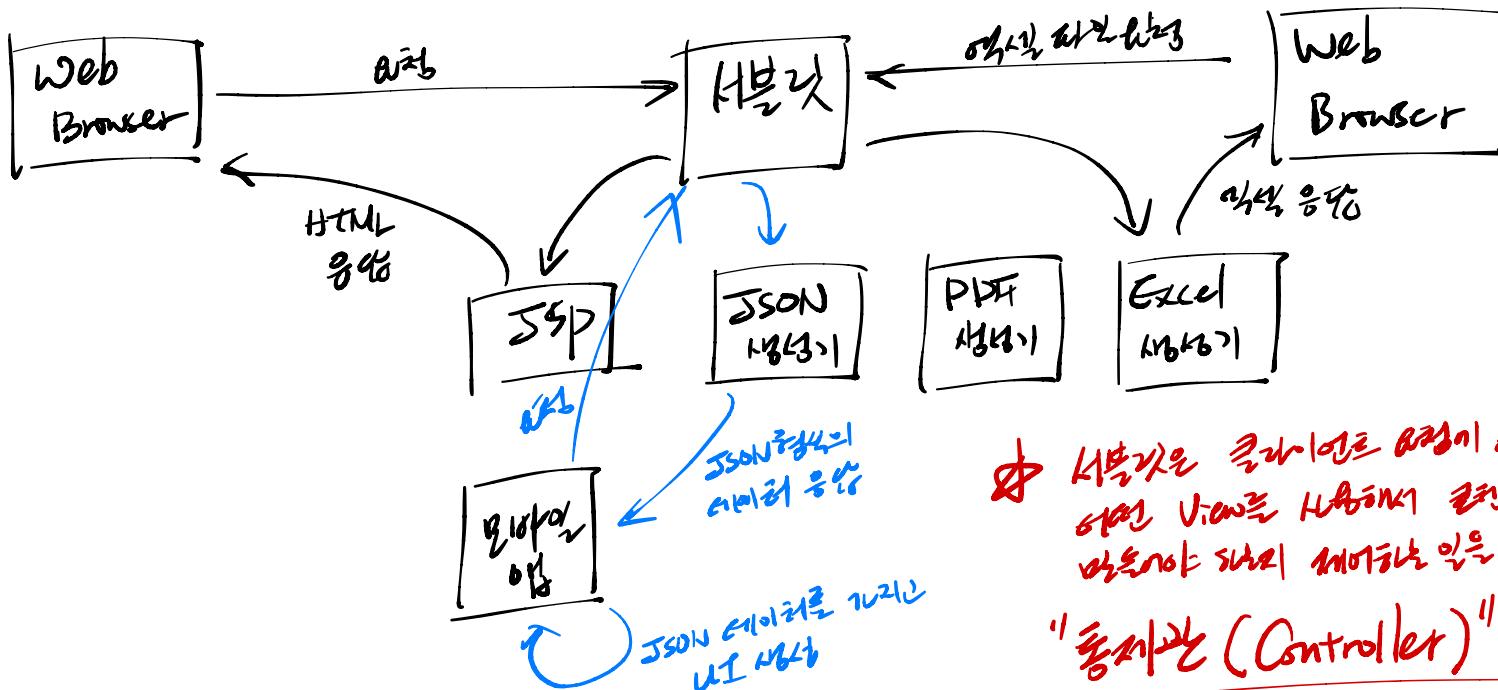


UI는 View (UI)  
View (UI)는 View (UI)

