

Spring Boot

Part1

웹(Web)과 인터넷(Internet)

정적 웹의 동작 원리

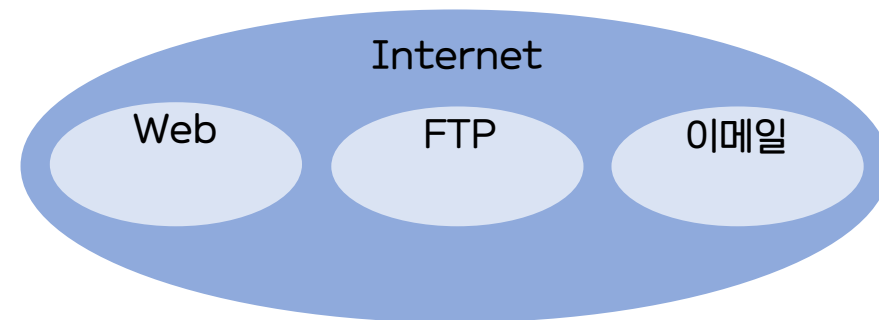
웹의 진화, 동적 웹의 동작 원리

Spring과 React로 구성된 웹의 동작 원리

웹과 인터넷은 같은 것일까?

웹은 인터넷이라는 기술에 포함된 개념이다.

인터넷은 1960년대 등장하였고, 웹은 1990년대에 등장 하였다.



인터넷(Internet)

인터넷은 서로 분리되어 있던 PC를 네트워크 기술을 통해 서로 연결하여 데이터를 통신할 수 있도록 환경을 구성한 기술이다.

(각 도시가 교류가 어려워 사회발전이 어려운 시기에 각 도시를 연결하는 도로를 만들었음. 인터넷이 도로와 같은 개념임)

웹(Web)

각각의 PC를 연결한 인터넷 환경에서 html언어로 만들어진 파일을 전송할 수 있도록 만들어진 기술로 팀 버너스 리(Tim Berners-Lee)가 고안 함.

(도로 위에는 여행을 위해, 지인을 만나기 위해, 물건 전달을 위해... 등 다양한 목적을 갖고 이동하는 차들이 있다. 이러한 차들 중 하나를 웹이라 생각하자)

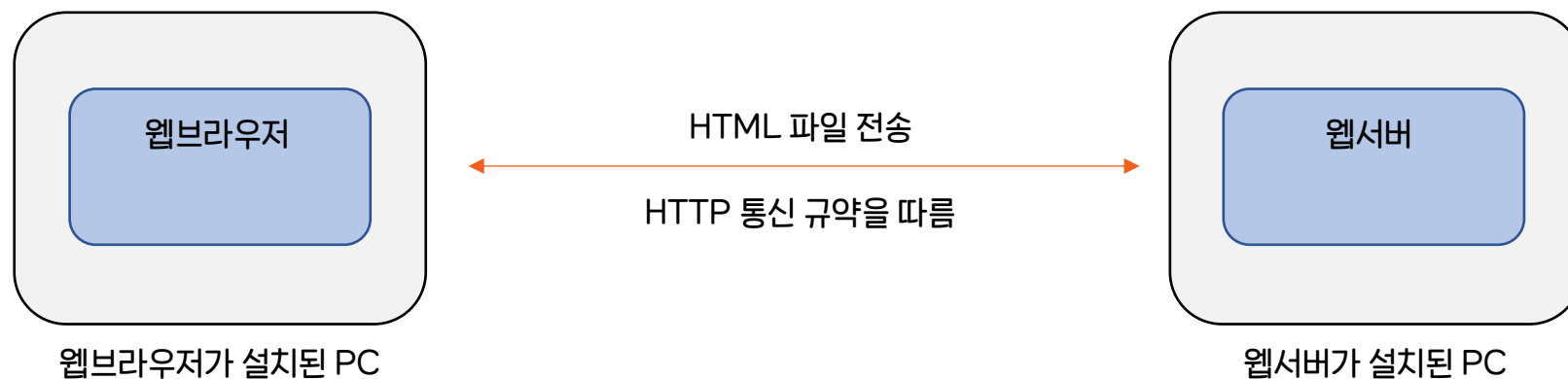
초장기 웹의 모습

팀버러스리는 웹 기술을 고안하며 웹이라는 기술이 인터넷상에서 구동될 수 있도록

웹브라우저(Web Browser)라는 소프트웨어와 웹서버(Web Server)라는 소프트웨어를 개발 함.

