클라이언트-서버 구조

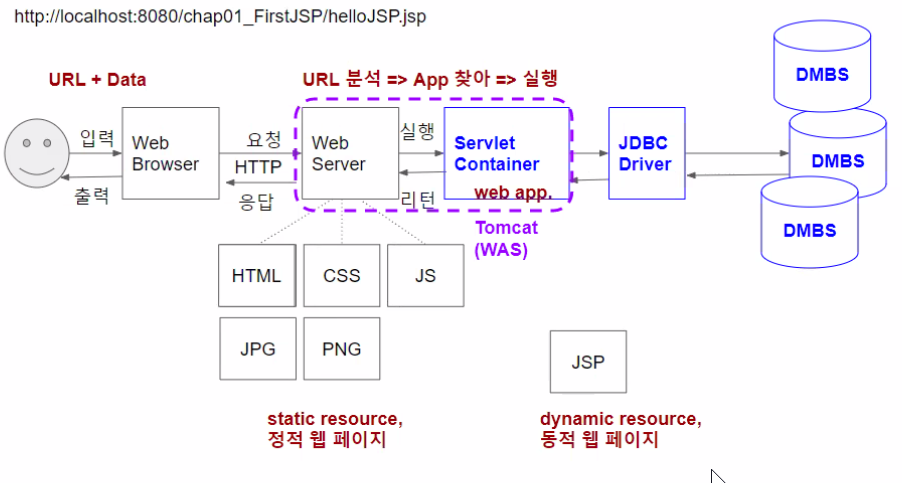
웹은 기본적으로 클라이언트-서버 구조

client-server 간 통신을 위해 HTTP 이용

Hyper Text를 전송하기 위한 프로토콜(통신을 위해 정의된 규약)

URL(Uniform Resource Locator) : 웹페이지 주소

프로토콜:// 서버 주소: 포트번호/파일경로



Web Server와 Application Server와 Servlet Container

Web Application Server

클라이언트로부터 요청을 받아 app에 대한 로직을 실행하여 웹서버로 반송해주는

소프트웨어

클라이언트와 DBMS를 연결

웹서버와 was는 독립적으로 존재할 수 있다

동적,정적 컨텐츠 제공 => web 서버 없어도 됨

그럼에도 web서버 사용하는 이유: was의 부담을 줄이기 위해,

was의 환경설정을 노출하지 않기 위해

Web Server

정적인 컨텐츠(http,img,jpg,JS,CSS)를 제공하는 서버

동적인 컨텐츠 제공을 위해 WAS에 요청을 보내고 결과를 클라이언트에 보냄

Servlet Container

서블릿은 스스로 작동하지 않으므로 서블릿을 관리해줌

서버사이드에서 동적 웹페이지를 생성 -Java 사용

Tomcat

web.xml(배포 서술자)

웹 애플리케이션의 환경설정 정보를 담은 파일

서블릿 설정, 필터 설정,

WAS(Web Application Server)가 처음 구동이 될 때 이 파일을 읽어

설정내용을 톰캣에 적용하게 됨

프로토콜(Protocol)

네트워크 통해 컴퓨터들이 정보를 주고받는 절차 혹은 통신 규약을 말함

서로 다른 컴퓨터들이 대화하는 데 필요한 공통 언어 역할을 해줌

HTTP도 프로토콜의 한 종류

FTP(File Transfer Protocol),SMTP(Simple Mail TP) 등

포트(Port)

컴퓨터 사이에서 데이터를 주고받을 수 있는 통로

IP 주소를 통해 서버 컴퓨터 위치 파악함

그 후 컴퓨터가 제공하는 특정 서비스 요청: 포트번호를 통해

모든 서비스는 IP+ 포트 번호까지 지정해야 제대로 요청 전달 가능

HTTP는 80번 포트, HTTPS는 443번 포트 등

web 프로그래밍

FrontEnd : 클라이언트 중심의 프로그래밍(HTML,CSS,JS) 영역

클라이언트에서 HTML을 가지고 있거나,

서버로부터 화면 구성에 필요한 데이터만 Nodejs로 받아 화면 출력

=>CSR(Client Side Rendering)

필요한 데이터만 갱신, 실시간 데이터 갱신

데이터 영역에선 결국 백엔드작업, SSR 적용 필요

BackEnd : 서버를 구성하며 서비스를 제공하기 위한 서버쪽 프로그래밍 영역

서버가 모든것을 담당

자바/JSP가 인기

다양한 서버 환경에 대응 가능

검색엔진 최적화에 유리

가장 안정적

서버에 화면 갱신의 과도한 요청이 발생 시 문제가 됨

기존의 대규모 구축된 모놀리틱 아키텍처 방식으로 서버가 적용이 됨

=>MSA(Micro Service Architecture)방식이 확산되고 있음

모놀리틱과 MSA

모놀리틱 : 웹 애플리케이션이 하나의 결과물로 패키징되어 배포

장애에 취약

scale-out(여러 server에서 나누어 일 처리)

서비스 변경이 어렵

한 framework 언어에 종속적

MSA:API를 통해서만 상호작용 가능

각각의 서비스가 모듈화되어 RPC 또는

message-driven API등을 이용하여 통신.

개발이 빠르고 유지보수 유리

단위별로 적절한 기술스택 사용 가능

대신 복잡