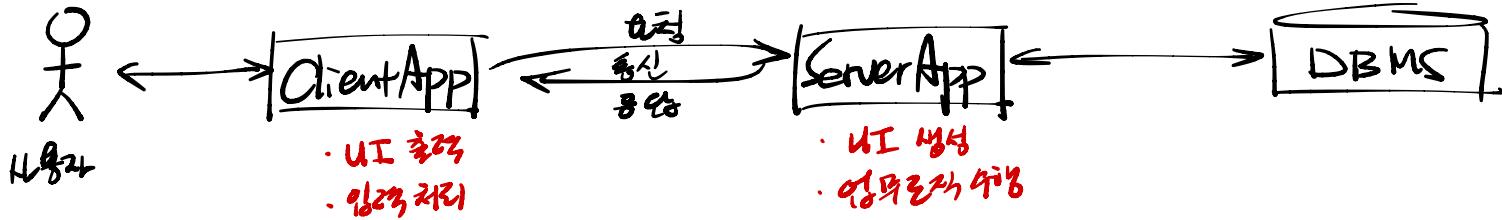
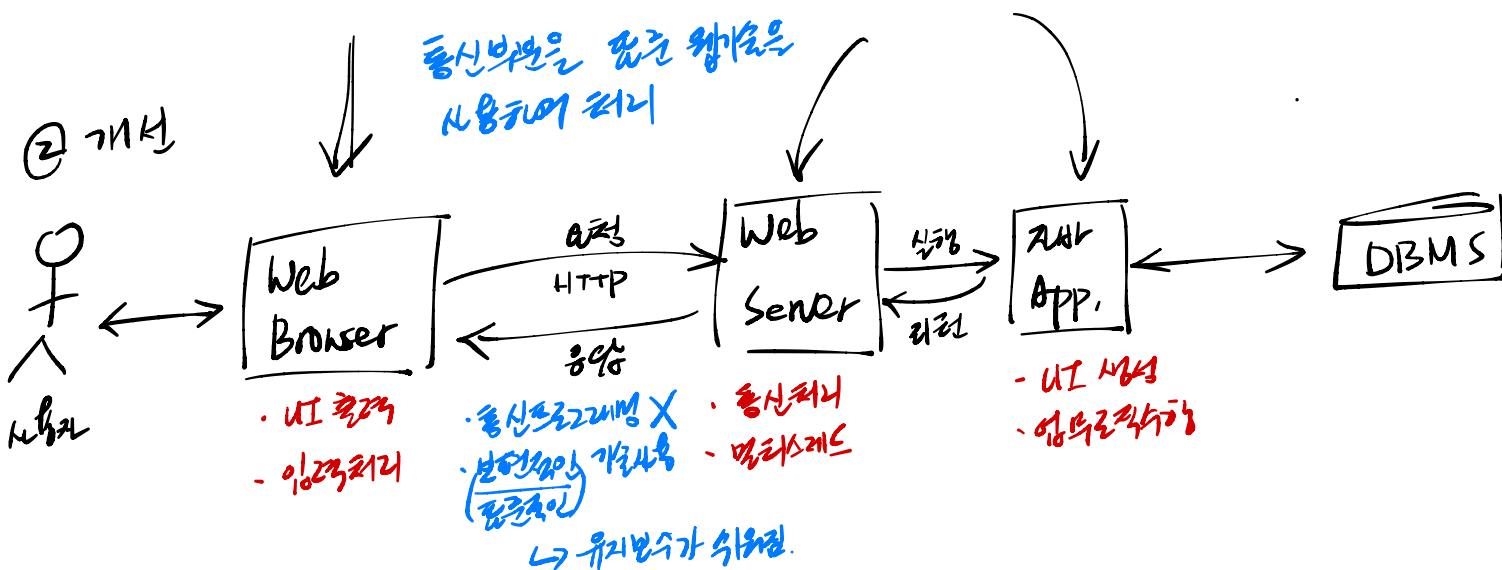


## 49. 웹애플리케이션 구조를 살펴보자

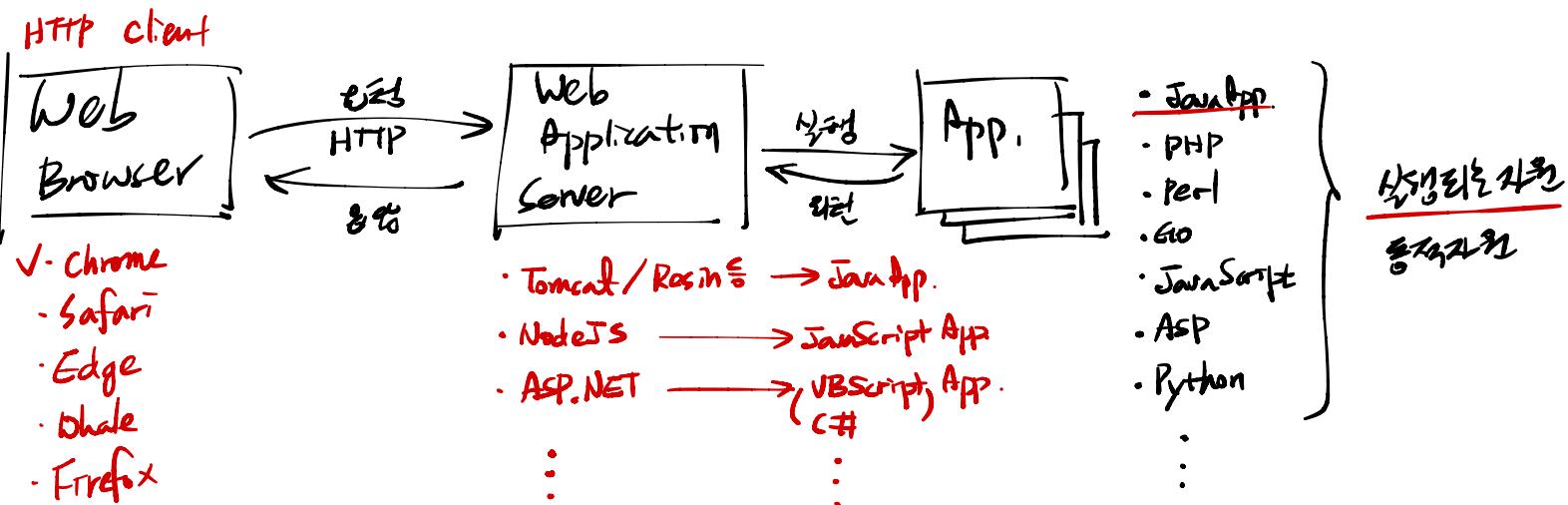
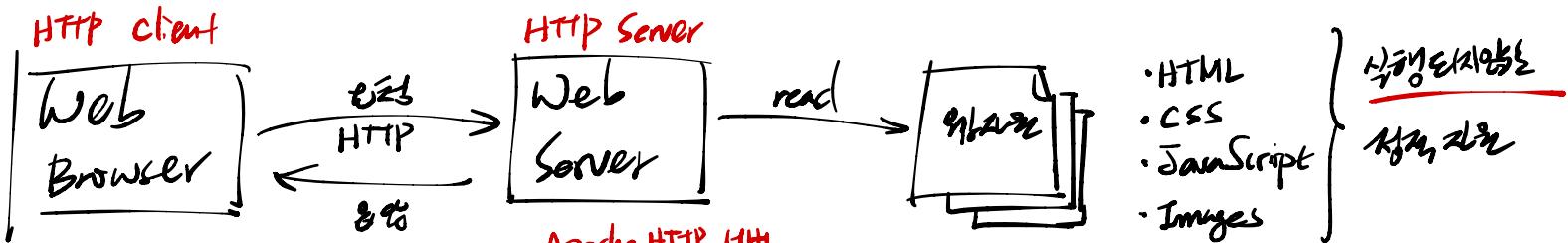
① 단계 → 전용 프로토콜을 사용하여 클라이언트/서버 통신



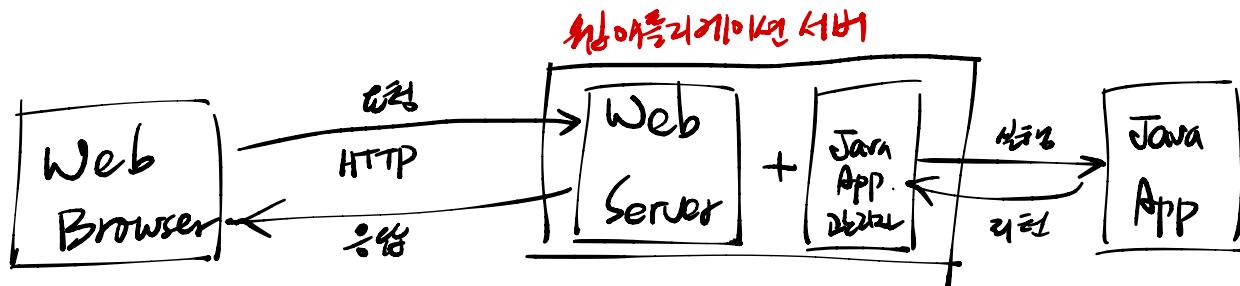
② 개발



\* 웹 사이트의 구조와 작동 원리

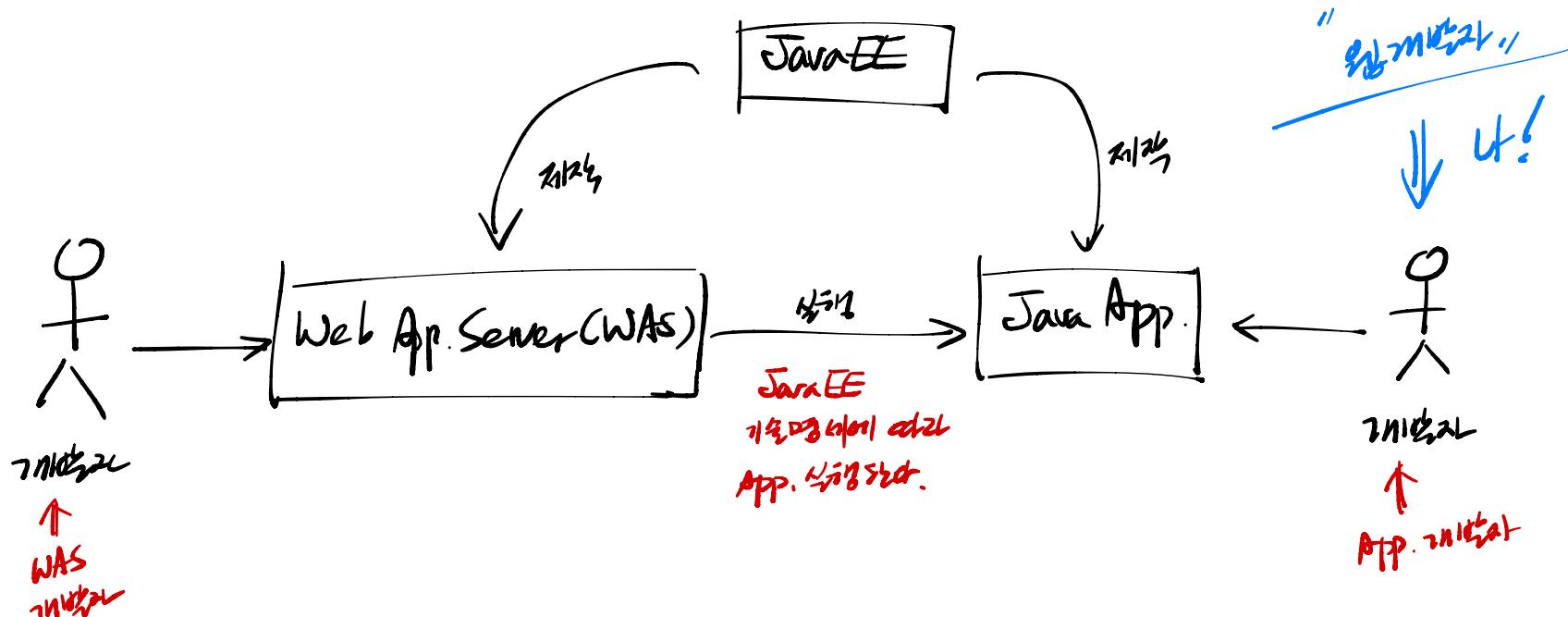


## \* Java Web Application Architecture



- Tomcat / Resin / Jetty : 웹 서버
- Weblogic
- Websphere
- JBoss
- JBoss Seam
- 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 기반 구조 기술

- 인증 및 배포 기술

## \* Java EE 7/8/10/11 버전의 특징

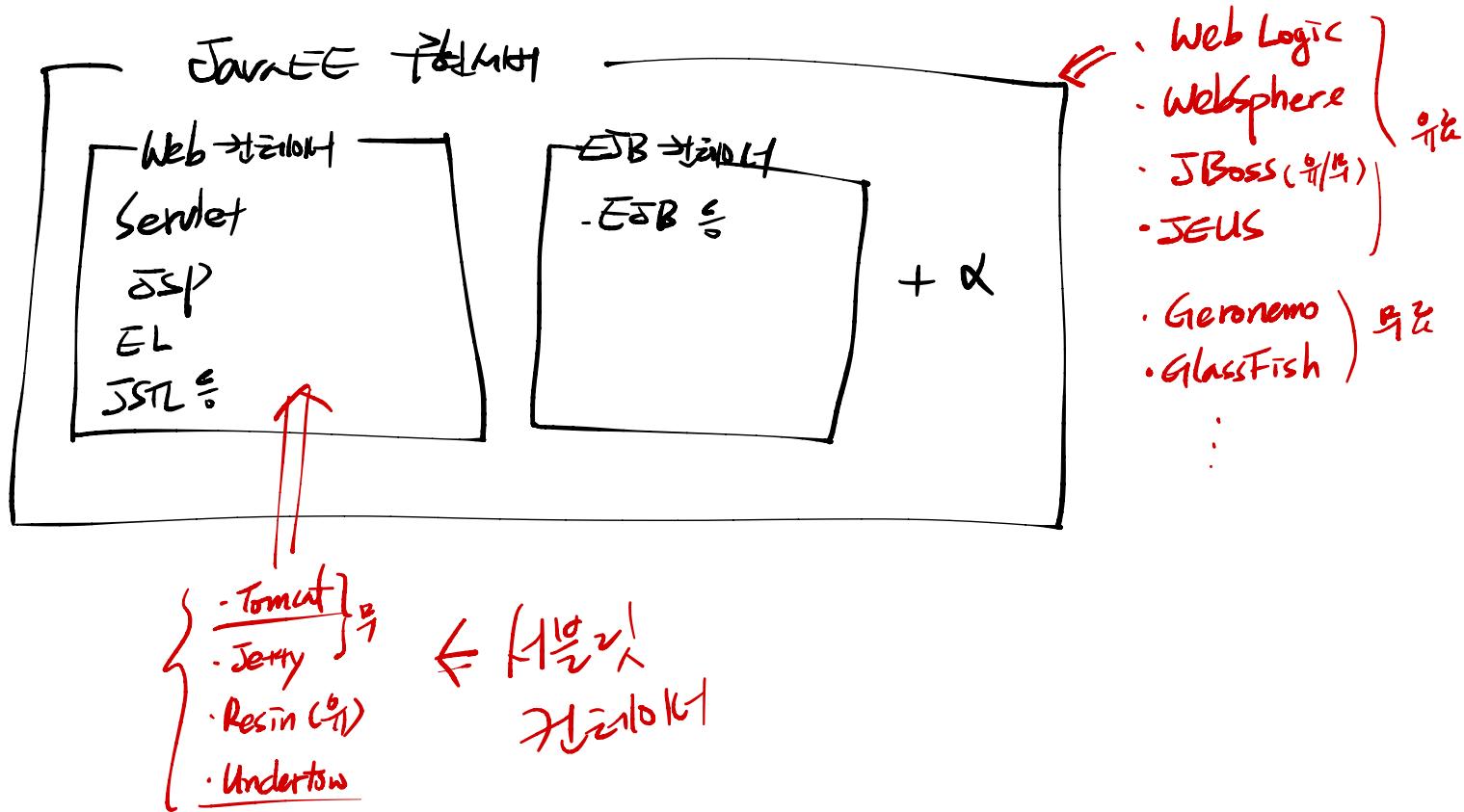
<del>JavaEE 버전</del> <del>제작년도</del>	Servlet	JSP	EL	EB
5	2.5	2.1		3.0
6	3.0	2.2	2.2	3.1
7	3.1	2.3	3.0	3.2
8	4.0	2.3	3.0	3.2