그리고 정보를 주고 받을 두 대의 PC 중 한 PC에 웹브라우저 소프트웨어를 설치, 다른 PC에는 웹서버 소프트웨어를 설치하여 HTML 코드를 전송 함.

두 대의 PC가 HTML 코드를 주고 받는 과정에서 사용되는 통신 규약(Protocol)인 HTTP도 팀버러스리가 고안 함.



통신규약(Protocol)

서로 다른 PC가 어떻게 데이터를 교환할지 정해 놓은 규칙. 신호 송신의 순서, 데이터 표현법, 오류 검출법 등을 결정한다.

HTTP(Hyper Text Transfer Protocol)

HTML 파일 전송을 위한 통신규약으로 가장 큰 특징은 서로 다른 PC가 요청(request)과 응답(response)을 통해 통신하며, 무전기와 비슷하게 반-양방향 통신을 한다.

그렇다면 서로 다른 PC에 설치된 소프트웨어인 웹브라우저와 웹서버는 어떻게 데이터를 송수신할 수 있을까?

웹서버와 웹브라우저의 데이터 송수신을 위해 사용하는 개념이 URL이다.

URL

Uniform Resource Locator의 줄임말로, 정형화된 자원 위치 탐색 장치 정도로 해석하면 된다. 여기서의 자원은 html 파일을 말한다.

즉, URL은 특정 HTML파일이 저장된 위치(주소)를 가리키는 말이다.

https://www.youtube.com/

리소스를 요청하는 **protocol**을 나타내는 부분.

https protocol을 이용하여 html파일 주소를

찾겠다는 의미로 해석하면 됨.

리소스를 요청하려는 웹서버의 주소를 나타내는 부분.

웹 서버의 주소는 기본적으로 IP주소와 포트번호로 이루어져 있음.

ex>196.168.0.30:8080

그러나, IP와 PORT는 숫자로만 이루어져 있어 사용에 불편함이 있음.

그래서 이를 사용하기 쉽게 IP와 PORT번호를 **영문으로 바꿔서 표기함**.

이를 **도메인(Domain)**이라 부름.

http vs https(Hyper Text Transfer Protocol Secure)

https는 인터넷상에서 html 송수신을 위한 통신규약인 http(Hyper Text Transfer Protocol)에 secure를 추가한 것이다.

쉽게 얘기하면, https는 기존 http 통신 규약을 보안적으로 업그레이드한 버전이라 생각하면 된다.

정리하면,

우리가 웹브라우저 주소창에 `https://www.youtube.com` 이라 url을 입력하면

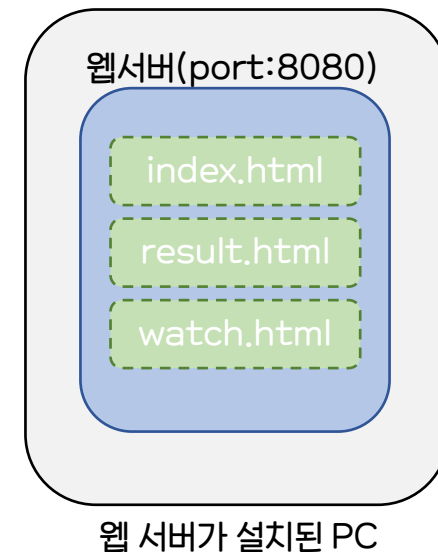
1. 도메인이 해석되어 `https://168.192.0.30:8080`로 변환되고,
2. 이는 `https` 통신 프로토콜 방식으로 `168.192.0.30:8080`를 해석하라는 의미이며,
3. IP주소가 `168.192.0.30` 인 PC에 접근하여 `8080 port`에 위치한 웹서버 소프트웨어를 실행하겠다는 의미이다.

이렇게 웹서버에 접근하면 웹서버가 관리하는 html 파일 중 특정 파일을 아래의 http 통신규약으로 요청할 수 있다.


`https://www.youtube.com/result.html` -> 접근한 웹서버의 `result.html` 파일을 요청하는 http 통신 문법

`https://www.youtube.com/watch.html` -> 접근한 웹서버의 `watch.html` 파일을 요청하는 http 통신 문법

`https://www.youtube.com` -> html파일이 명시되지 않으면, default 값으로 설정된 html파일이 실행 됨.

