강의 필기

섹션1 - 프로젝트 생성

예전에는 spring을 처음부터 다 짜면서 프로젝트 생성

-> but 요즘에는 다 스프링 부트로 스프링 프로젝트 생성

start.spring.io <- 여기서 생성하면 됨

project 종류

gradle <-

maven <- 필요한 라이브러리를 가져오고 빌드하는 라이프사이클까지 모두 관리해줌

과거에는 maven, 요즘에는 gradle을 사용함

group은 보통 회사명

artifact가 project name

dependency는

web <- web project이므로

thymeleaf <- html을 만들어주는 template engine

project open은 project 폴더 내에 build.gradle을 선택해서 open하면 됨

10

섹션1 - view 환경설정

resources/static/index.html <- 이 html이 welcome page가 됨(domain만 들어왔을 때, 첫 화면)

++ spirng은 web과 app을 포함한 방대하게 사용 -> spirig boot는 이러한 spring을 감싸서 편리하게 사용할 수 있게 도와줌

=> 내용이 굉장히 방대해서 필요한 것을 찾는 능력이 중요함

-> sping.io -> project에서 spring boot -> 맞는 버전의 refence document -> 검색

위에 검색을 할 줄 알아야 함

11

index.hmel은 그냥 정적 페이지

thymeleaf 템플릿 엔진을 통해 html을 변화 가능

12

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

HelloController와 hello.html을 추가한 후 동작과정

웹 브라우저에서 localhost:8080/hello를 호출하면 스프링부트는 내장한 톰켓 서버에서 받아서 /hello와 get method인 것을 확인 -> spring container에서 @getMapping인 method를 찾아 실행시킴(이때 메소드를 실행시킬 때 spring이 model이란 것을 만들어서 파라미터로 넘겨줌)

-> 넘어온 model에 우리는 attribute를 추가했음(key가 data, value는 hello!!)

++ 여기서 value는 바뀔 수 있을 것(예를 들어 DB에서 조회해서 가져온 값을 value에 넣거나 함)

이후에 Controller 내에서 return “hello”를 하면 spring booter는 resource아래 template아래에 hello.html을 찾아서 rendering을 해 줌(thymeleaf 템플릿 엔진이 처리해주면서) <- 이걸 viewResolver가 해주는 것

섹션1 - 빌드하고 실행하기

콘솔에서

./gradlew build <- build

cd build.libs

java -jar hello-spring-0.0.1-SNAPSHOT.jar <- 실행

이후 서버 배포할 때는 위 실행파일만 서버에 넣어놓고 java -jar로 실행하면 배포가 됨

섹션 2

-정적 콘텐츠

웹을 개발한다는 것은 크게 3가지 방법이 존재

1. 정적 컨텐츠, 즉 서버에서 하는 것 없이 file을 웹 브라우저에 내려주는 것

2. MVC와 템플릿 엔진 <- 가장 많이 하는 방식

HTML을 그냥 주는 것이 아니라 서버에서 프로그래밍을 해서 동적으로 바꿔서 내리는 것

(이런 걸 템플릿 엔진이라고 함)

=> 이런 것을 하기 위해 MVC 이 3가지를 제어하면서 동작하는 것

3. API

안드로이드나, 아이폰과 같은 모바일 client를 대상으로 개발하면 json이라는 format으로 보통 내려줌 <- 이런게 API 방식

요즘에는 server는 data만 내려주면 client가 data를 활용해 화면을 기르고 처리하는 등의 구조에서도 API를 자주 활용

server끼리 통신할 때도 API를 자주 활용(server끼리는 HTML을 보낼 필요 없이 data만 보내면 되므로)

현재 내가 이해한 거로는

1. 정적 컨턴체 <- 정적인 HTML

2. MVC와 템플릿 엔진 <- 템플릿을 활용해 동적으로 HTML 제어

3. API <- HTML이 아닌 data만 주고받는 방식(JSON, XML 등)

13

스프링부트는 정적 컨텐츠 기능을 자동으로 제공

/main/resources/static

<- 여기가 정적 컨텐츠

여기에 file을 넣으면 url/{filename}으로 들어가면 해당 file이 display 됨

/main/resources/template

<- 여기가 MVC로 관리되는 템플릿 엔진 HTML

14

정적 컨텐츠 전체 흐름

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

웹 브라우저에서 localhost:8080/hello-static.html을 입력 -> 내장 톰켓 서버가 받음 -> spring한테 해당 정보를 넘김 -> spring은 controller쪽에서 먼저 hello-static으로 등록된 controller가 있는지 탐색 -> 현재는 Hello 밖에 없으니 그 다음에 resource/아래에 hello-static.html을 탐색 -> 있으니 반환해서 전송