## \* Java EE 7/8 버전별 지원 현황

<del>JavaEE 6 버전</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

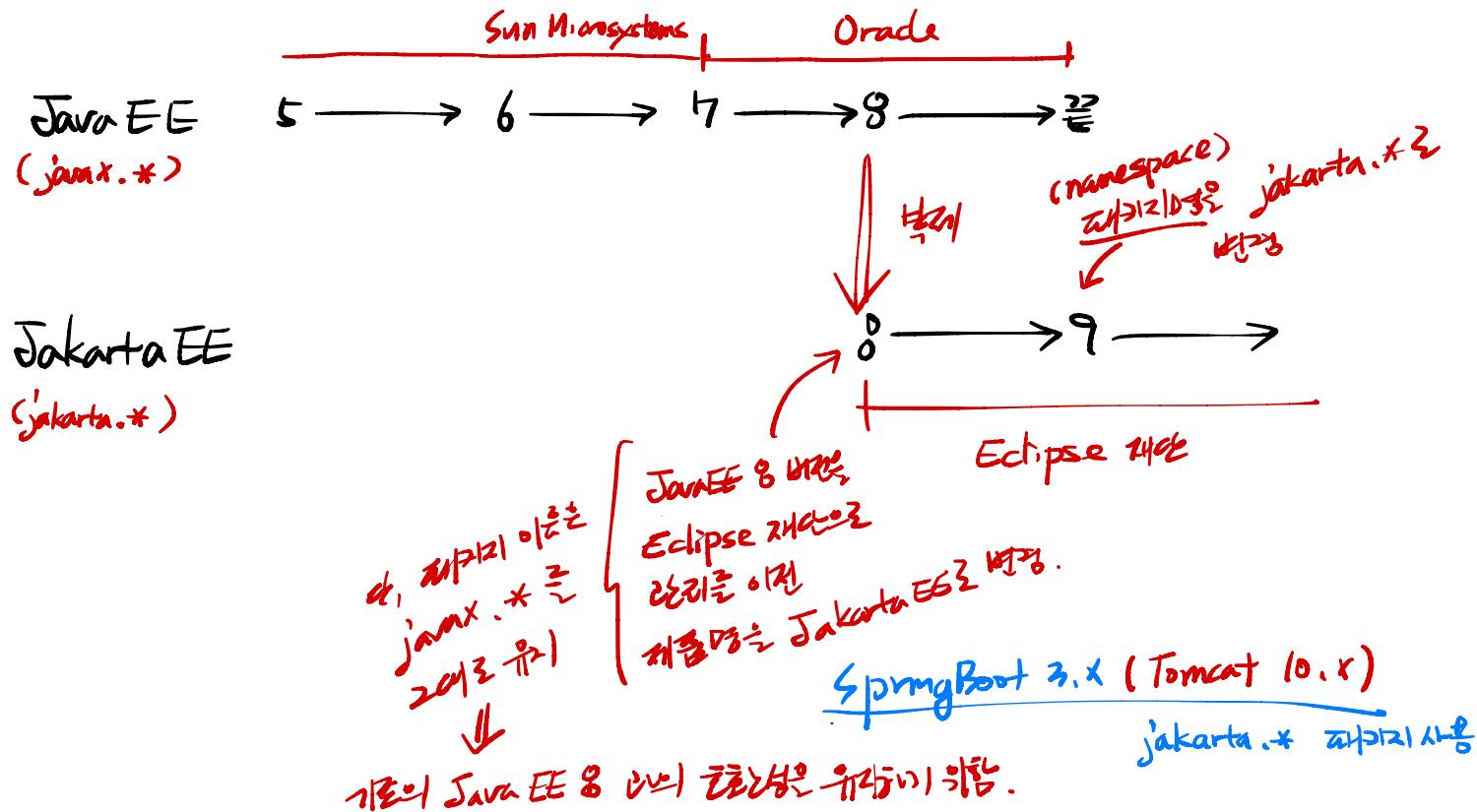
⇒ 2014년 10월 버전으로  
2014년 Java EE 버전별 지원 현황은 다음과 같다.

## \* JavaEE 페미터 서브 분류

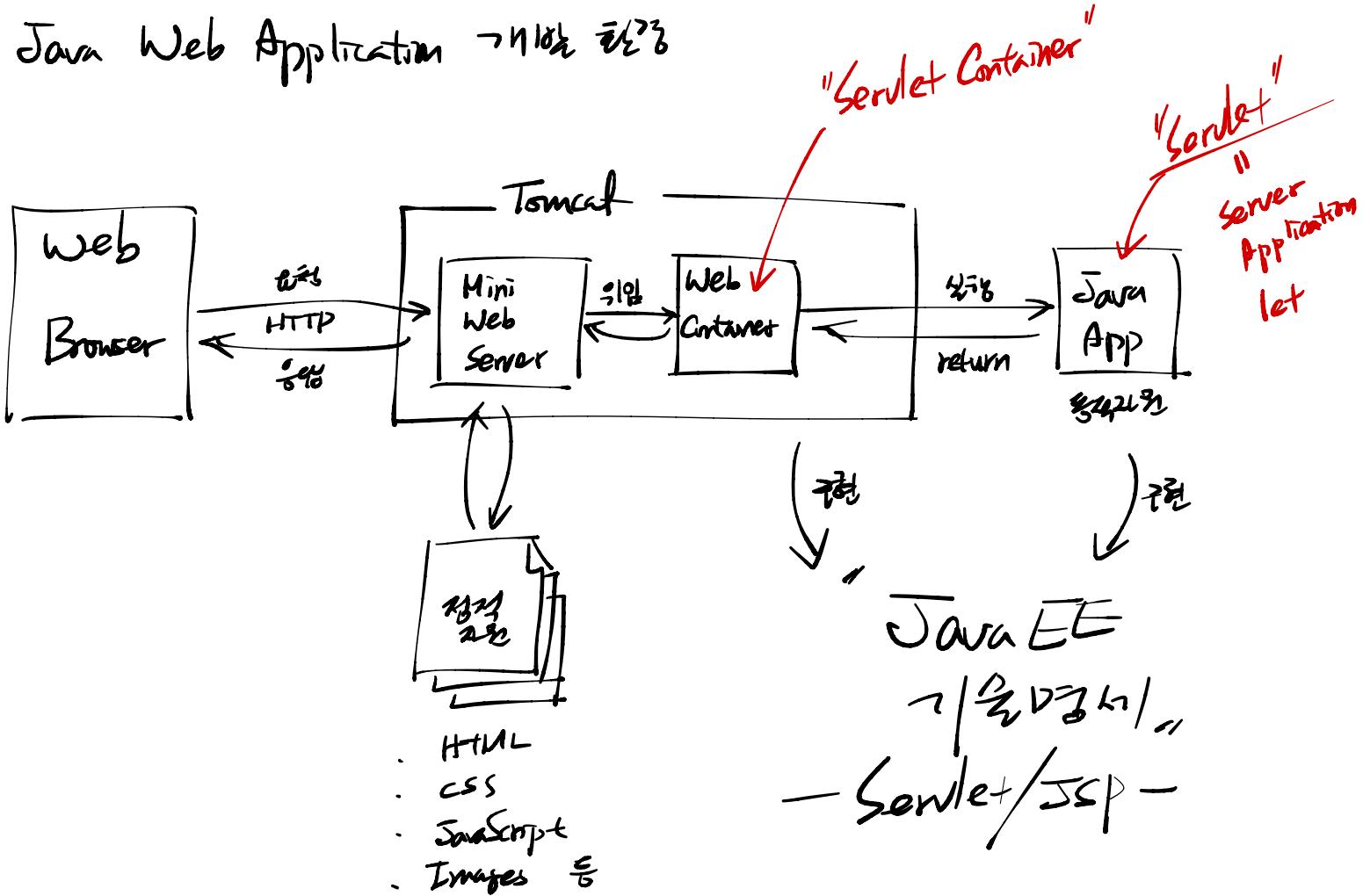


## \* JavaEE & JakartaEE

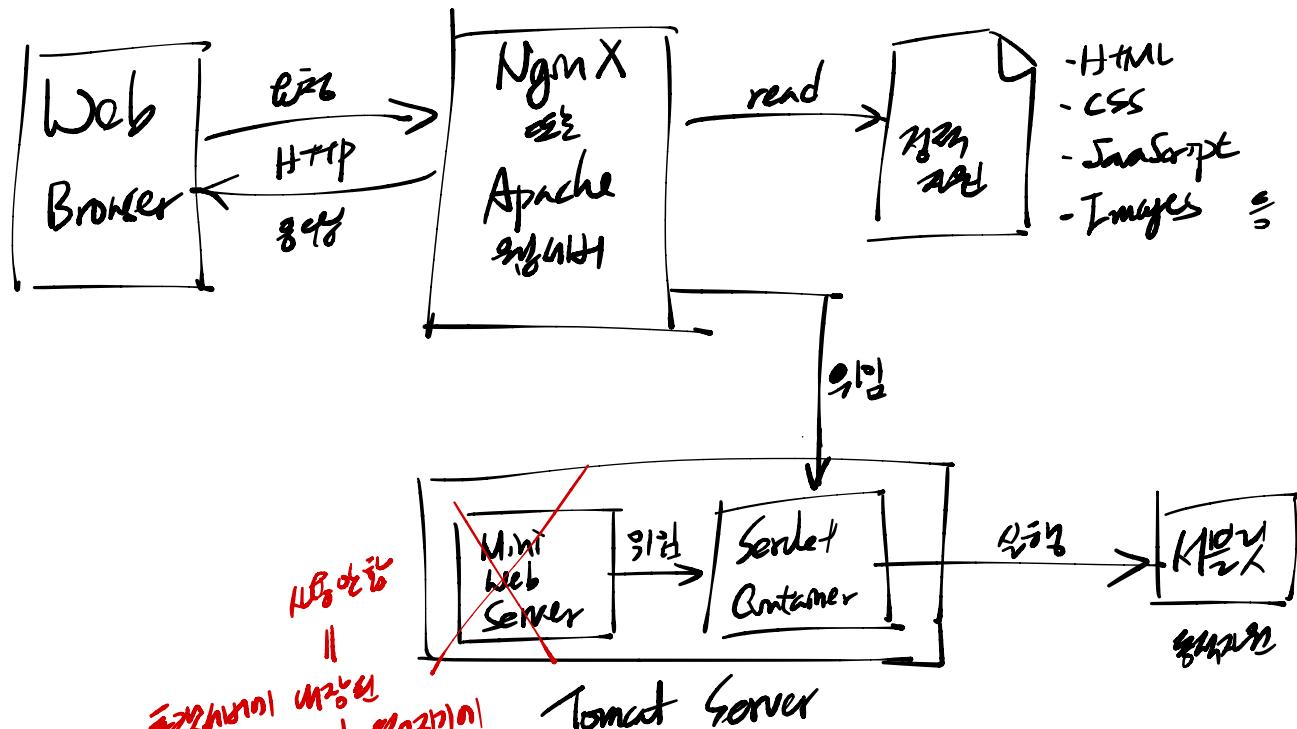
SpringBoot 2.x.x (Tomcat 9.x)



\* Java Web Application → چی یعنی



## \* Java Web Application 운영 원리



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

## \* Tomcat Server

CATALINA\_HOME/

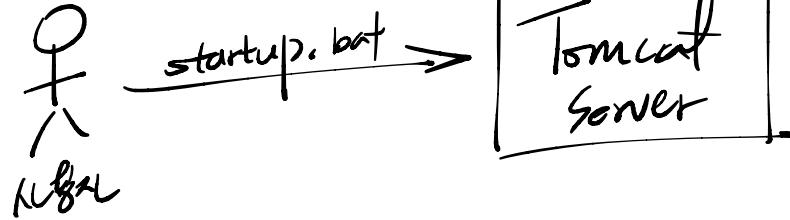
- bin/ ← 허버 실행과 관련된 파일
- conf/ ← 허버 설정 파일
- lib/ ← 허버 라이브러리 파일
- logs/ ← 허버 실행 로그 파일
- temp/ ← 허버 실행 중에 사용하는 파일을 두는 폴더
- webapps/ ← 웹 어플리케이션 폴더를 두다.
- work/ ← JSP를 Servlet 처리하기로 변환한 파일을 두는 폴더

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

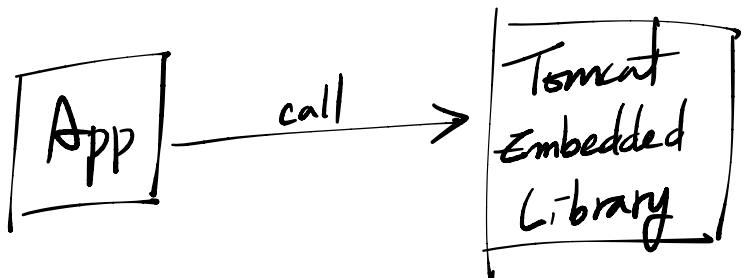
停止文件 => shutdown.bat (Windows)  
shutdown.sh (Mac, Linux)

## \* Tomcat 끊기 방식

① 재시작



② 리부트



## \* Embedded Tomcat 끝까지 써보기

new Tomcat()