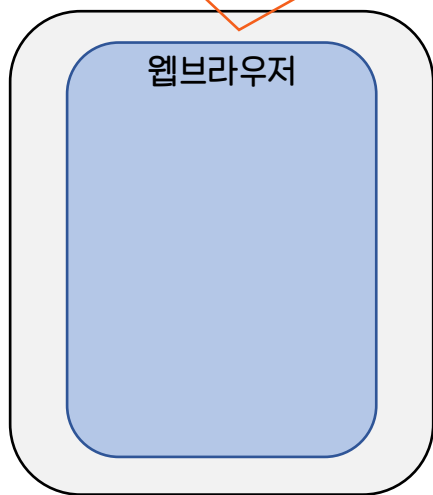


최종 정리

- 인터넷 환경(서로 다른 PC가 데이터 통신을 할 수 있는 환경)에서 HTML코드를 송수신 할 수 있는 기술을 웹(Web)이라 한다.
 - 웹을 인터넷환경에서 동작시키기 위해 추가적으로 개발된 소프트웨어가 웹 브라우저와 웹 서버이다.
 - 웹 브라우저는 HTML코드를 해석하여 화면에 띄워주는 기능을 가진 소프트웨어이다.(크롬, 사파리, 웨일...)
 - 웹 서버는 여러 HTML코드를 저장하고, 저장된 HTML 코드를 관리하는 기능을 가진 소프트웨어이다.(아파치, 엔진엑스...)
 - HTML 코드를 주고 받을 2대의 PC 중 한 대에는 웹 브라우저를 설치하고, 다른 한 대에는 웹 서버를 설치한다.
 - 웹 브라우저에 URL을 입력하면 웹 서버로 특정 HTML 코드를 전송해 달라는 명령을 할 수 있다. 이를 **요청(Request)**라 한다.
 - 요청을 받은 웹 서버는 자신이 관리하는 HTML 파일 중 요청에 부합하는 HTML파일을 웹 브라우저에 전송한다. 이를 **응답(Response)**라 한다.
 - 웹서버는 요청에 따라 html을 제공하는 역할을 하기에, 웹서버가 설치된 PC를 제공자라는 의미에서 **서버(Server)**라 부른다.
 - 웹브라우저는 웹서버에 html파일을 요청하기에, 웹브라우저가 설치된 PC를 서비스를 제공받는 자라는 의미에서 **클라이언트(Client)**라 부른다.
 - 웹 브라우저는 Client 측에 설치된 소프트웨어라는 뜻으로, **웹 클라이언트**라고도 부른다.
- 

초장기 웹 기술의 도식화

1. https://www.youtube.com/watch.html url로 입력
2. https://192.168.0.30:8080/watch.html로 해석
3. 해당 url이 가리키는 주소의 웹 서버를 실행하면서 watch.html 파일을 요청함



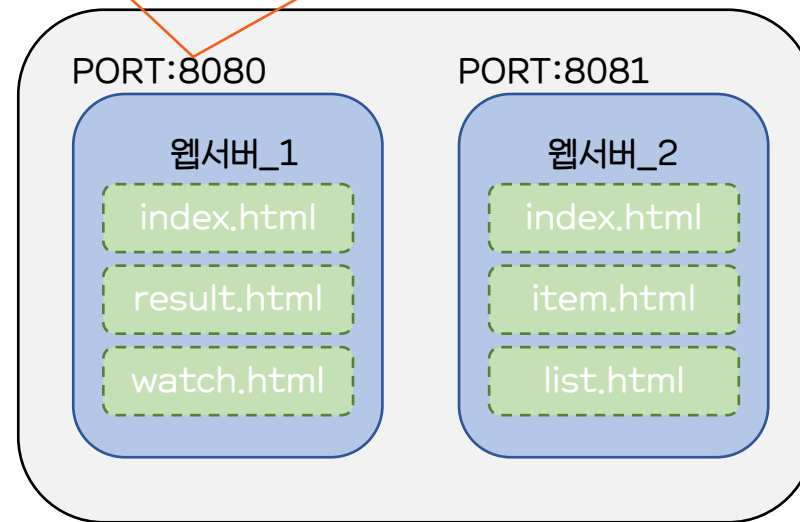
Client

(웹 브라우저/클라이언트가 설치된 PC)

웹 브라우저에서 웹 서버 측으로 특정 html파일을 전송해달라고 명령 함.
이 명령을 요청(request)이라 부르며, 요청은 url 문법에 의해 전달 됨.

서버는 요청에 맞는 html파일을 전송함. 이를 응답(response)라 함.
응답을 통해 전달받은 html파일은 웹 브라우저가 해석하여 화면에 띄워줌.