(resource보다 controller가 더 우선순위가 높아 먼저 controller부터 탐색하는 것)

-MVC와 템플릿 엔진

MVC는 model, view, controller

예전에는 model 하나에 전부 통합되어 있었지만, 현재는 MVC모델을 거의 사용

view는 화면을 그리는 역할

controller와 model은 내부적인 로직이나 처리 담당

텍스트, 도표, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명

전체적인 동작은 위와 동일

url을 넘기면 내장 톰캣 서버가 확인후 spring에게 hello-mvc가 왔다고 알림

-> hellocontroller에 hello-mvc가 get방식으로 정의되어 있는 것을 확인

=> 해당 method를 호출해서 실행

-> return인 hello-template와 파라미터로 온 값을 model에 대입 후 viewResolver에게 넘겨줌

-> viewResolver가 view를 찾아주고, template 엔진을 연결시켜줌

-> template engine이 rendering을 해서 변환을 한 뒤에 HTML을 웹 브라우저에 넘겨줌

15

사실상 정적컨텐츠는 그냥 HTML을 가지고 있다가 넘기는 것

그 이외에는 MVC 혹은 API 2가지 방법이 있다고 생각하면 됨

MVC는 template 엔진으로 변환 후 HTML로 내려줌

API는 data를 바로 내려줌

텍스트, 도표, 라인, 그래프이(가) 표시된 사진

자동 생성된 설명

동작방식은 전체적으로 비슷함

hello-api가 오면 톰캣서버가 보고 spring한테 넘겨줌

-> spring은 controller에 있는 것을 확인함 ++ @ResponseBody annotation 붙어있는 것을 확인

(@ResponseBody가 없으면 ViewResolver한테 전달해 template를 만들라고 지시)

=> Http응답이니 그대로 data를 http reply로 넘겨야겠다고 판단

-> 여기서 data가 문자면 그냥 넘겨서 display

-> data가 객체라면 기본 default는 json방식으로 data를 만들어서 http 응답으로 반환

(일단 Http Message Converter가 동작을 함 -> 여기서 return 값이 단순 문자면 StringConverter가 동작, 객체면 JsonConverter가 동작)

섹션3 - 백엔드 개발

비즈니스 요구사항 정리

-> 가장 쉬운 것으로 할 것

=> 회원 ID, 이름 data만 존재 -> 등록, 조회만 가능

(아직 DB가 선정되지 않았고, 일단 개발 먼저 한다는 상황을 가정)

(DB의 경우 성능을 중시할지, RDB로 할지, noSQL처럼 간단하게 할지 등에 따라 결정됨)

도표, 스크린샷, 직사각형, 라인이(가) 표시된 사진

자동 생성된 설명

일반적인 웹은 다음과 같은 구조를 가짐

컨트롤러: MVC에서 controller와 같은 역할

서비스: 서비스의 핵심 비즈니스 로직(중복회원가입 불가 등)

리포지토리: 비즈니스 도메인 객체를 가지고 서비스가 동작하도록 구현한 객체

도메인: 데이터베이스에 저장하고 관리되는 도메인 객체

텍스트, 스크린샷, 도표, 라인이(가) 표시된 사진

자동 생성된 설명

회원 서비스와 회원 리포지토리가 존재

여기서 리포지토리는 interface로 생성하고, 구현체는 우선 memory 구현체로 생성

(아직 DB가 결정되지 않았는데, 개발은 시작해야 하기에 간단하게 메모리 기반 데이터 저장소 사용)

intellij 단축키

Alt + insert 자주 활용됨 <- 파일 혹은 메소드 생성기

ctrl + shift + enter도 자주 활용 <- 스마트 자동완성

alt + enter도 자주 사용 <- import할 거나 문제 자동 해결