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

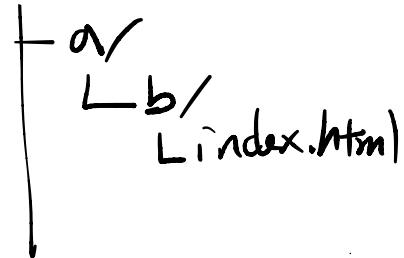
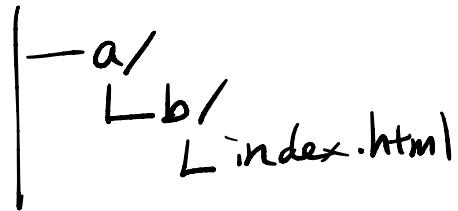
- 웹 어플리케이션 경로  $\Rightarrow$  /
  - 정적 자원 경로  $\rightarrow$  src/main/webapp/  $\leftarrow$  .html, .css, .js, .gif 등
  - 동적 자원 경로  $\rightarrow$  bin/main  $\leftarrow$  .class, .properties, .xml 등

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

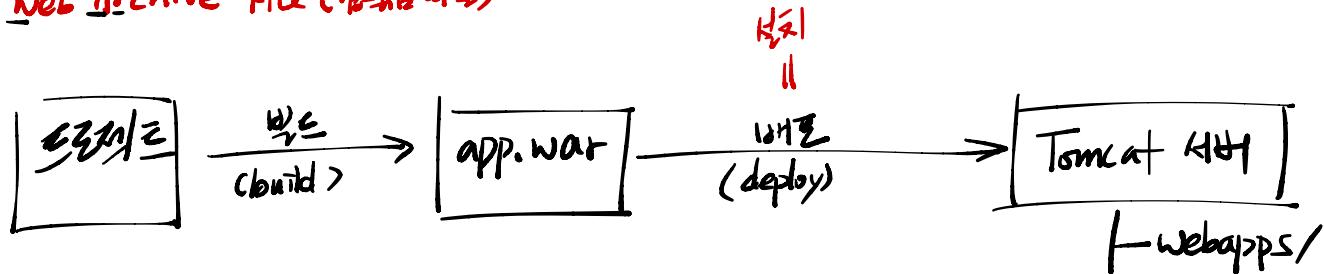
http://localhost:8888/ → src/main/webapp/



\* 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 (Deployment Descriptor File; DD File)

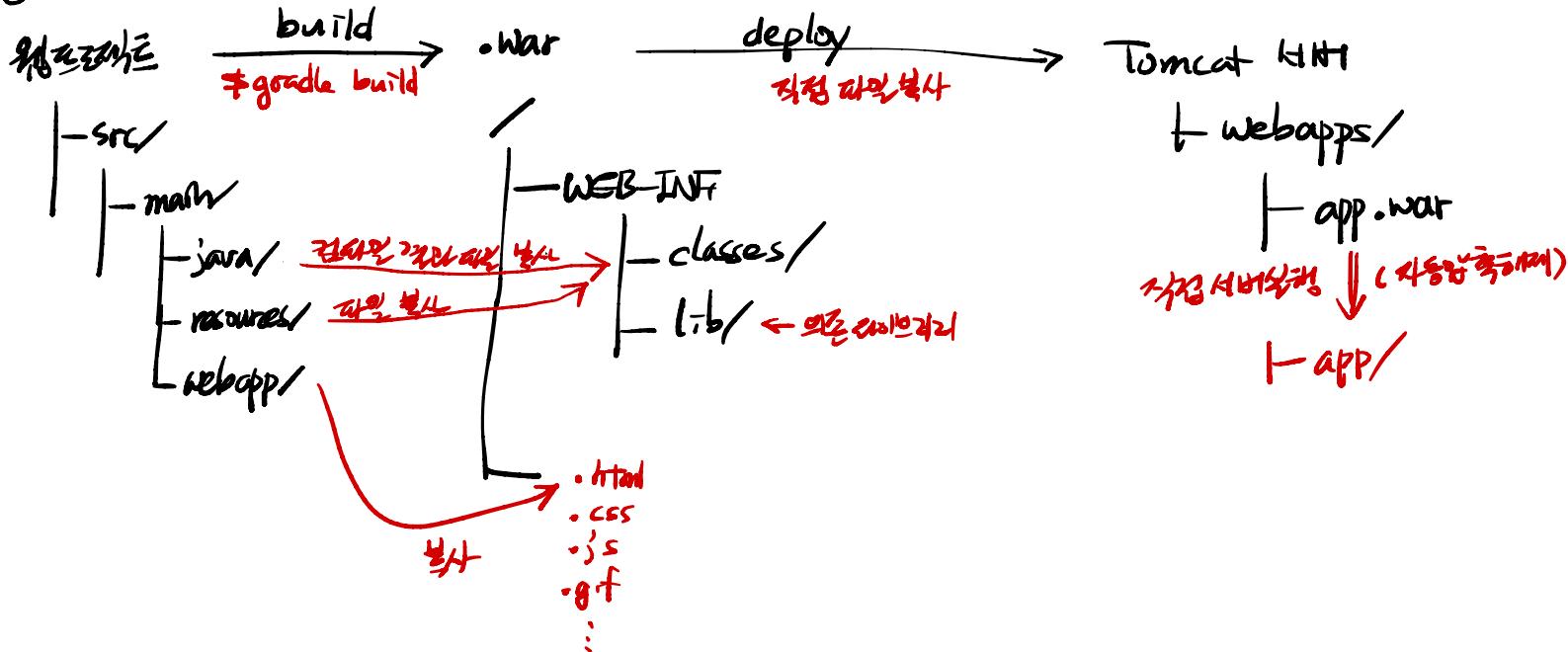
classes/ ← .class, .properties, .xml

lib/ ← .jar

.xml, .properties ← 설정파일

\* 웹프로젝트 디렉토리

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



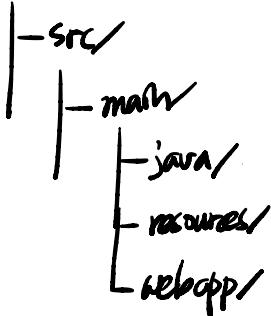
\* 웹프로젝트 파일

② Eclipse 웹 파일 <서버 설정을 적용하는 동안 서버를 파일에

생성되는

Eclipse 임시 파일 폴더

이경스 위치는/.metadata/.plugins/org.eclipse.wst.server.core/



tmp0/

- conf/ ← tomcat 설정 / conf, 폴더는 복사해온다

- logs/ ← 실행 기록 파일

- temp/ ← 일정 주기로 자동으로 파일을 두는 곳

- webapps/ ← 사용 안 함

- work/ ← JSP 변환 파일 두는 곳

- wtpwebapps/ ← 웹프로젝트 바로 폴더

생성되는/

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

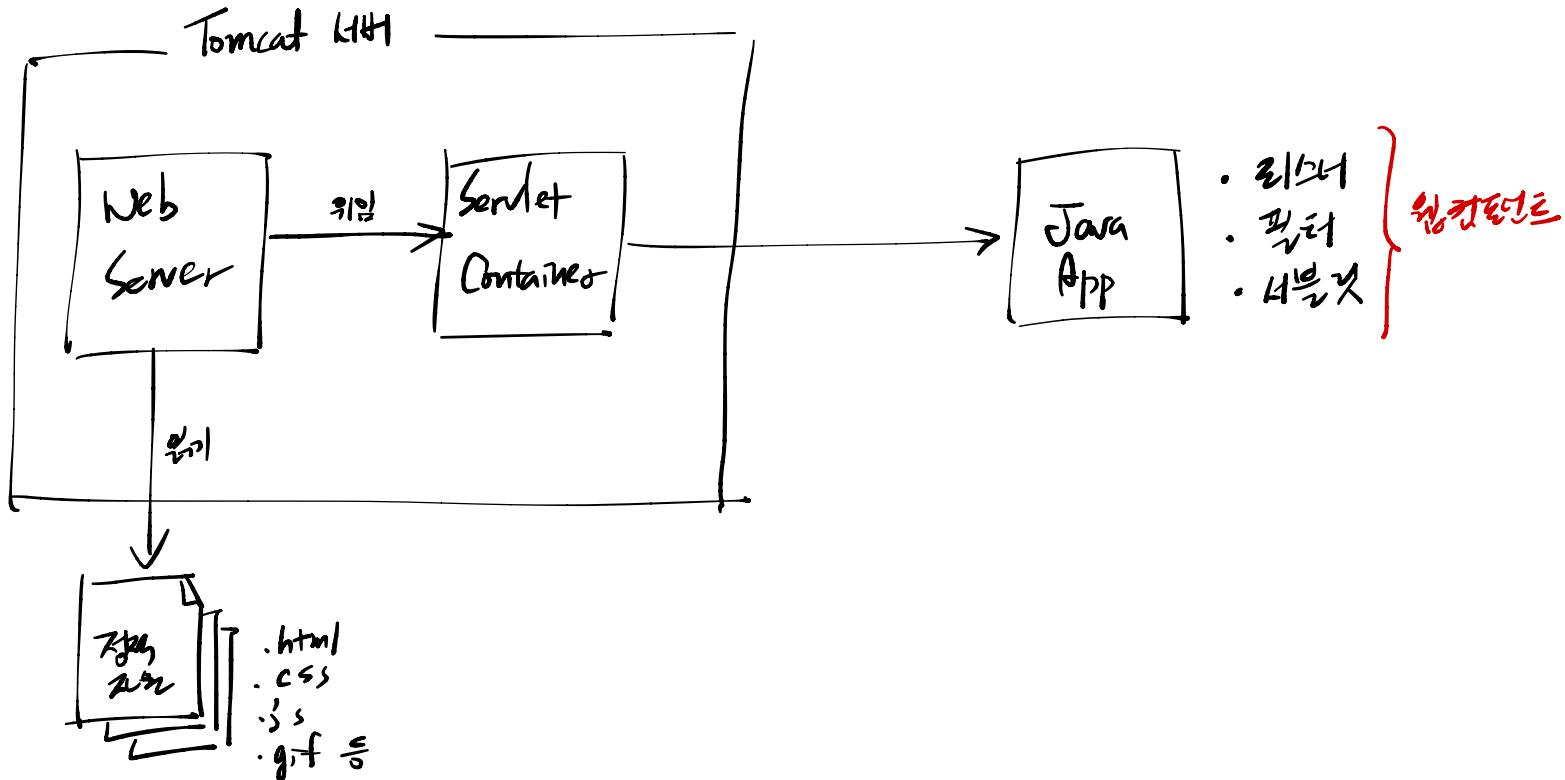
생성되는

\* 웹프로젝트 ID IntelliJ

② 웹프로젝트 + WAS 라이브리 <= SpringBoot 환경



\* 웹 컨테이너 (부록)  
↳ 한 개 이상의 기능으로 구성되어 특별한 권한을 부여받을 것. = 특별한 역할은 부여받는 기능



\* 8.7장 틀 번역 - 8.11번 = Observer  
Grati

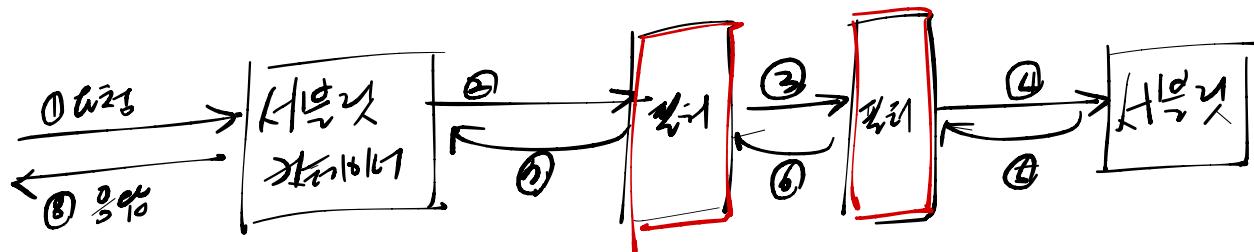
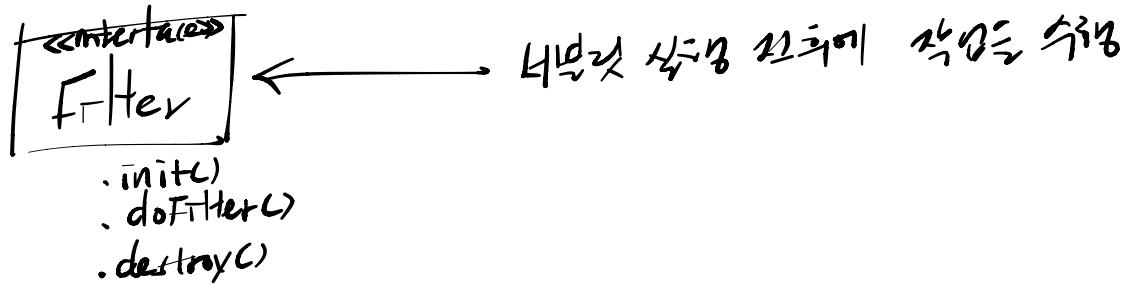
ServletContextListener ← 서버가 컨텍스트가 제작되거나 종료될 때 호출되는  
contextInitialized() contextDestroyed()

ServletRequestListener ← 요청이 제작되거나 제거될 때 호출되는  
requestInitialized() requestDestroyed()

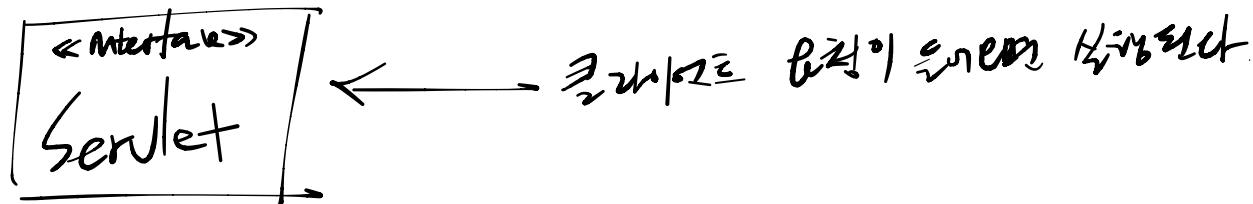
HttpSessionListener ← 클라이언트가 생성되거나 종료될 때 호출되는  
sessionCreated() sessionDestroyed()

:

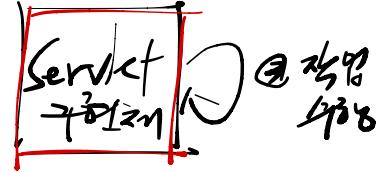
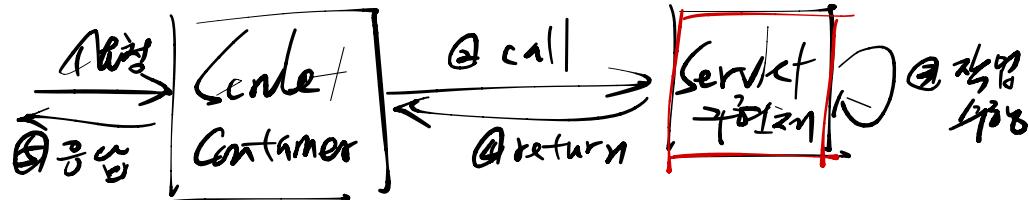
\* 필터 구조화 - 책임 = chain of Responsibility  
 GOF의 책임 체인