1. client의 요청 정보에서 IP와 PORT를 해석하여 부합하는 웹 서버가 실행 됨.
2. 실행된 웹 서버는 요청한 HTML를 찾아 client에 전송



Server(IP : 168.192.0.30)

(웹 서버가 설치된 PC, 한 대의 PC에는 여러 웹 서버 소프트웨어를 실행할 수 있으며 각각의 웹 서버는 고유한 PORT번호안에서 독립적으로 실행 됨)

우리는 실습 시 어떻게 웹 브라우저와 웹 서버를 사용하여 코드 연습을 하고 있을까? (html, css 실습 한정)

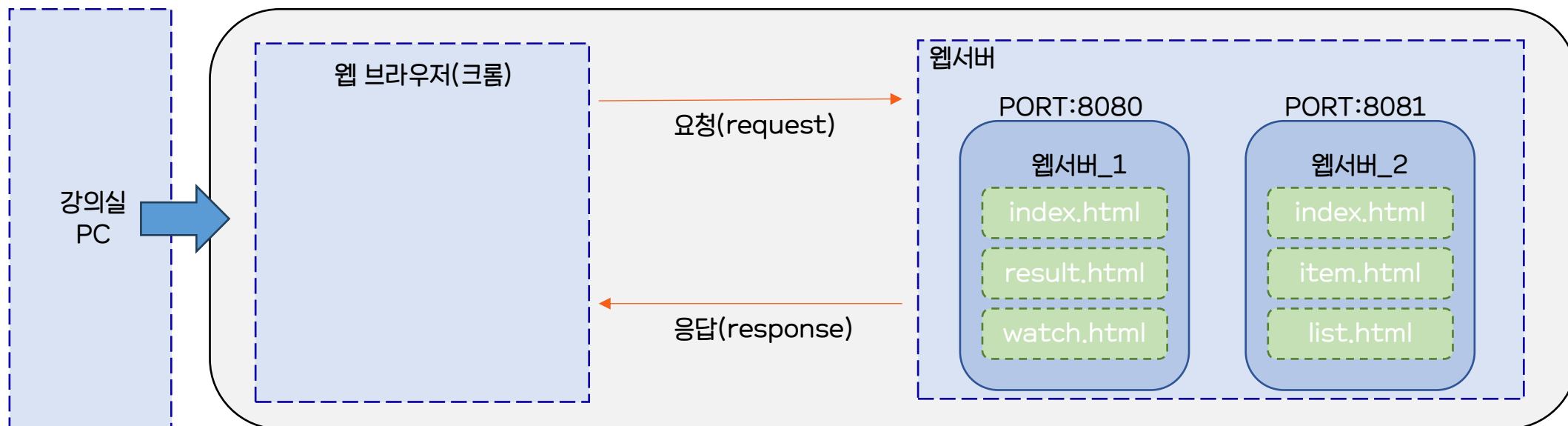
앞선 이야기는 실제 운영에서의 웹 구조이며, 두 대의 PC에 각각 웹 서버와 웹브라우저가 설치된 환경이었다.

실무에서 우리는 클라이언트 측은 생각할 필요없이, 서버 측 PC의 웹서버 안에서 관리되고 실행되는 파일들을 만드는 일을 하는 것이라 생각하면 된다.

하지만 우리는 HTML을 직접 만들고 하고, 만든 HTML을 웹브라우저를 통해 실행까지 하고 있으며, 이는 한 대의 PC에서 진행되고 있다.

결국 우리는 한 대의 PC에 웹서버와 웹브라우저를 모두 설치하여, **한 대의 PC가 클라이언트의 역할과 서버의 역할 모두를 진행**하고 있는 것이다.

그렇다면, 웹 브라우저는 크롬인데.. 웹 서버는 설치한 적이 있는가? **VSCode의 Live Server 확장 프로그램이 바로 웹 서버인 것이다.**



인터넷 상에서 다른 PC에 접근하기 위해 사용하는 것이 IP주소다. 그리고 내 PC에 접근하는 IP주소가 `http://localhost/` 다.

결국 `http://localhost:8080/result.html` 라는 url은 내 pc의 8080 port에서 실행되는 웹 서버를 찾아, 그 안의 `result.html`을 전송해달라는 명령인 것이다. 그리고 이렇게 전달받은 `result.html`코드를 크롬(웹브라우저)이 해석하여 화면에 띄워준다.

지금까지의 학습을 통해 이미 알고 있겠지만 html과 css는 java, javascript 등의 프로그래밍 언어와는 다른 특징이 있다.

