# 〈 개념 정리 〉

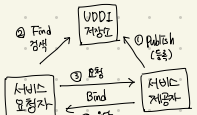
## 웹문서
클라이언트가 서버에 정보를 요청하면 전달 해주는 컨텐츠

- 동적 웹 문서 : 움직이는 웹문서 ex) Java Script..
- 정적 웹문서 : 멈춰있는 웹문서 ex) HTML..

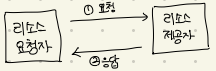
## 웹 애플리케이션
- 웹에서 수행하는 애플리케이션

## 웹 서비스
- 서로 다른 종류의 컴퓨터들 간의 상호작용을 하기 위한 소프트웨어 시스템

### ① SOAP 기반 웹 서비스



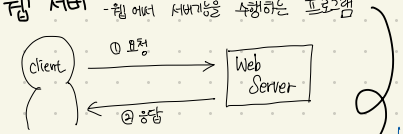

### ② RESTful




〈 둘다 웹 애플리케이션 간에 데이터 통신을 허용하는 API 추출방법을 정의! 〉

| | | |
|---|---|---|
| 프로토콜 | VS | 아키텍처 스타일 |
| 서비스 인터페이스로 서버접근 | VS | URI을 통해 서버접근 |
| XML | VS | HTTP, JSON |
| 엄격한 통신규약 | VS | 미리 알필 필요가 없음 |

## 웹 서버
- 웹 에서 서버기능을 수행하는 프로그램




## 웹 애플리케이션 서버
- 웹 서버의 업그레이드 버전으로 다양한 목적을 서비스하기 위해 HTML 뿐만아니라 다양한 형식들을 처리

Upgrade한 이유
① 여러 개의 서버를 별개로 처리 X → 서버를 개별 컴퓨터에 저장하기 위해
② c/s (client, Server) 구조 각각 개별적으로 처리, 구축하기 위해

(X) Web Logic (웹로직)
Jeus (제우스)
Tomcat (톰캣)

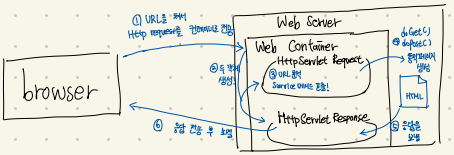
## 컨테이너
- 서블릿과 JSP는 물론 콘텐츠를 생성하는 웹 컴포넌트들, 이런 웹 컴포넌트를 저장, 메모리로딩, 객체 생성 하는곳!

### 서블릿 컨테이너
- Java 수행환경
- 웹 서버
- 서블릿 컨테이너

### JSP 컨테이너
- 자바 수행 환경
- 웹 서버
- 서블릿 컨테이너
- JSP 컨테이너

# Servlet
- Java 파일
- Java 코드안에 HTML 코드
- DB 통신, 비즈니스 로직 처리 위주
- Servlet 이 수정 된 후 컴파일 후 전체코드 업데이트 필요 (.class 파일 생성)

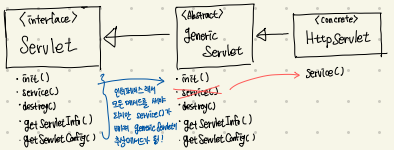
## 동작 방식



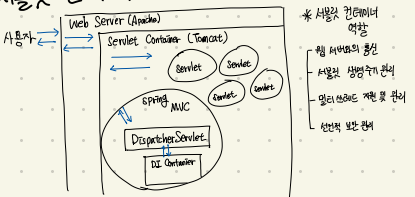

① URL을 통해 Http request를 컨테이너에 전송
② 컨테이너는 2개의 Servlet 객체 생성! (HttpServletRequest req, HttpServletResponse resp)
③ URL을 분석해 service() 메서드 호출
④ 클라이언트의 POST, GET 여부에 따라 doGet(), doPost() 호출
⑤ 동적 페이지 생성 후 HttpServletResponse로 응답하고 소멸

## 서블릿 API 계층 구조

Servlet (Interface)
ServletConfig (Interface)
→ Generic Servlet (Abstract class) → HttpServlet (class)



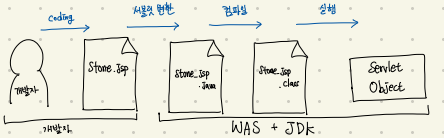

## 서블릿 컨테이너




* 서블릿 컨테이너 역할
- 웹 서버와의 통신
- 서블릿 생명주기 관리
- 멀티쓰레드 지원 및 관리
- 선언적 보안 관리

# JSP (Java Server Page)
- JSP 파일
- HTML 코드안에 Java 코드
- 요청 결과를 나타내는 HTML 작성에 유리
- JSP 수정된 경우 재배포를 필요로함 WAS가 알아서 처리

## 동작 방식



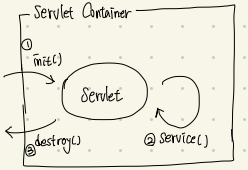

① 서블릿에 대한 요청을 받으면 서블릿 컨테이너에 요청을 넘김
② 요청 받은 컨테이너는 Http Request, Http Response 객체를 만들어 서블릿 doPost(), doGet() 메서드 중 하나를 호출
③ 만들어진 결과물은 해당 페이지를 요청하면 컴파일되어 클라이언트로 감

## 서블릿 API 계층 구조

Servlet (Interface)
ServletConfig (Interface)
→ Generic Servlet (Abstract class) → HttpServlet (class)




## Servlet Life Cycle




① init()
클라이언트 요청이 들어오면 컨테이너에 해당 서블릿 메모리가 있는지 확인 → 없으면 init() 메서드를 호출하여 객체 생성
→ 있다면 init() 처음 한번만 실행되기 때문에 서블릿 스레드에 공유적으로 사용해야 할 것을 오버라이딩!

② service()
- doGet()
- doPost()

③ 컨테이너가 서블릿에 종료요청하면 destroy() 메서드실행 (한 번만 실행되며 오버라이딩 하지 않음)

## JSP Life Cycle