\* 웹 개발언어 - HTML = Command  
 GOF의 디자인 패턴



- . init()
- service()
- . destroy()
- . getServletInfo()
- . getServletConfig()



\* 풀不尽 서버로 진화하여 등장

① 웹서버 → ② 애플리케이션 서버

Server App

- 파일통신구현
- 서비스레이팅 구현
- 전송통신망
- 웹서버 구현

Server App

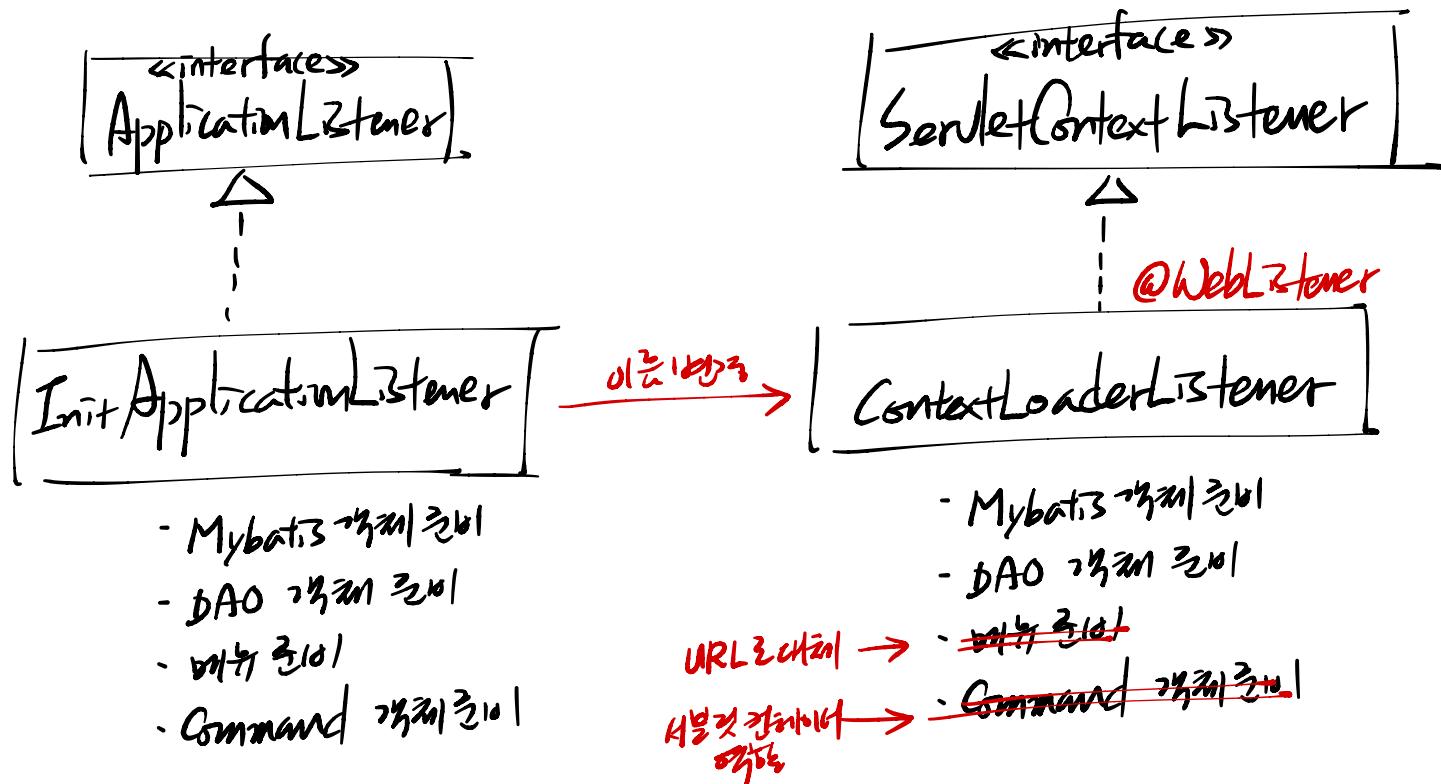
- ~~파일통신구현~~
- ~~서비스레이팅 구현~~
- ~~전송통신망~~
- ~~웹서버 구현~~

Web

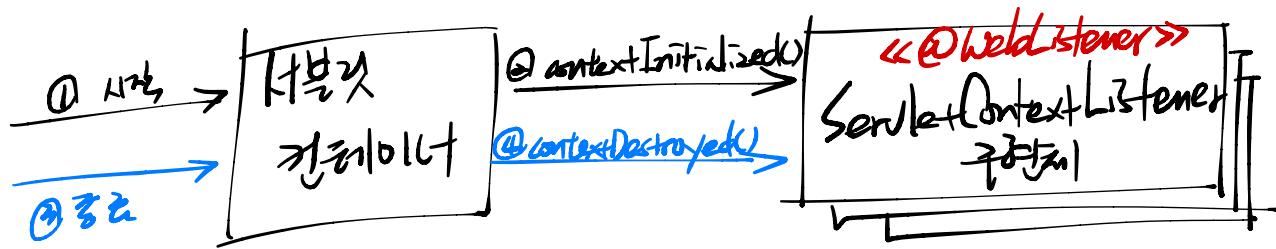
Tomcat

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

\* 리스너 만들기



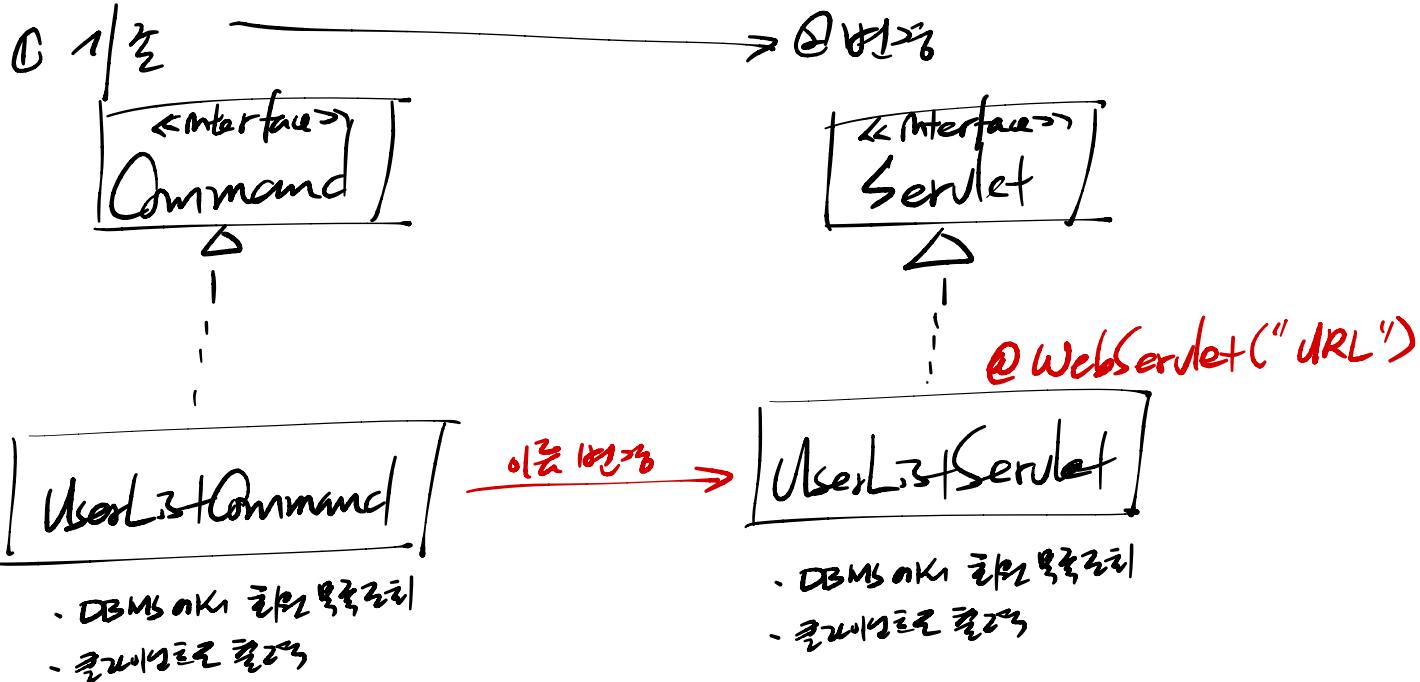
## \* ServletContext Listener



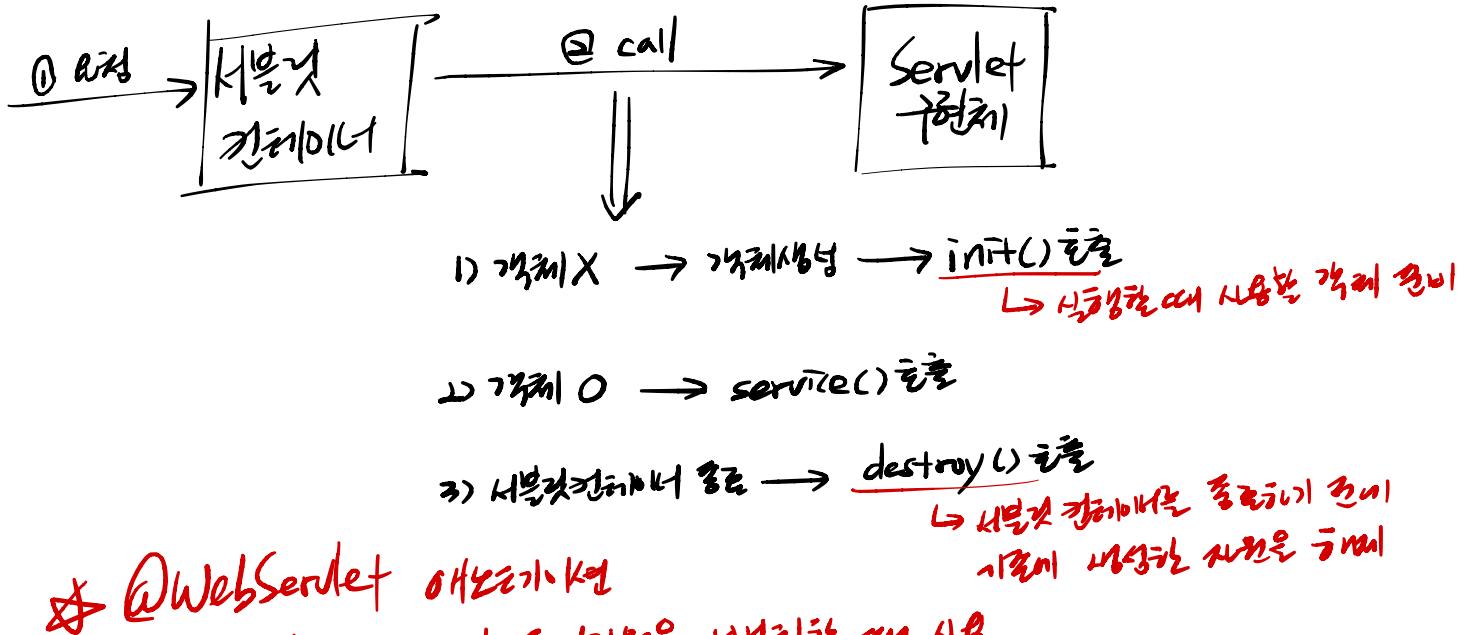
\* **@WebListener** effect하기

- 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.

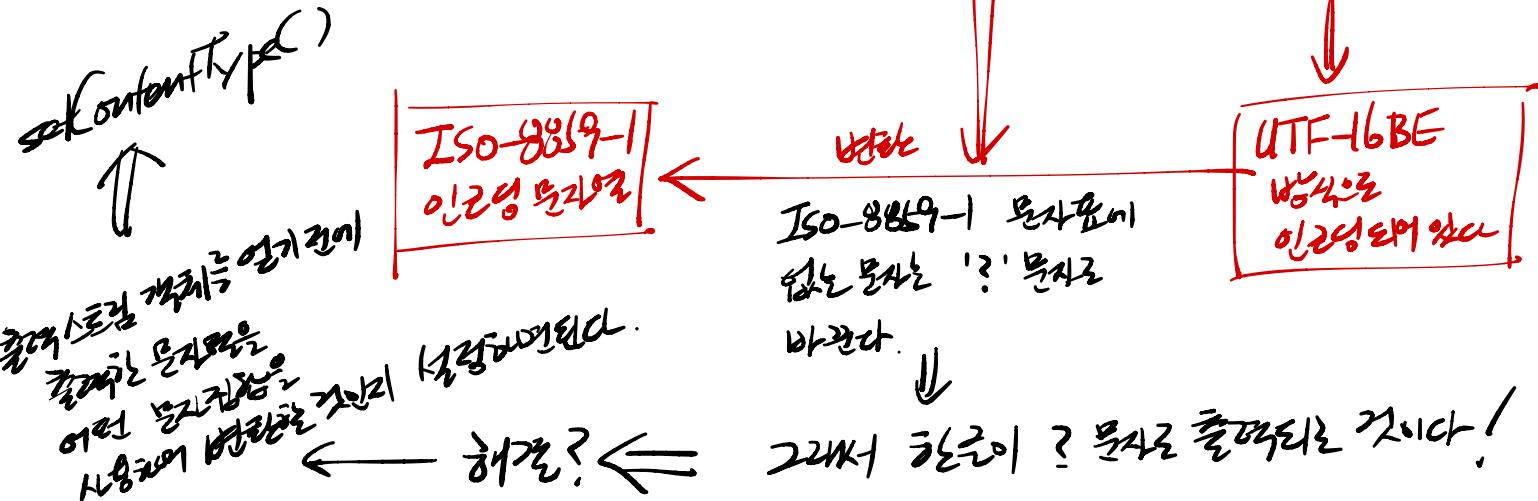
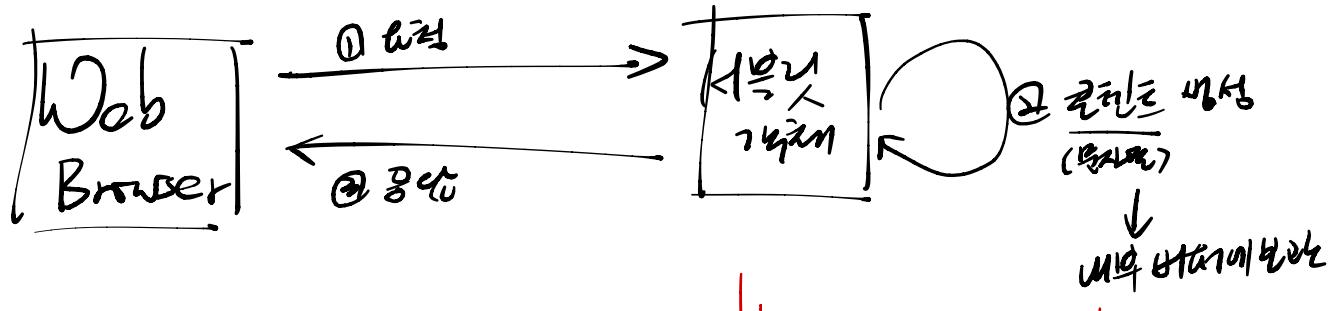
\* service() method

service(ServletRequest req, ServletResponse resp) { }

- ✓ Request pass thru
- ✓ Response thru Method call

• Request thru Method call

\* 문자열 출력하고 읽을 때 깨짐



\* 한글이 흐赡한 문자열에서 한글이 깨지는 예

한글

"ABC??"