바로 프로그래밍적 요소나 문법이 존재하지 않는다는 것이다. (변수, 조건문, 반복문 등)

그렇기에 html과 css는 엄밀히 말하면 프로그래밍 언어라 하지 않는다.

초창기 웹은 모두 정적 웹이었다.

정적이라는 말은 움직임이 없음을 의미한다. html 코드로 말하자면, html은 변수라는 개념이 없기 때문에 화면에 나타나는 정보에 변화를 주기 위해서는 html 코드를 수정하는 방법 밖에 없음을 말한다. 결국 코드를 수정하지 않는 한 html코드의 해석 결과 실행 내용은 절대 변하지 않는다. 이것을 정적이라 표현하는 것이다.

애초에 html이 탄생한 계기가 모든 정보를 책으로만 간직하던 것에서 발생하던 여러 제한(공간제한, 시간제한 등)을 해결할 방법을 찾던 중, 인터넷이라는 네트워크 환경이 발전함에 따라 인터넷 상에 문서를 저장하여 여러 제한을 해결하고자 나온 기술이다. 그렇기에 필요한 정보만 문서처럼 만들면 되었기 때문에 변수라는 개념이 필요하지 않았다.

하지만 시간이 흘러 인터넷과 웹은 엄청난 속도로 발전하였고, 정적 웹의 한계(생산성 낮음)로 인해 동적 웹의 필요성이 대두되었다.

동적 웹이란 코드를 한번 만들어 놓으면 조건 혹은 입력되는 데이터에 따라 결과가 달라지는 페이지를 말한다.

html 에서는 이러한 개념을 도입하는 것이 불가능 했기에, 새로운 먹거리를 찾던 프로그래밍 언어(C, JAVA)진영에서 동적 웹에 대한 해결책을 내놓기 시작했다.

프로그래밍언어(C, JAVA)를 개발한 회사에서는 본인들의 기술력을 바탕으로 HTML을 동적으로 구현할 수 있는 새로운 기술을 선보였다.

ASP(C 진영)와 Servlet, JSP(JAVA 진영) 등이 바로 동적 웹 개발을 위해 내놓은 기술이다. (여기서부터는 자바 진영에 대한 이야기만 하겠다)

Servlet(서블릿)과 JSP(Java Server Page)

Servlet : 자바언어로 구현한 코드를 베이스로하여 html코드를 추가. -> 해석하면 html 코드로 변환 됨.

JSP : HTML 코드로 구현한 Page를 베이스로 자바 코드를 추가. -> 해석하면 html 코드로 변환 됨.

Servlet과 JSP는 현재 현업에서도 많이 사용되고 있다. 그리고 자바 베이스의 모든 웹 개발 기술(spring도 해당됨)은 모두 servlet과 jsp기술을 바탕으로 업그레이드 된 기술이기 때문에, 이 둘의 이해도를 높이는 것이 웹 개발의 기초가 탄탄해지는 밑바탕이 된다.

우리가 생각해봐야 하는 것은...

우리 과정에서는 servlet과 jsp를 학습하지 않기 때문에 이에 대해서는 더이상 언급하지 않겠다.

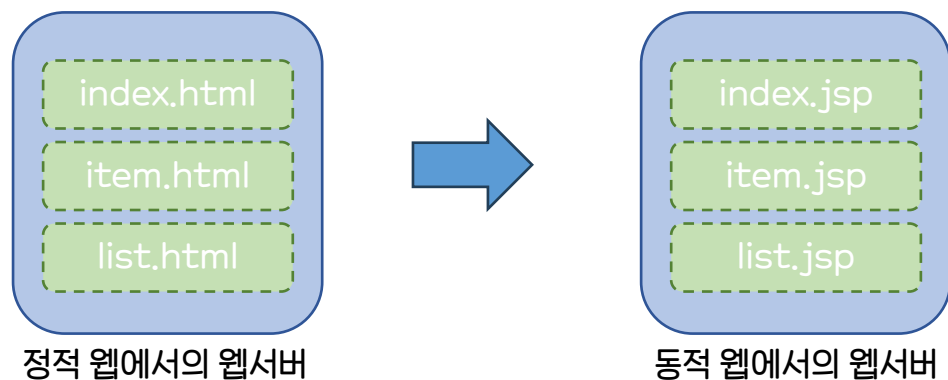
지금 이야기하고 싶은 것은 이렇게 웹 기술이 발전함에 따라 원시적인 웹 동작방식(초창기 웹 동작 원리)은 어떻게 바뀌게 되었을까 하는 것이다.

