왜 프레임워크(그 중에서도 스프링부트)를 사용하는가?

HelloControler.java, HelloDao.java, schema.sql 단 세 개의 파일 수정으로 동작하는 애플리케이션을 만들 수 있었음

Build.gradle에 선언된 의존성(우리가 필요한 라이브러리들을 참조하게끔 gradle 파일에 선언해주면) 덕분에 몇 줄의 코드로 수많은 다른 코드를 사용할 수 있음

프레임워크는 공통적!포스트잇 노트, 폰트, 텍스트이(가) 표시된 사진

자동 생성된 설명

스프링은 사실 단순히 프레임워크라기보다는 다양한 분야로 애플리케이션 제작에 도움을 주는 프로젝트 그룹이다.

자바 기반 엔터프라이즈 애플리케이션 제작에 도움을 주는 전반적인 프로그램이 / 설정 모델을 제공하고 있음도표, 라인, 직사각형, 평면도이(가) 표시된 사진

자동 생성된 설명

우리가 집중할 모듈은 Spring MVC, Spring JDBC, Spring Core (Gradle이라는 관리자 모듈의 구조가 src – main, test 이렇게 자동으로 생성됨)

Build.gradle: 사용자 의존성을 명시(정의)해주는 파일

HTTP 요청과 응답

사용자가 브라우저를 통해 Nextstep 페이지에 들어가는 동안 어떤 과정을 거칠까?

사용자는 브라우저 주소창에 입력 Nextstep 사이트의 주소를 입력

브라우저는 주소를 인식해서 어디에 요청을 보낼지 판단하고 해당 서버에 요청을 보냄

서버는 여러 절차를 거친 뒤 요청을 한 브라우저에 요청받은 것을 응답

브라우저는 서버로부터 받은 정보를 이용하여 페이지를 만들어 사용자에게 보여줌스케치, 도표, 디자인이(가) 표시된 사진

자동 생성된 설명

우리는 서버에 요청했다고 하고 클라이언트에 응답했다고 한다.

물론 브라우저가 아니라도 터미널로도 응답을 확인할 수 있음텍스트, 스크린샷, 폰트, 블랙이(가) 표시된 사진

자동 생성된 설명

응답이 오긴 왔음. 대신 터미널은 해석해서 가시화 할 수 있는 기능이 없음. 브라우저는 렌더링 기능을 통해 요청된 페이지를 잘 보여줌텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

서버와 클라이언트 간에 어떤 요청과 응답이 오갔는지는 개발자도구를 통해 확인할 수 있음. 내용을 살펴보면 특별한 양식이 있는 것을 확인할 수 있는데, 그 양식을 HTTP라고 부름텍스트, 폰트, 도표, 스크린샷이(가) 표시된 사진

자동 생성된 설명

HTTP

HTTP를 알아야 하는 이유는 HTTP 형식에 맞는 요청을 받을 때 HTTP 형식에 맞는 응답을 내려주는 애플리케이션을 만들기 위해서임

텍스트, 폰트, 도표, 스크린샷이(가) 표시된 사진

자동 생성된 설명

HTTP Request Line

텍스트, 폰트, 스크린샷, 라인이(가) 표시된 사진

자동 생성된 설명a) 메서드

요청의 종류를 나타냄. 여기서는 GET, 즉 ‘URI’에서 지정한 정보를 보내주세요 라는 의미가 됨. 메서드에는 그 밖에도 몇 가지가 정의되어 있는데, 웹 브라우저에서 웹 서버로 송신되는 요청의 대부분은 GET 메서드에 따른 요청임

b) URI (Uniform Resource Identifier)

GET 메서드는 단순히 정보를 주세요, 라는 의미에 불과하며 URI는 무엇을 원하는가를 나타냄. 일단 URL과 URI는 거의 같은 것이라고 생각해도 무방

HTTP response

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

1. 상태라인

HTTP 요청과 마찬가지로 HTTP 응답에서도 첫번째 줄이 가장 중요하며, 이것을 상태라인이라고 한다. HTTP 상태라인도 구성은 간단하다. HTTP 버전과 상태 코드, 응답 구문 세 부분으로 나뉜다.