442433F3F

한글

한글

"ABC가?"



004100420043 AC00AC01 (UTF-16BE)

ISO-8859-1 문자셋 (256자)

A → 41  
B → 42

a → 61  
b → 62

⋮  
o → 30  
i → 31

⋮

\* UTF-16BE vs UTF-16LE  
" "(UTF-16)

'f' 0xAC00 0x00AC  
'A' 0x0041 0x4100

\* 한글기호가 쓰여 있는 문자열이나 bytes의 경우에 대해 인코딩이 필요

한글인코딩

"ABC가되"



41 42 43 EAB080 EAB081

한글인코딩 bytes

EAB080 41 42 43

setContentType("text/plain; charset=UTF-8")

bytes

UTF-8 문자코드

A → 41

B → 42

:

가 → EAB080

나 → EAB081

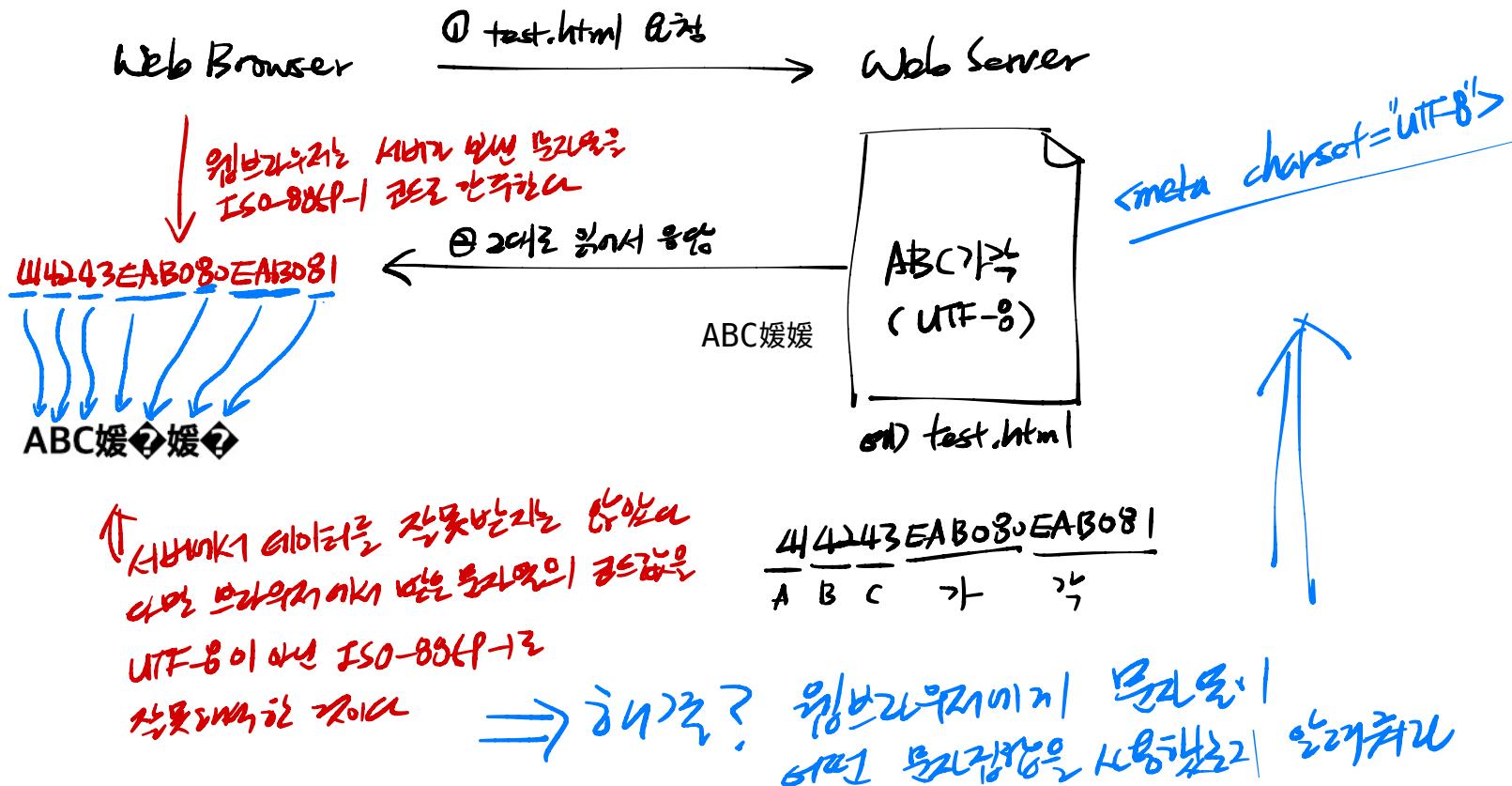
한글

"ABC가되"

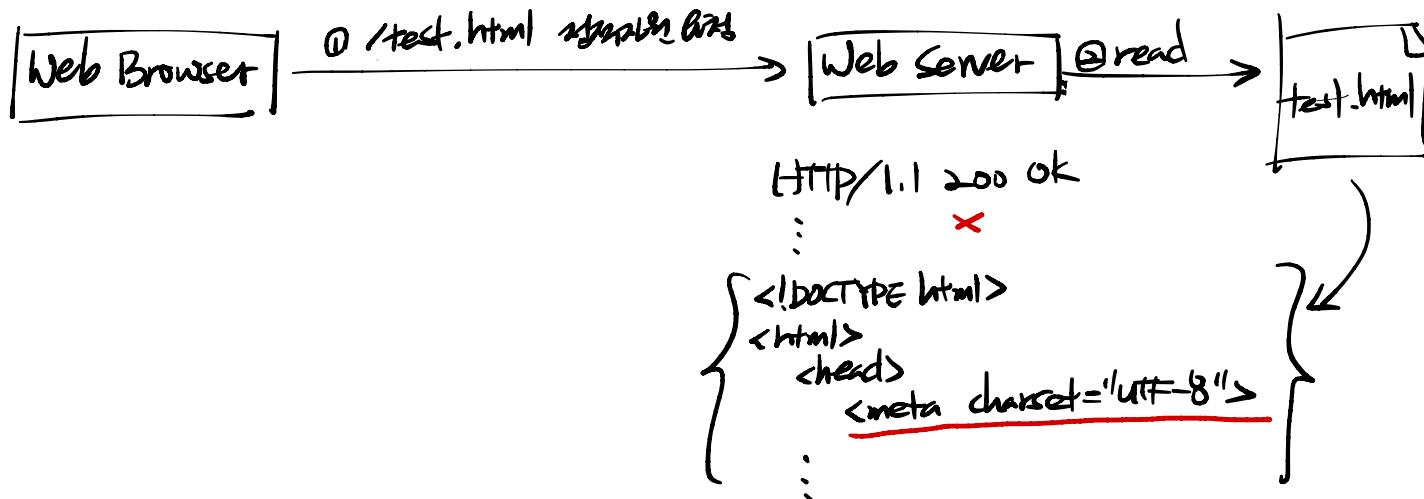
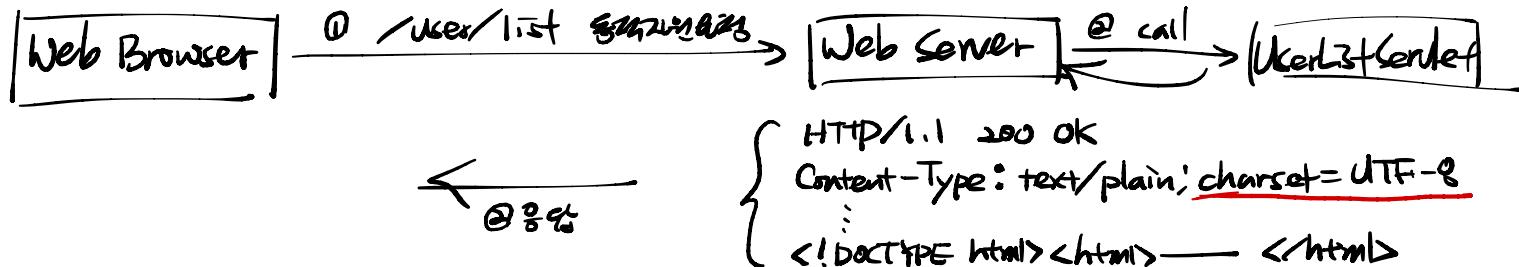


004100420043A000A001 (UTF-8)

## \* HTML 문서의 한글 깨짐



\* សេវាទូរសព្ទ



\* setContenttype()

Multi-purpose  
Internet  
Mail  
Extension

MIME Type  
설정방법

이메일로 전송되는 경우로  
문서의 확장자와 일치하지 않아  
문서형식  
→ 파일은 미리워크로  
작은 파일로  
파일을 사용하는 경우  
문서형식을 사용하는 경우로  
선택.

setContenttype("MIME 타입; charset");

① MIME의 인터넷 기본

MIME Type 형식

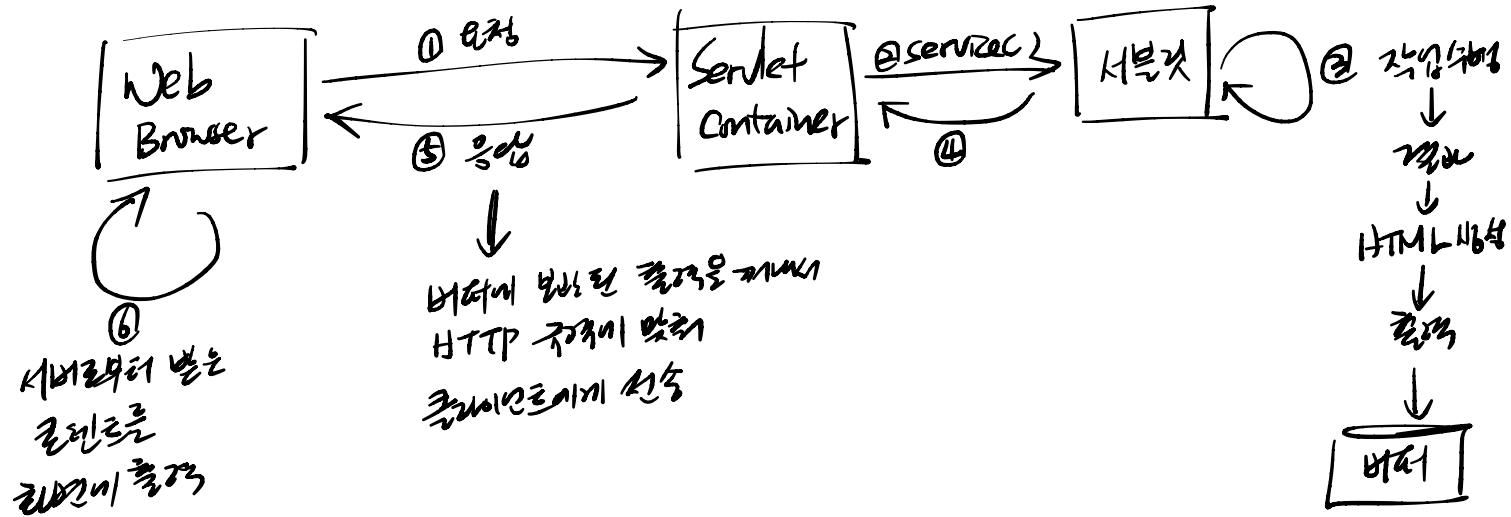
↓  
type/sub-type ; charset = 값

ex) "text/plain; charset=UTF-8"