* Servlet과 JSP를 알면 Spring 학습에 도움이 되는 것은 맞지만, 모른다고 Spring 학습이 어려워지는 건 아닙니다. Spring 학습을 하며 필요한 기초 지식은 그때 그때 학습할 예정이니, 걱정하지 않으셔도 됩니다.

초창기 정적 웹은 html, css로만 구성되어 있었다. 그리고 이러한 html, css 코드는 웹 브라우저에서 해석되어 화면으로 표현된다.

웹 서버는 웹 브라우저(웹 클라이언트)의 요청에 맞는 HTML, CSS파일을 찾아 응답을 해주는 역할을 하였다.

그리고 동적 웹 기술이 태어남과 동시에 정적 웹 페이지 제작에 사용한 HTML파일은 JSP파일로 전환되어 갔다.



WAS(Web Application Server)의 등장

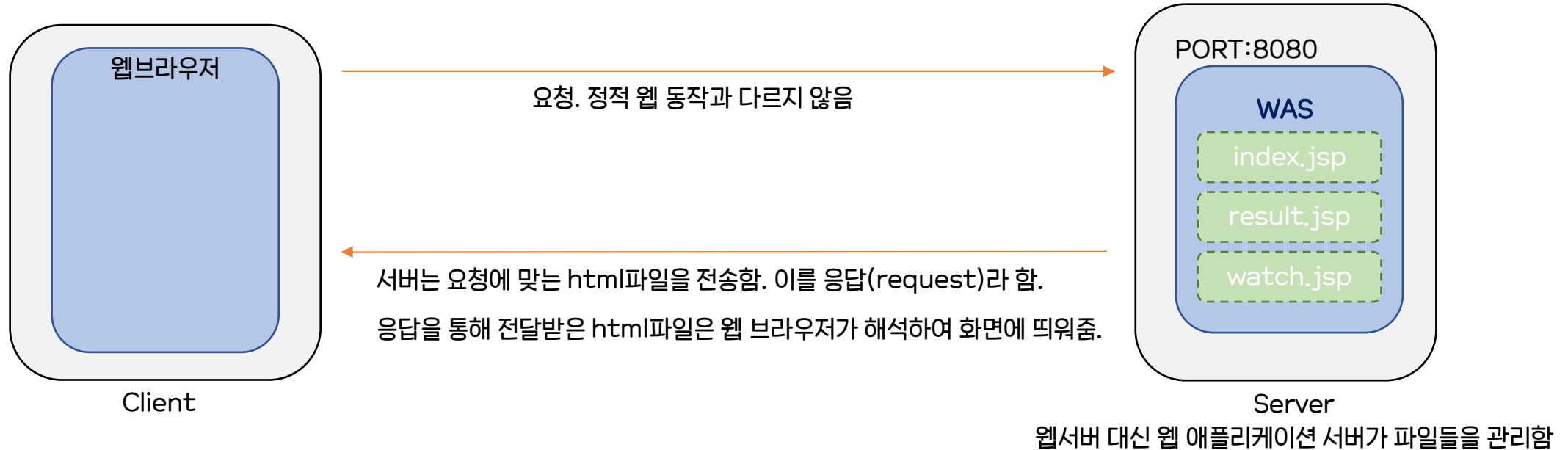
초창기 시절 웹서버는 html을 해석하고, 클라이언트의 요청에 맞는 html을 찾아 클라이언트에 전송하는 기능을 담당하는 프로그램이었다. 그렇기 때문에 html파일이 jsp파일로 대체되면서 웹서버는 제 구실을 할 수가 없었다. 결국 이러한 동적 웹 기술(jsp, servlet)을 해석할 수 있는 소프트웨어가 필요했으며, 이러한 요구를 충족시키기 위해 나온 소프트웨어를 WAS(Web Application Server)라 부른다.(제우스, 톰캣 등)

결국 서버 PC에 설치되었던 웹서버(WS)는 점차 웹 애플리케이션 서버(WAS)로 교체되어 갔다.

* WS가 WAS로 교체되었다고 해서 요즘 WS를 사용하지 않는 건 아닙니다. 실무에서는 WS, WAS를 함께 사용합니다.

* Spring이 동작되기 위해서도 WAS를 설치해야 합니다. 하지만 Spring Boot에는 톰캣이라는 WAS가 내장되어 있어, 별도의 설치 과정은 필요하지 않습니다.