HTTP/1.1 200 OK

HTTP 버전은 요청에서와 마찬가지로 사용하는 프로토콜의 버전을 나타냄. 중요한 것은 상태코드로 이 부분을 보면 요청이 성공했는지 실패했는지 쉽게 알 수 있다는 것. 200은 요청이 성공해 정상적인 응답이 들어왔음을 나타내는 상태코드

200: OK / 201: Created(자원이 잘 생성됨) / 400번대: 당신의 요청이 잘못되었음

**WEB 서버 구조와 WAS의 정체 | Web 서버, WAS, DB가 하는 일들**

WAS란? Web Application Server

WAS가 무엇인지 알려면 웹 서비스가 어떻게 이루어지는지 알아야함. 내가 크롬을 통해서 네이버를 들어가는데 네이버에서는 어떻게 하고 있길래 지금 내가 네이버 페이지에 들어가서 메일도 보고 카페도 들어가는지 구조를 알고 있어야 함

구조는 크게 웹서버를 3가지로 나누어 표현할 수 있음

1. 우리가 있음. 컴퓨터를 쓰면서 인터넷을 하는 주체
2. 웹 서버
3. 웹 서버 뒤에 WAS가 있음
4. 그 뒤에 db라고 부르는 Database Server가 존재

우리는 그냥 웺 서버에 접속하는 것이 아니라 실제로는 이 서버에 있는 3가지 중에서 웹서버랑 통신하고 있어었던 것. 웹 서버한테 ‘데이터 혹은 어떤 파일을 줘!’’ 라고 하면 웹 서버는 그것을 응답해주는 역할을 함. 즉 우리는 이 웹 서버랑만 이야기 한다는 점!

웹 서버는 실제로 보면 매우 단순

사용하는 사람, 이용자가 웹서버한테 떙땡 파일 주세요! 라고 하면 그 웹서버는 땡떙 파일을 오케이 알겠어 잠깐만.. 하고 찾아봐 그래서 찾으면 그걸 요청한 사용자에게 갖다줌. 이게 끗! 사용자는 어떤 파일을 웹 서버에게 달라고 요청한다. 웹서버는 요청받은 파일을 응답한다.

그런데 생각해보면, 네이버에 우리가 로그인하면 박소현 님, 안녕하세요 이런 페이지가 뜸. 그럼 이 파일 및 페이지도 결국 웹 서버에 있었다는 소리. 그런데 네이버를 쓰는 사람이 나 혼자일까? ㄴㄴ 수십 수백만 명임. 그렇다면 수십 수백만 개의 페이지가 있어야 하나? 말도 안됨. 즉 네이버는 하나의 페이지만 만들어 놓고, 그 페이지의 이름만 바꾸는 식으로 구성. 그럼 지금 들어온 사람의 이름을 가져다가 페이지에 적어서 응답해주면 됨.

즉 기존에 있던 파일을 갖다주면 안됨. 여기서 WAS의 역할이 나옴

웹서버는 머리를 쓰지 않는, 생각을 하지 않는 친구임. 요청받으면 파일을 받아 응답해주는 게 끝. 그런데 결과적으로 웹 서버가 머리를 쓰는 일이 생기기 마련. 머리를 쓰는 일을 웹 서버 뒤에 있는 WAS가 해주는 것. 웹 서버를 구축한다, 웹 서버에서 돌아가는 페이지를 만든다 = 웹 애플리케이션 및 웹 프로그램을 만드는 행위. 웹 프로그램을 웹 서버에 올린다 이 말이 되는 것!

우리가 웹 서버에 땡떙 파일 주세요! 하면 웹 서버가 땡떙 파일 여기 있어! 하는 구조

만약 머리를 써야 하는 기능 및 서비스가 있고 그런 요청을 받으면 웹 서버는 자기 머리를 쓰는 대신 WAS에게 해당 요청을 토스함. WAS는 그 요청을 처리해서 응답한 결과를 웹서버에게 줌. 웹 서버는 그 응답을 사용자에게 그대로 전달

그래서 WAS는 웹 서버가 넘긴 일을 처리하고 응답, 웹 서버는 받은 파일을 사용자에게 준다 🡺 웹 서비스가 동작하는 궁극적 방식

그래서 WAS는 웹 서버한테 오는 내용을 처리하는데 이떄 뒤에 있는 데이터베이스랑 이야기를 하기도 함. 이 데이터베이스는 데이터가 저장되는 곳인데, 저장된 곳에서 데이터를 가져와서 처리를 함. 대표적인 경우는 로그인이 있음.