↓

UTF-16BE → ~~ISO-8859-1~~  
UTF-8

\* Web Browser et HTML  
 ↳ 웹 클라우드를 제공하는 웹서버 = 브라우저 + 웹서버 + 서버 + DB



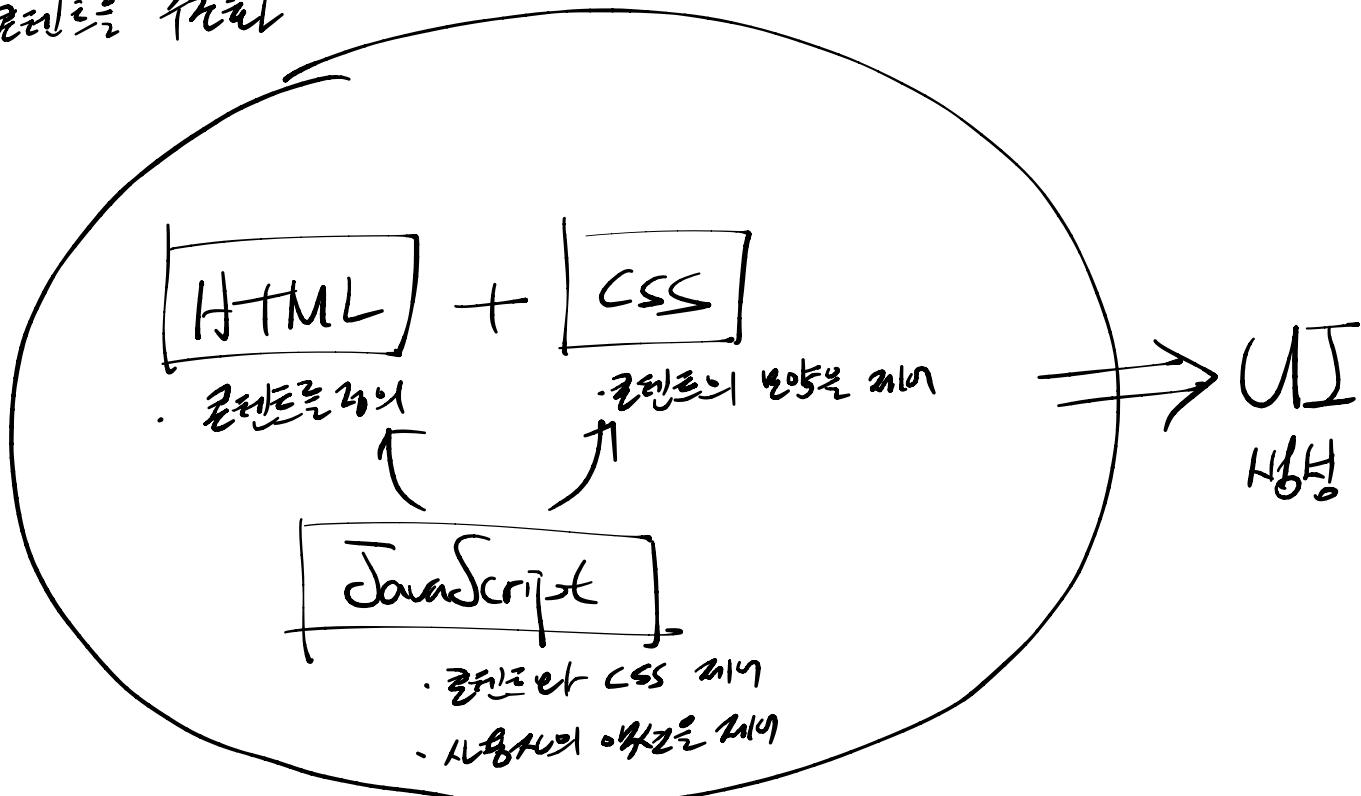
text/plain → 2013 323

text/html → HTML 렌더링

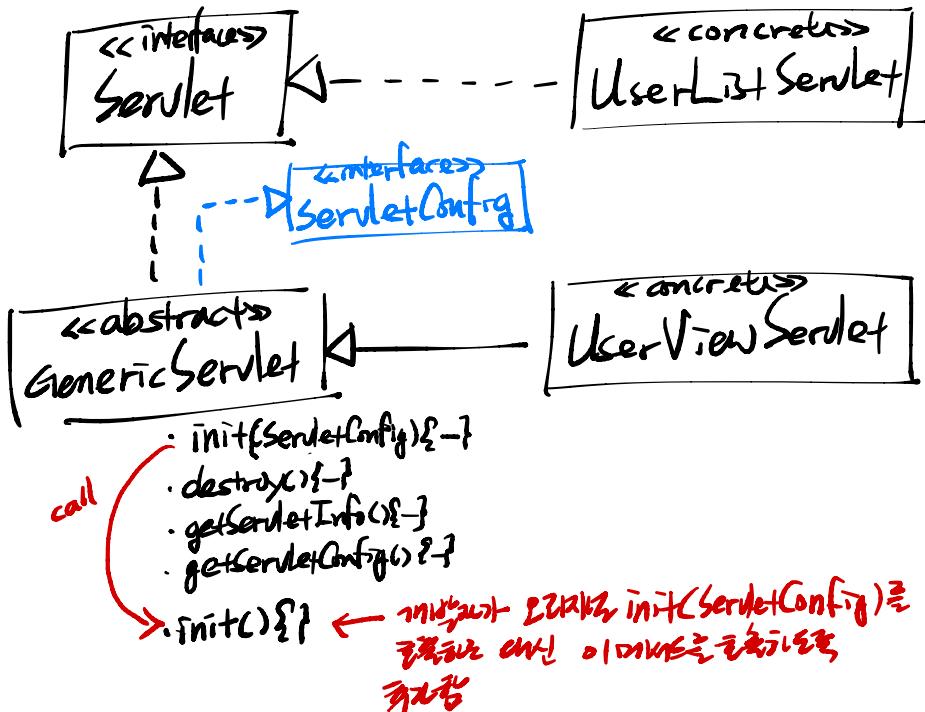
기타 → 파일

## \* HTML (Hyper-Text Markup Language)

↳ 원래는 주제



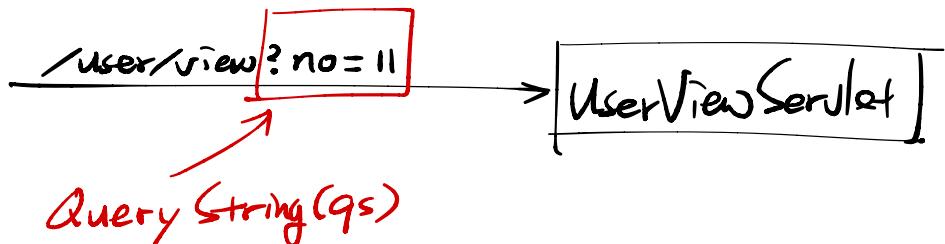
\* 亂世のアーキテクチャ



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

• service() ↗

\* URL 주소의 쿼리 문자열 - URL이 표현하기 편한



? no = 11

↑      ↑  
parameter name    parameter value

ServletRequest.getParameter("no")

return  
"11" 문자열

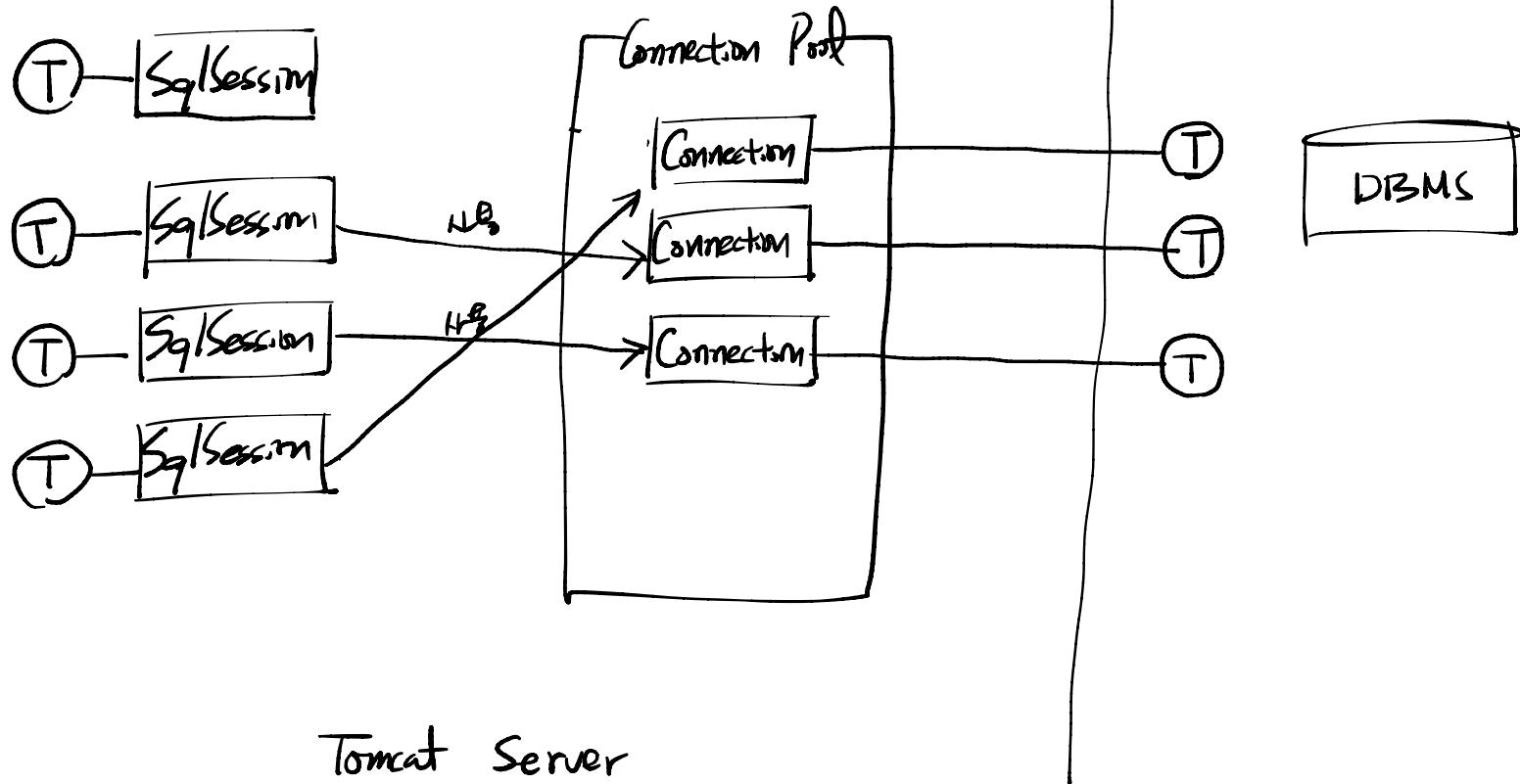
\* HTML 문서에서 다른 자원을 요청하는 경로 주소

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

HTML reference

→ 템플릿 엔진  
→ 컨트롤러  
→ 서비스  
→ 뷰  
→ 편집된다.

## \* DBMS et SqlSession

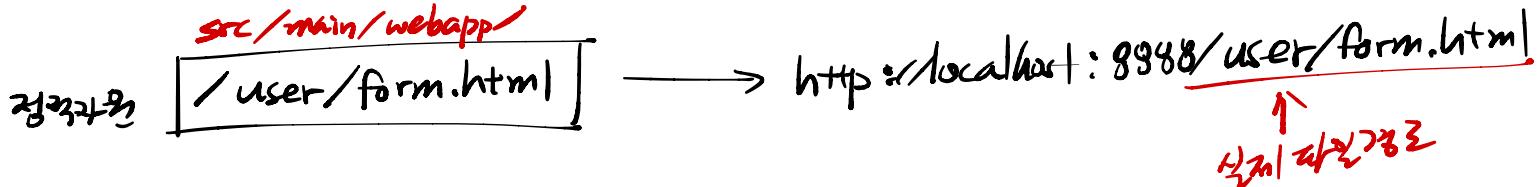
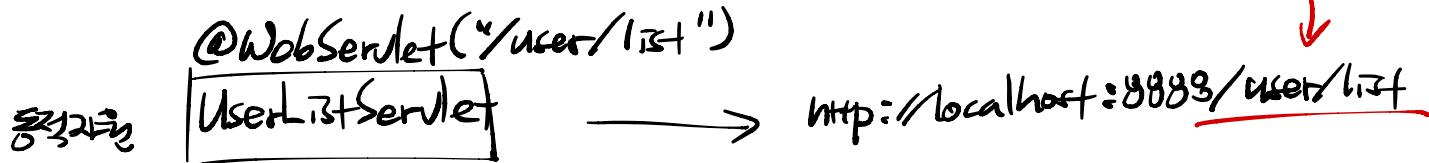


## \* URL (Uniform Resource Locator)

Identifier  
URI

- URL ex) `http://www.google.com`
- URN ex) `urn:isbn:123456`

统一资源定位符  
全局唯一标识符



全局唯一标识符

## \* URL Encoding = Percent(%) Encoding

↪ URL 을 작성할 때 URL에서 특별한 의미를 지니는 문자를 URL에서 사용할 수 있게  
기호에 따라 변환해야 한다.

RFC-3986에 의해 URL을 작성  
(기준)

일부 문자를 특별한  
의미로 사용하는  
기호

Reserved keyword → URL에서 특별한 의미로  
사용된다.

!	%21	;	%3B
#	%23	=	%3D
\$	%24	?	%3F
&	%26	@	%40
,	%2C	[	%5B
)	%29	]	%5D
*	%2A		
+	%2B		
,	%2C		
/	%2F		
:	%3A		

Unreserved characters

A-Z a-z 0-9 - . ~

한글 ?  
한국어 → %xx / %xx

"ABC가?" → ABC%EA%BD%80%CA  
EAB080 EAB081  
%BD%81

\* Query String یعنی

Query String

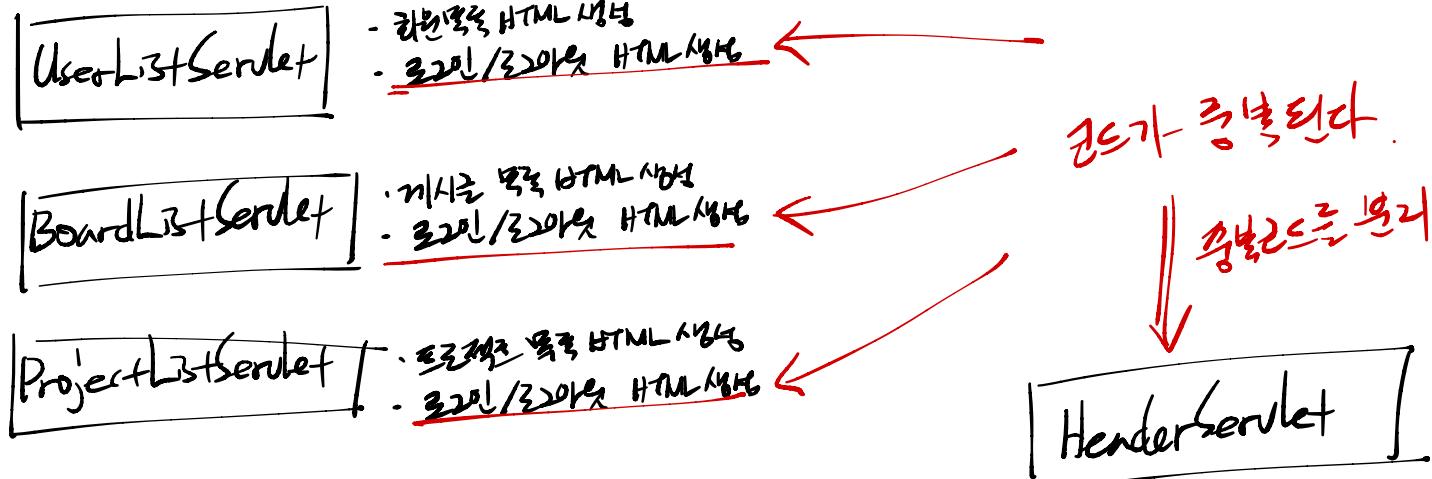
http://localhost:8888/user/add?name=ABC&email=bbb%40test.com&password=&tel=ddd

ServletRequest.getParameter("name")

"ABC"