동적 웹 동적 원리의 도식화



- 클라이언트에서 url을 통해 서버로 요청을 보낸다.
- 서버의 WAS가 요청을 분석하여 요청에 맞는 JSP파일을 찾는다.
- JSP파일은 자바코드 + HTML 코드가 합쳐진 코드이다. WAS는 이를 해석하여 HTML파일로 변환한다.
- HTML파일로 변환된 코드를 클라이언트에 전달하고, 클라이언트의 웹 브라우저는 전달받은 HTML을 해석하여 화면에 띄워준다.

우리가 학습하는 Spring과 React는 안타깝게도 동적 웹 구동 방식에서 좀 더 변화된 구동 메커니즘을 가지고 있다.

정확히 말하면 Spring만 사용할 때는 동적 웹 동작원리와 같지만, React를 추가적으로 사용하기 때문에 달라지는 것이다.

여기에 대해 더 이야기하기 전에 현재 실무에서 가장 많이 사용되는 프로젝트 구조는 다음과 같다.

- Servlet + JSP + DB

Spring 프레임워크가 등장하기 전 웹 프로젝트의 구조. 코드가 좀 더럽긴 하지만 매우 직관적이다. 프론트와 백의 구분이 명확하지 않다.

1990년대부터 2010년대까지 사용했기에 현재 운영되는 시스템 중 많은 시스템이 이 구조로 만들어져 있다.

Servlet에는 자바코드를 작성하고, 화면에 보여지는 html 대신 jsp를 사용한 것이다.

- Spring Framework + JSP + DB

Servlet에서 작성하던 자바코드를 Spring Framework로 대체하고, 화면은 html 대신 jsp를 사용한다.

Servlet 대신 Spring Framework를 채택하여 자바 코드가 깔끔해졌지만 여전히 프론트와 백의 구분이 명확하지 않다.

개인적 판단으로 위의 구조와 함께 현재 실무에서 가장 많이 운영되고 있는 방식이다.

다음장에 계속...

- Spring Boot + tymeleaf(타임리프) + DB

위의 구조에서 jsp 기술을 통해 화면을 구현한 것을 tymeleaf로 대처한 것이다.

jsp보다 tymeleaf가 더 안정적이었고, 이러한 이유로 Spring 진영에서 jsp보다 tymeleaf 사용을 권장하였으며,

Spring Framework의 업그레이드 버전인 Spring Boot에서는

jsp 기술의 공식 지원의 종단을 선언하며, 현재 화면을 구성할 때 react를 사용하지 않는다면 가장 많이 선택받은 화면 구현 기술이다.

tymeleaf의 문법은 jsp와 비슷하지만 개인적인 견해로는 jsp가 더 직관적이고 쉽다. jsp와의 차별점은 프론트와 백의 분리라고 얘기하지만 tymeleaf를 사용해도 여전히 프론트와 백을 명확히 구분하는 것에는 한계가 있다.

- Spring Boot + React + DB

자바코드는 Spring에서, 화면은 react로 구성한다. react 자체의 난이도가 높지만 프론트와 백을 완벽히 구분할 수 있다.

그렇기에 만약 지원하는 회사가 풀스택 개발자를 요구한다면 react를 사용하지 않을 가능성이 높고,

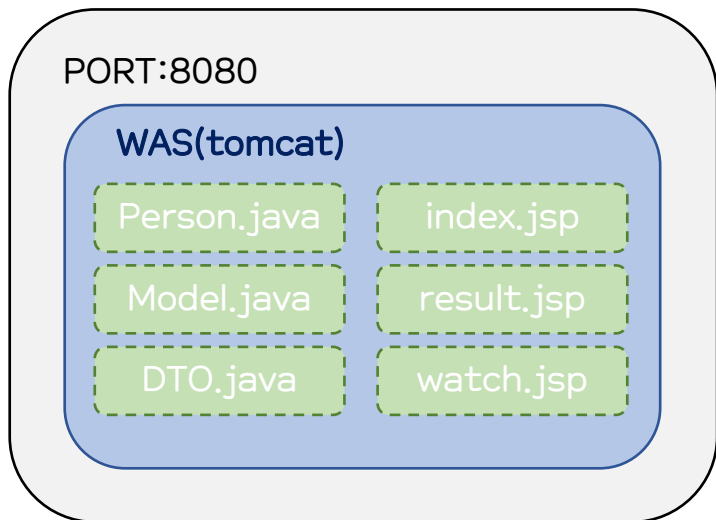
프론트개발자와 백엔드개발자를 따로 뽑는다면 react를 사용할 가능성이 상대적으로 높다.