\* 서블릿에서 웹면 흐름을 통제할 때의 예는 아래

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

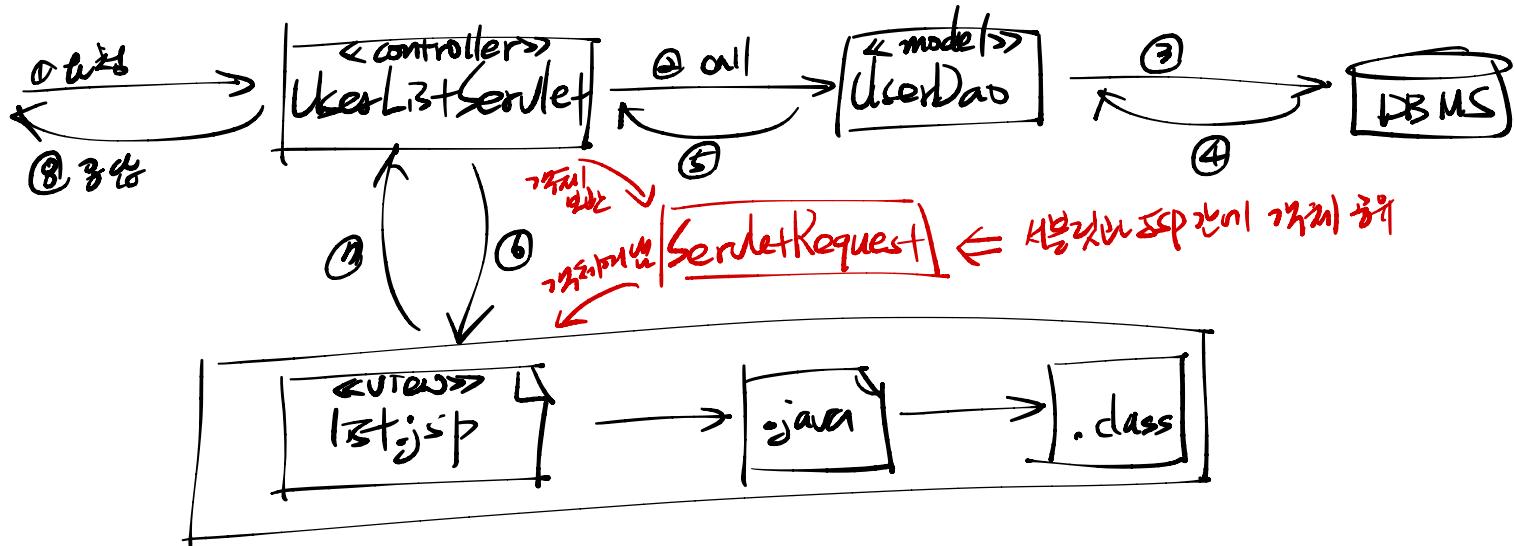
결과적으로 디자이너가 통제해 어렵다. 정부에 있는 경우 관리하기 어렵다.



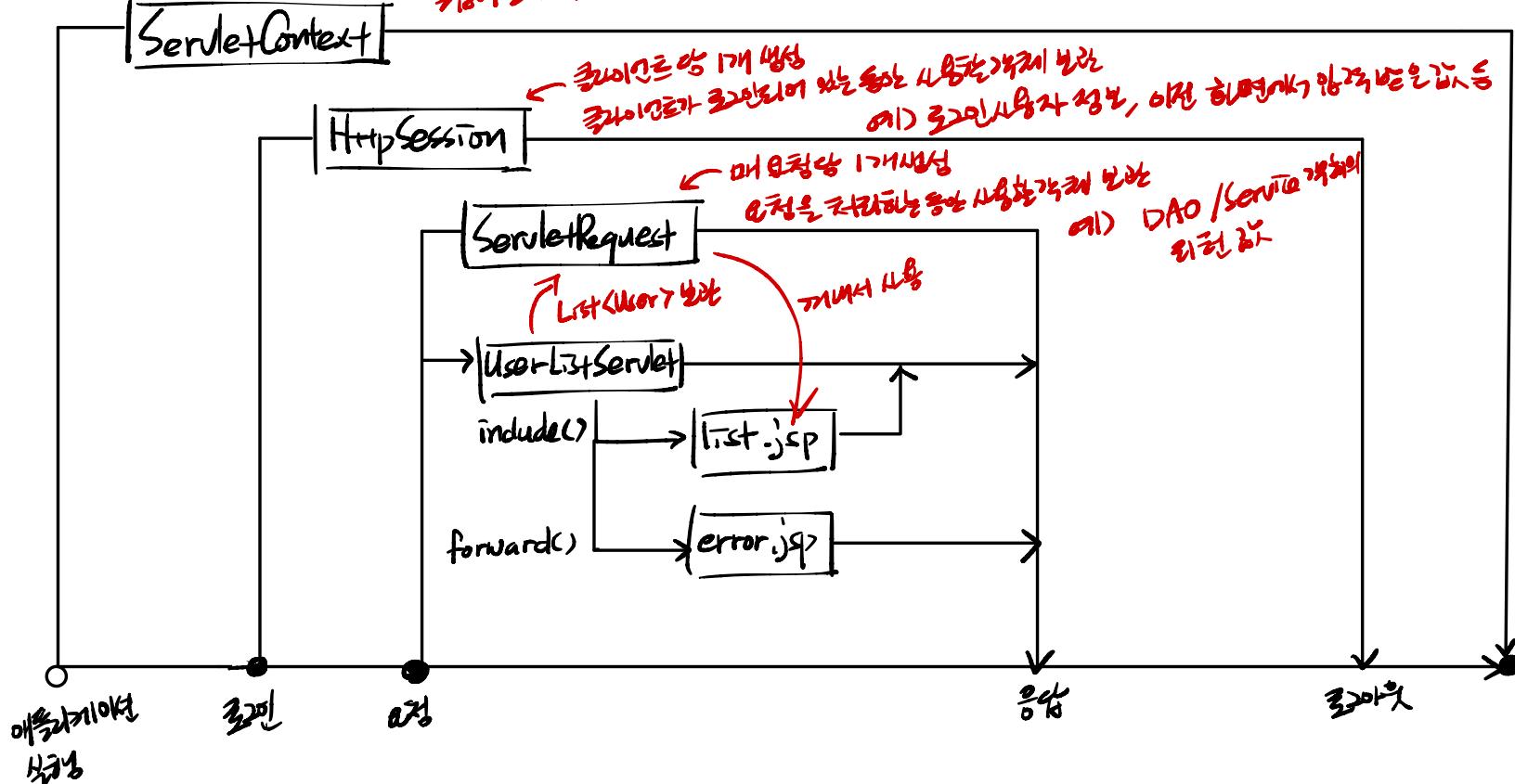
Servlet는 결과로 보낼 수 있는 다양한 View를 헤더와 함께 전송을 하도록 한다  
View는 JSON을 처리한다