\* 서블릿 include → 다른 서블릿의 실행 결과를 포함시키는 방법

## ① 예제



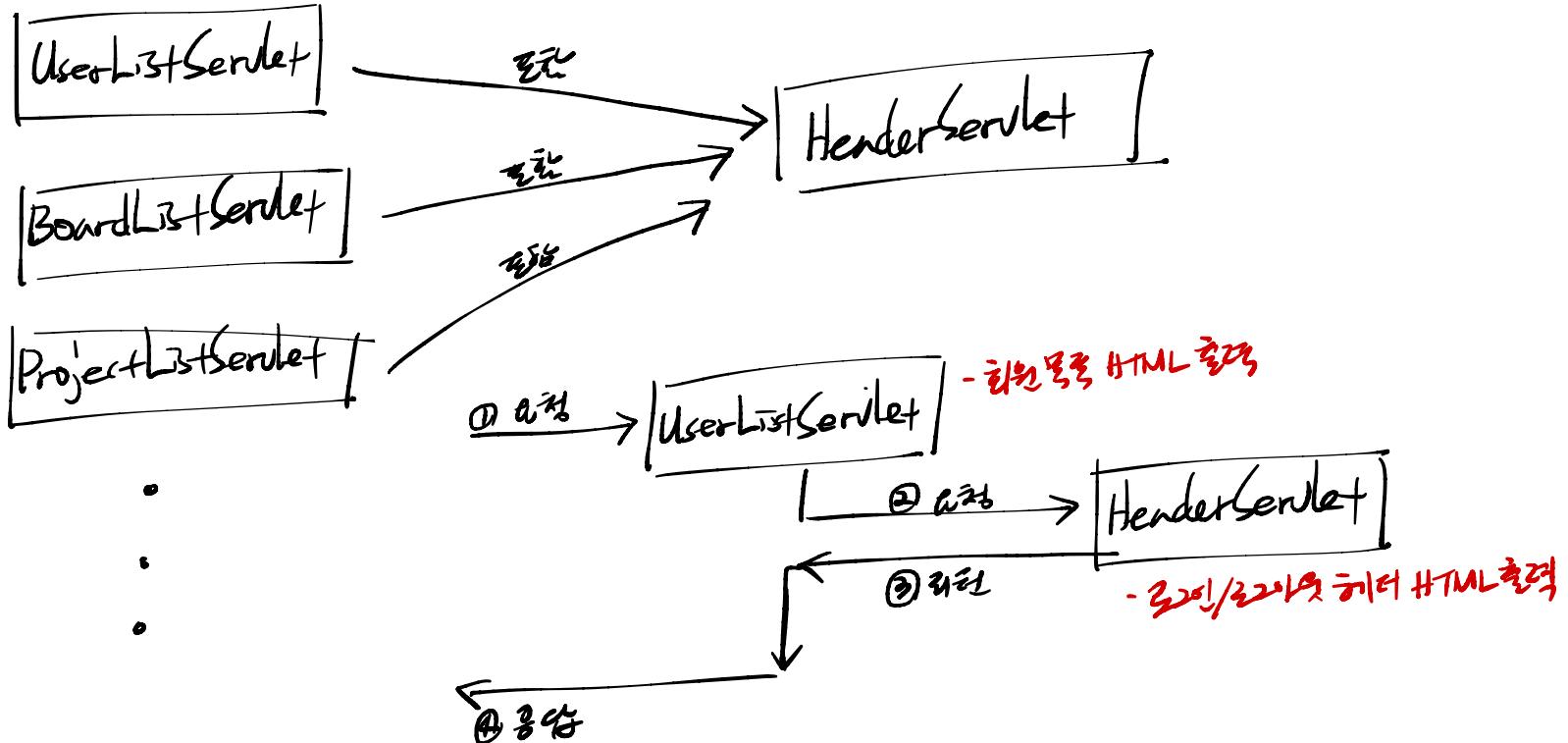
•

•

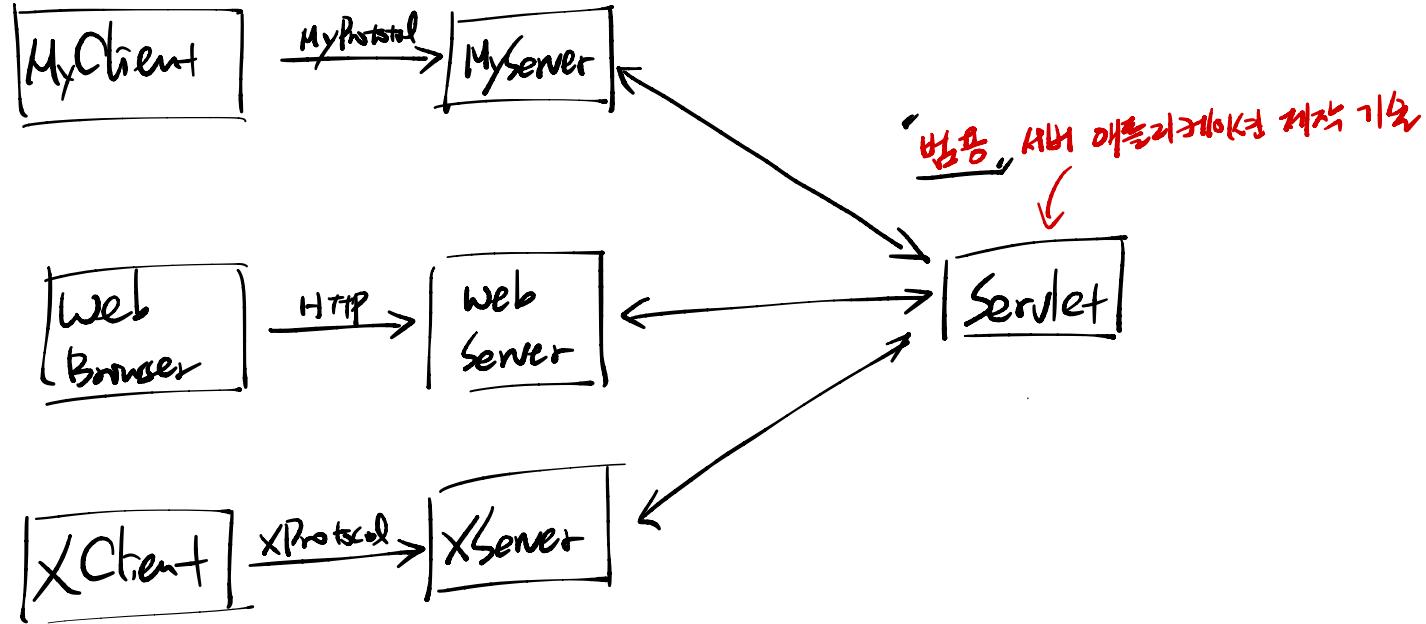
•

\* 서블릿 include → 다른 서블릿의 실행 결과를 포함시키는 방법

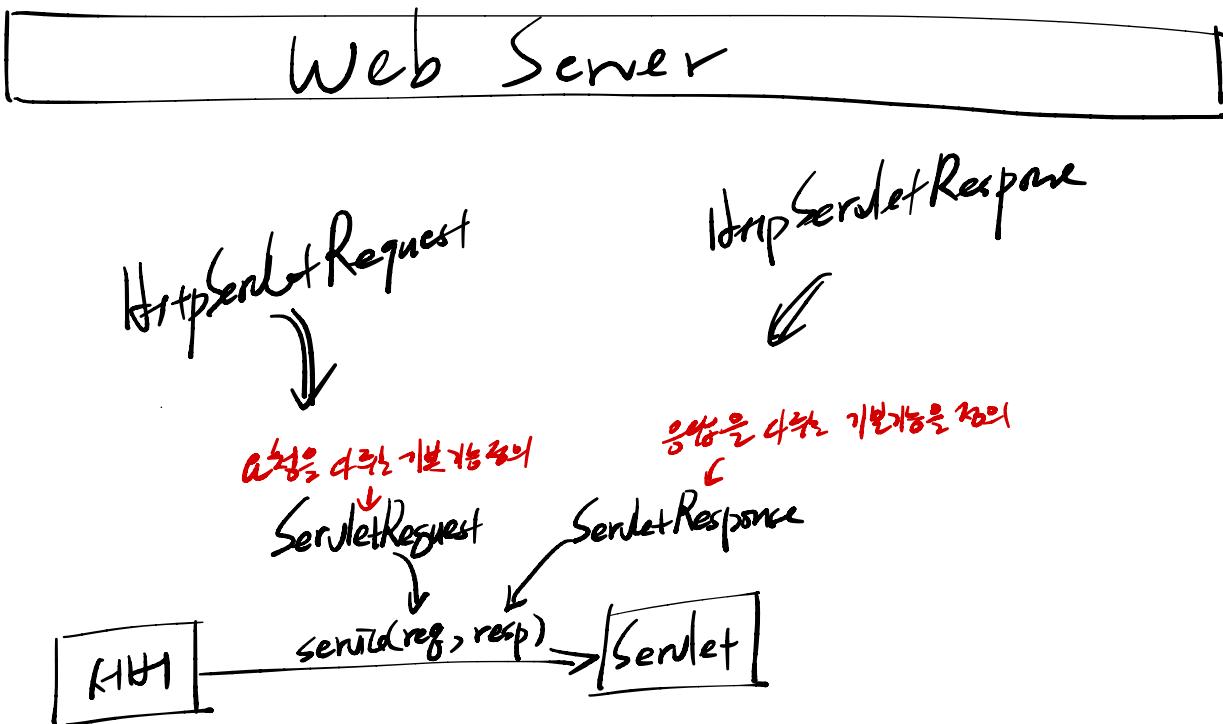
② 구조



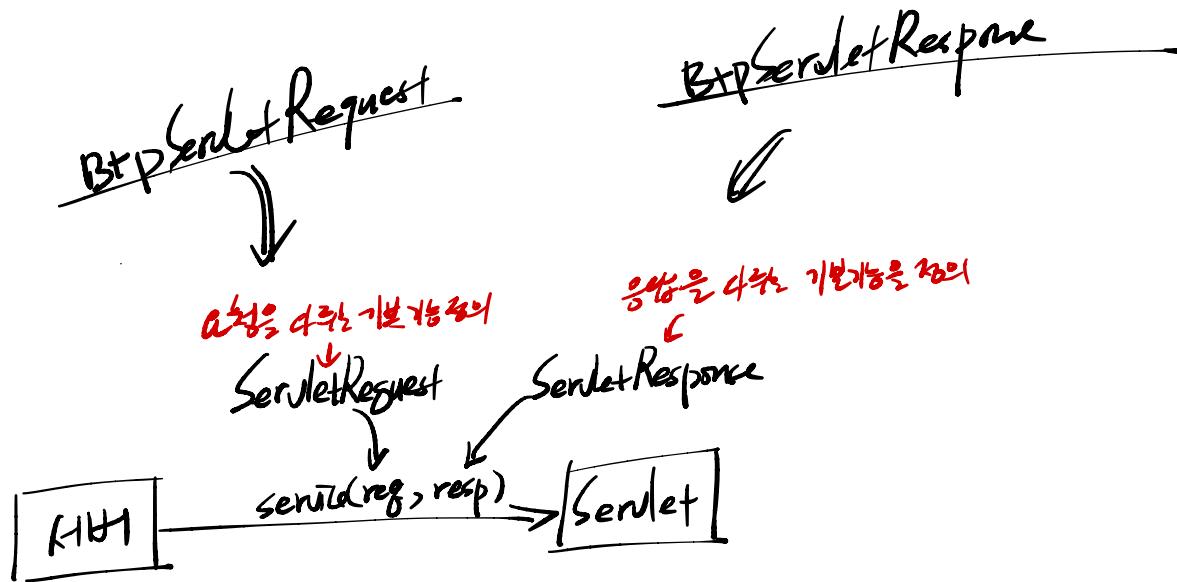
## \* Servlet چیزی چی



## \* ServletRequest et ServletResponse

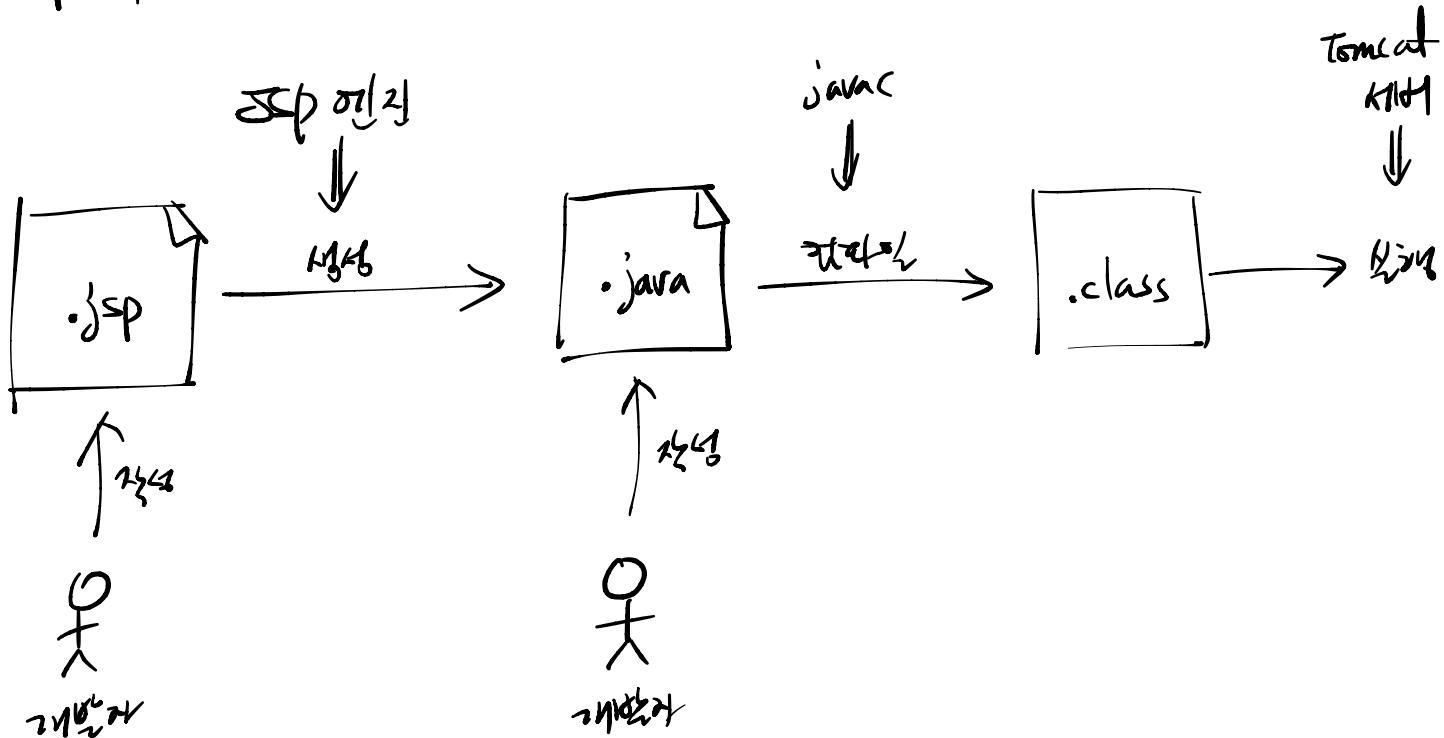


# Bitcamp Server

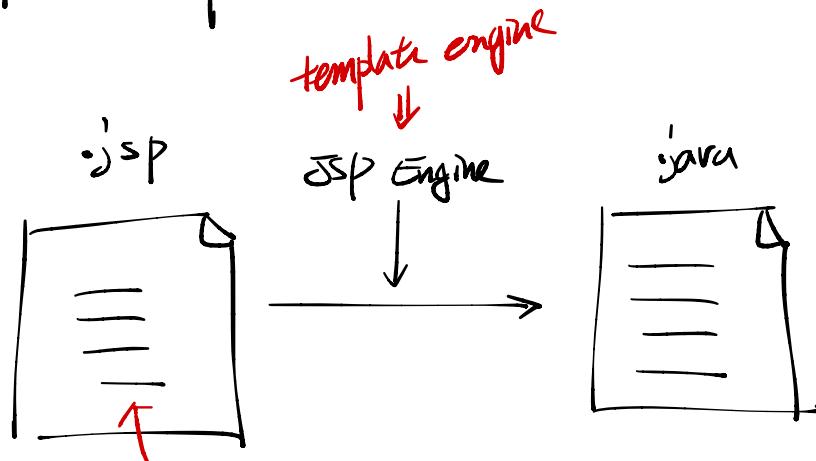


## 50. UI 흐름을 자동 생성하기 — JSP (Java Server Page)

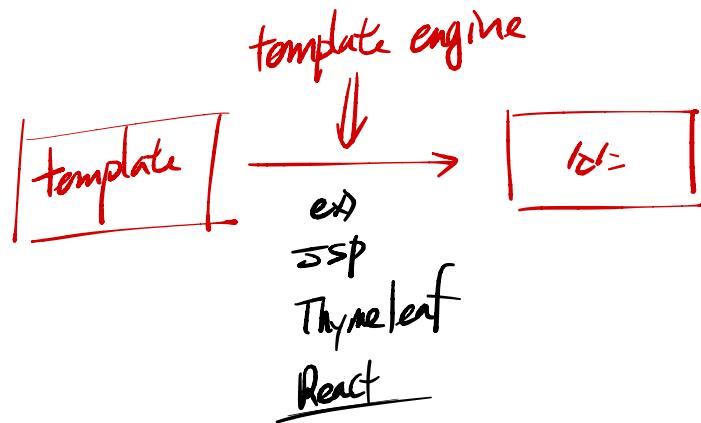
\* JSP 구동 원리



## \* JSP - Template 엔진

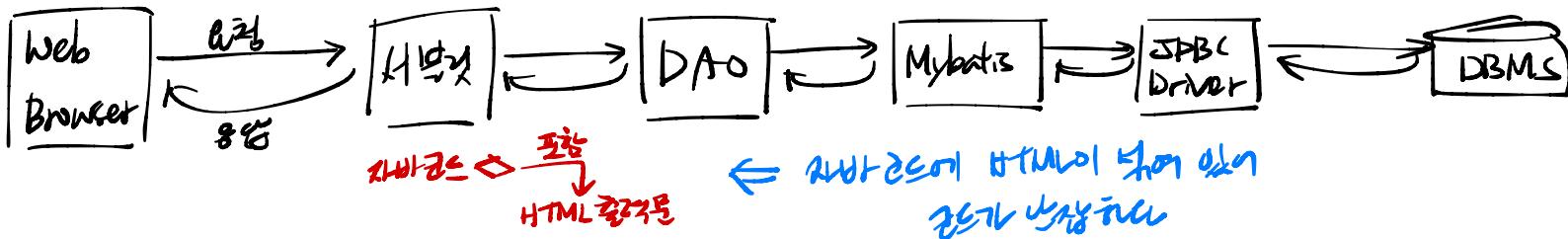