약 2~3년 전 Spring 진영에서 공식적으로 react를 지원함에 따라 급부상하고 있다. 다만, react가 jsp, tymeleaf보다 어렵고 전혀 다른 메커니즘을 가지고 있기 때문에, 실무 경력이 오래된 개발자일수록 react로의 전환을 꺼려하고 있어, 지금은 react로 변해가는 과도기다.

참고로 말하지만 위의 웹 구조 중 좋고 나쁜 것은 없다. 장점과 단점이 있을 뿐이며, 시대에 따라 유행하는 기술이 다른 것 뿐이다.

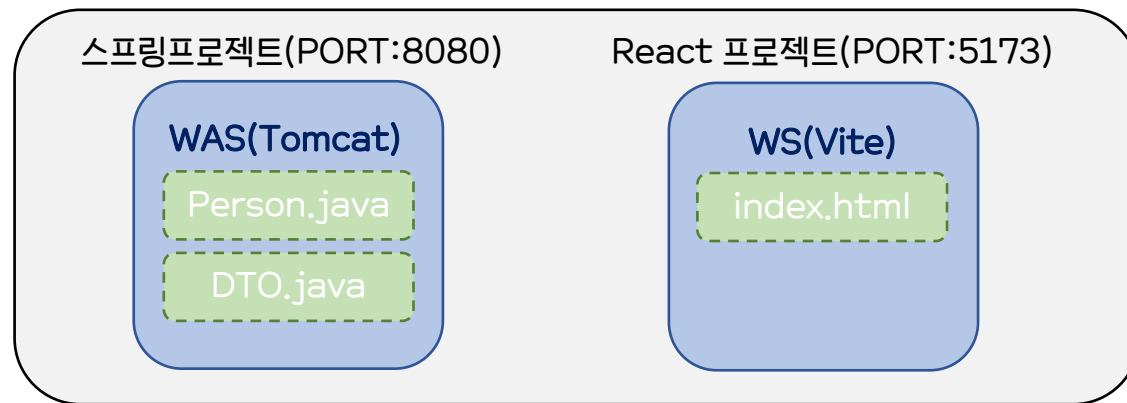
앞서 이야기한 여러 자바기반 웹 프로젝트의 구조와 react가 포함된 웹 프로젝트의 차이점은 WAS 혹은 서버의 분리이다.

React를 사용하지 않은 웹 프로젝트의 구조



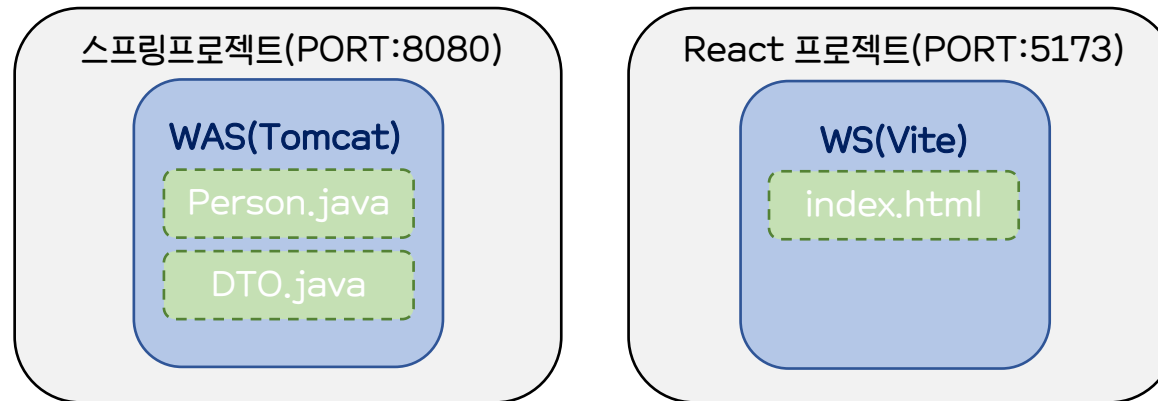
React를 사용하지 않는 서버
백엔드 코드인 자바코드와 프론트 코드인 jsp, tymeleaf 등이
하나의 WAS에서 관리된다.

React를 사용하는 웹 프로젝트의 서버 구조 - 1(수업에서의 구조)



한 대의 서버에 Spring 파일을 관리하는 WAS(Tomcat)와
React 파일을 관리하는 WS(Vite)로 별도로 실행 됨

React를 사용하는 웹 프로젝트의 서버 구조 - 2 (서버 자체를 분리)



아래의 그림은 Spring과 React를 사용한 웹 프로젝트의 흐름을 도식화한 것이다.(react의 자세한 부분은 React 학습 파트에서...)