"통제권 (Controller)"

\* MVC2 ဆို ဘွဲ့တော်များ

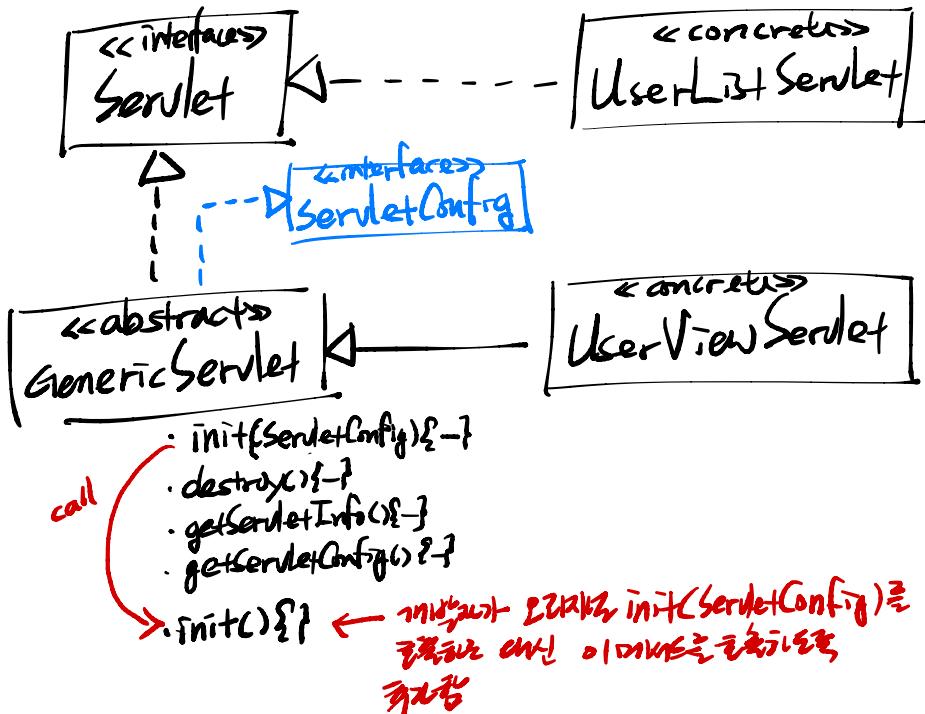


\* 원인 분석기  
원인분석기 기반 원인 분석  
원인분석기 기반 원인 분석을 위한 사용자 가정치 보관 AI) DAO, Service, 프레젠테이션 관리자 등.





\* 亂世のアーキテクチャ



• `init()` { } ★  
• `service()` { } ★  
• `destroy()` { }  
• `getServletInfo()` { }  
• `getServletConfig()` { }

• `service()` { } →