자바 코드를 템플릿으로  
서는 템플릿  
자바 코드는 템플릿  
"template"

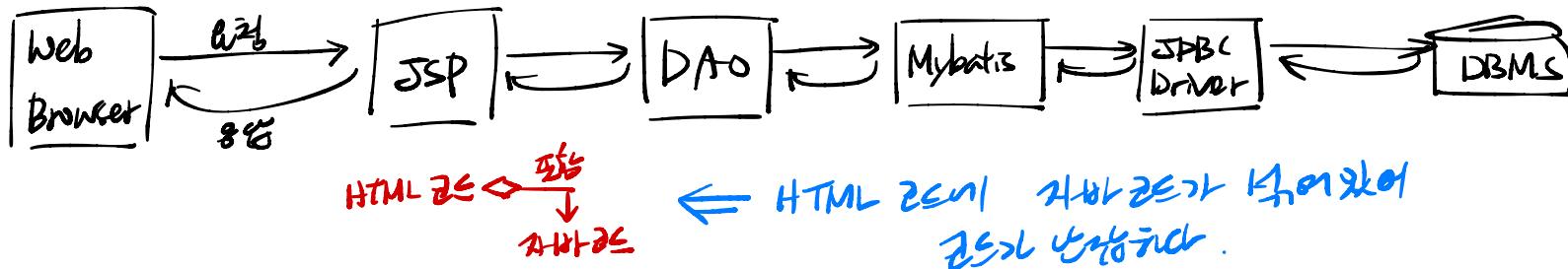


\* Servlet 와 JSP 애플리케이션 아키텍처

① 서블릿 애플리케이션 아키텍처 - 이전 방식

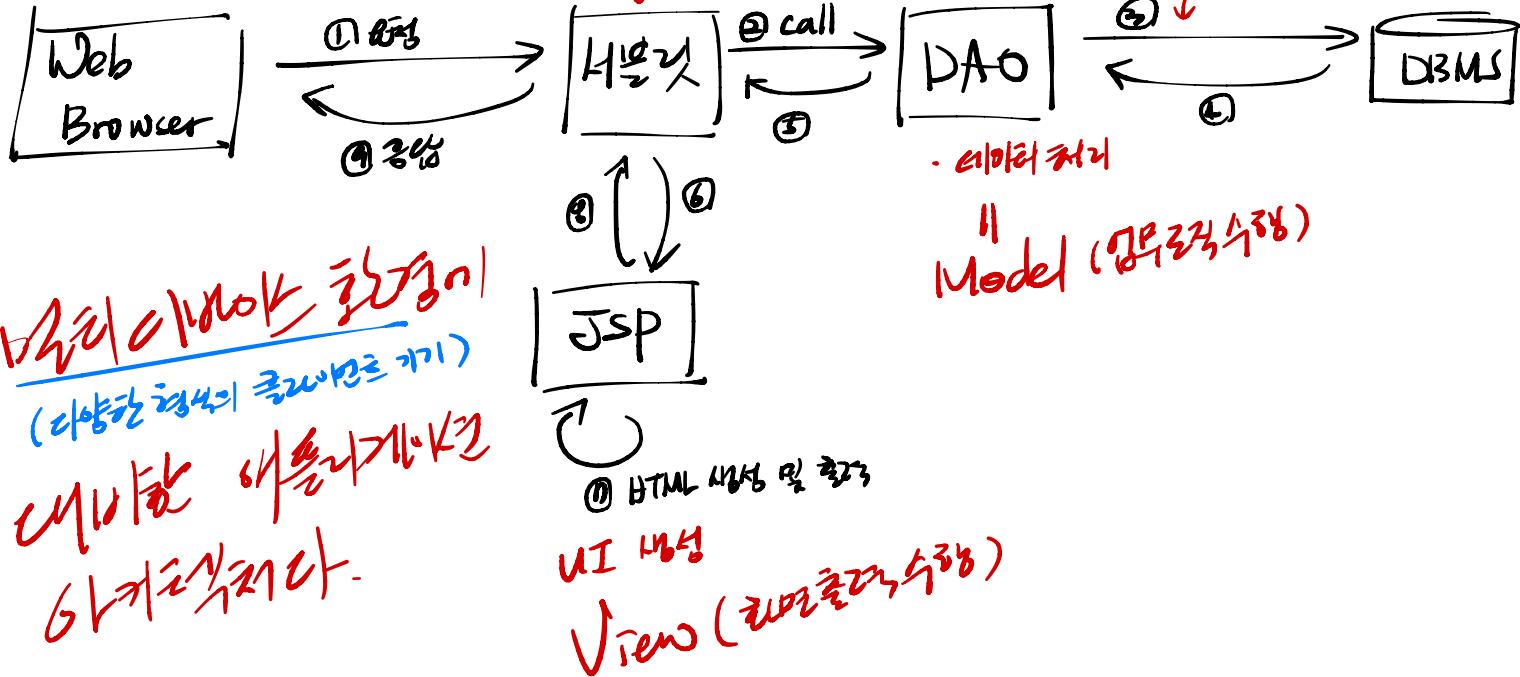


② JSP 애플리케이션 아키텍처 = MVC | 단점



## \* MVC 2 아키텍처

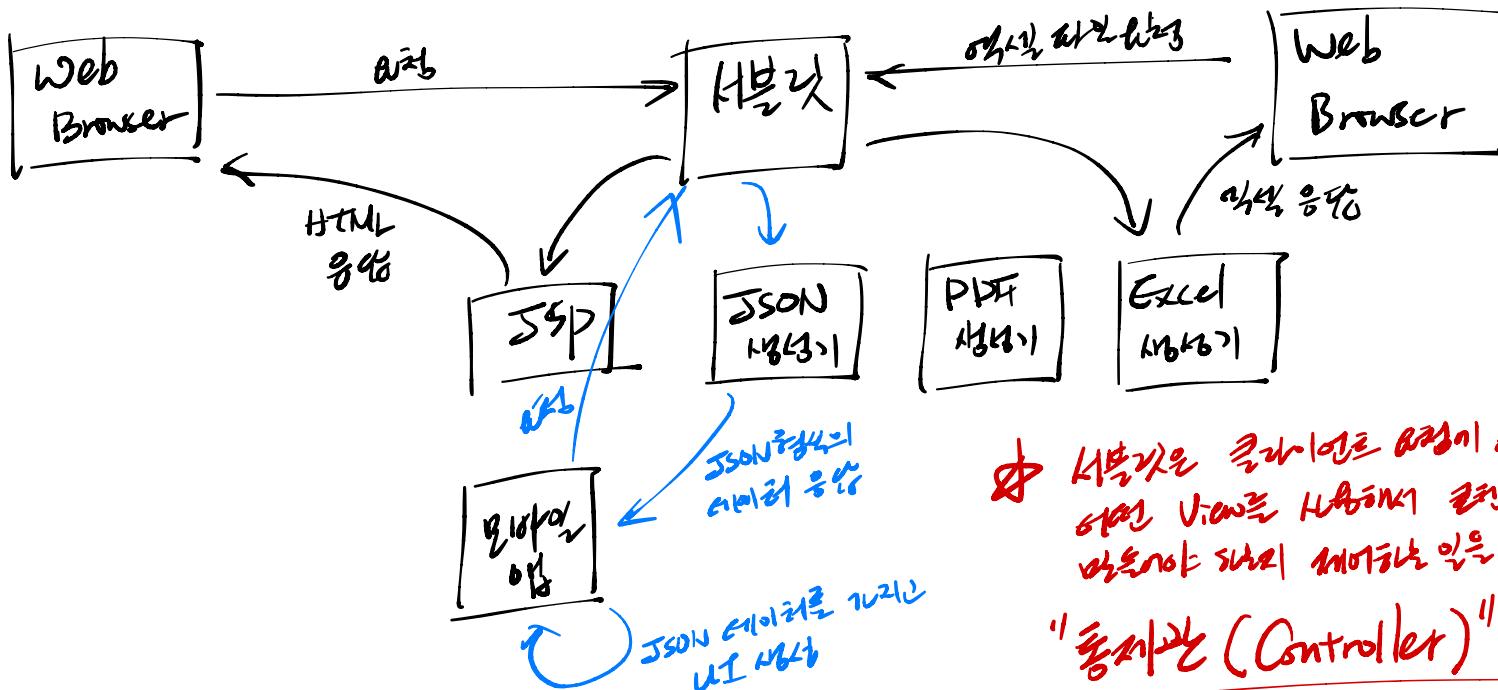
- ✓ 투과율이 높아서 좋다
  - ✓ 일관성이 있다
  - ✓ 구조가 단순
- Controller (어떤 모양)



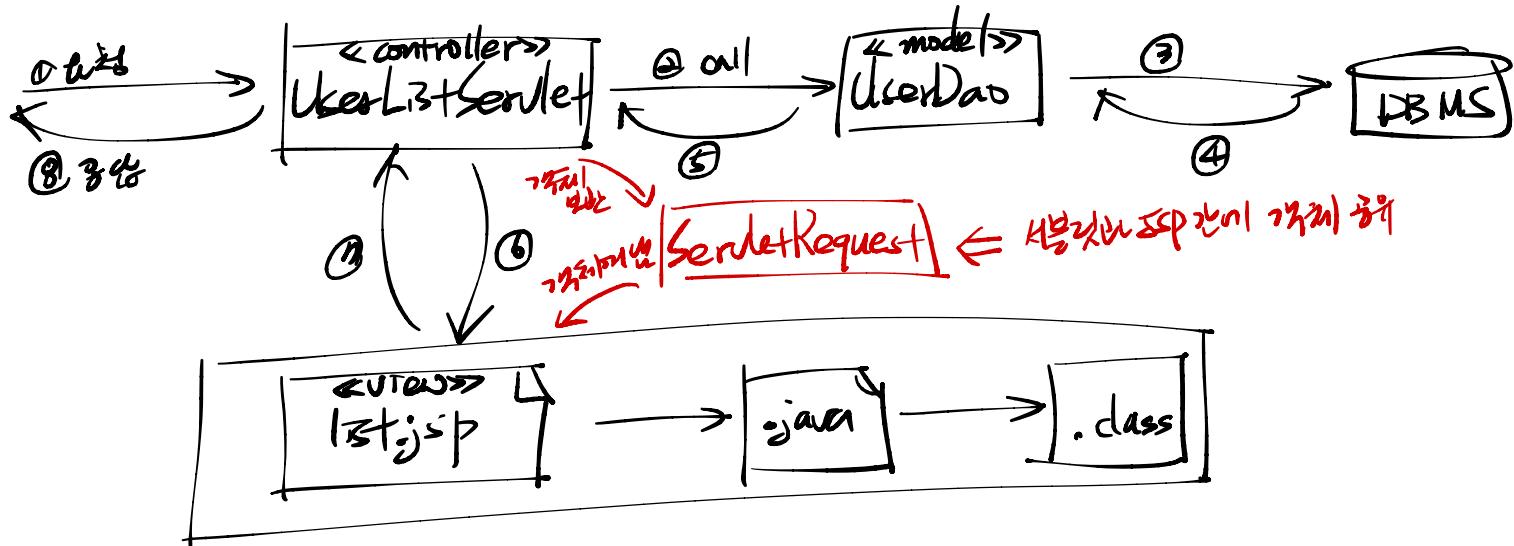
\* 서블릿에서 웹면 흐름을 통제할 때의 예는 아래

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

결과적으로 디자이너가 통제해 어렵다. 정부에 있는 경우 관리하기 어렵다.



\* MVC2 ဆို ဘွဲ့တော်များ



\* 91) 뷰단계

정의하기 위한 초기화  
설정하는 내용은 사용자에게 보여줄 때 DAO, Service, DB연결정보 등.