우리가 아이디랑 비밀번호를 웹 서버에 보냄. 웹서버는 머리를 쓰기 싫음. 그래서 아이디랑 비밀번호를 was에 보냄. WAS는 받은 아이디와 비밀번호를 가지고 DB로부터 데이터를 가져와서 비교를 함. WAS는 이 DB에서 데이터를 가져온 다음에 사용자가 입력한 아이디와 비밀번호를 비교해서 페이지를 만듦. 비교결과 일치하면 로그인을 성공한 페이지, 불일치하면 로그인을 실패한 페이지. 그러고 나면 웹 서버(머리쓰기 싫고 전달만 해주는 애)는 그 로그인 성공 or 실패 페이지를 사용자에게 응답해줌

Spring MVC

HTTP만 안다고 해서 웹 애플리케이션을 만들 수는 없음

클라이언트에서 서버로 요청을 보내기 위해서는 수많은 작업이 필요함

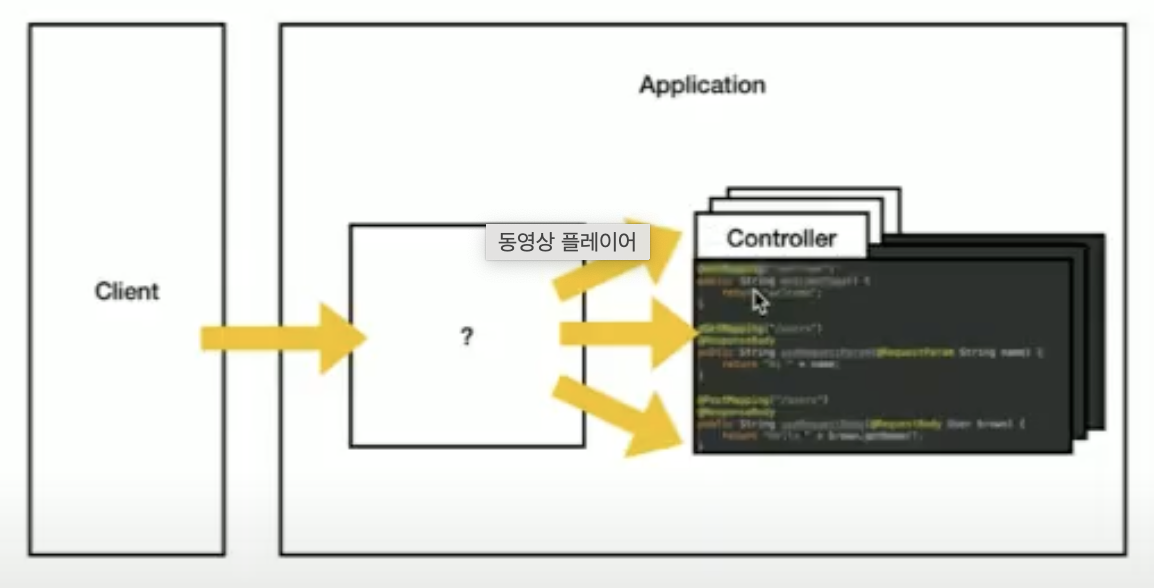
그 부분들을 다 구현하면 실제 비즈니스 로직보다 더 많은 부분의 코드를 작성해야 할 지도 모름

만약 부수적인 부분을 **프레임워크**를 통해 해소한다면 개발자들은 비즈니스로직에 조금 더 집중할 수 있게 될 것

웹 애플리케이션 서버의 **로직**을 우리는 개발하고자 함

스프링 MVC를 도식화하면 클라이언트와 소통하는 모듈로 설명할 수 있음.

클라이언트로부터 온 요청을 처리할 로직에 따라서 처리한 후 응답을 하는 역할을 가지고 있음



이를 위해 요청에 따라 처리하는 로직을 분기하는 기능도 있고, 요청에 담긴 정보를 객체로 사용할 수 있게 도와주는 기능도 있고, 클라이언트가 필요한 형태로 응답을 만들어주는 기능도 있음