20220428 - Swagger

Created	@2022년 4월 27일 오후 10:19
Tags	Personal Study
Property	

✅ Swagger

- 프론트 개발자와 백 개발자 사이 개발 상황 변화에 따른 API의 변동 사항 발생 시 이때마다 문서를 수행하는 번거로움이 발생 → 이를 해결하기 위해 Swagger 사용
- 간단한 설정으로 프로젝트의 API 목록을 웹에서 확인 및 테스트 할 수 있게 해주는 라이브러리
- Swagger를 사용하면 컨트롤러(RestController)에 정의되어 있는 모든 URL을 바로 확인 가능
- API 목록 뿐 아니라 API의 명세 및 설명도 볼 수 있음
- API 직접 테스트 가능

```
<!-- swagger를 위한 의존성 추가 -->
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-boot-starter -->
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-boot-starter</artifactId>
  <version>3.0.0</version>
</dependency>
```

```
@Bean
public Docket api() {
  return new Docket(DocumentationType.SWAGGER_2).consumes(getConsumeContentTypes()).produces(getProduceContentTypes())
        .apiInfo(apiInfo()).groupName(version).select()
        .apis(RequestHandlerSelectors.basePackage("com.ssafy.guestbook.controller"))
        .paths(regex("/admin/.*")).build()
        .useDefaultResponseMessages(false);
}
```

- `@EnableSwagger2`: Swagger2 버전 활성화
- `Docket`: Swagger 설정을 할 수 있게 도와주는 클래스
- `select()`: ApiSelectorBuilder를 생성하여 apis()와 path()를 사용할 수 있게 해줌
- `apiInfo()`: 제목, 설명 등 문서에 대한 정보들을 설정하기 위해 호출
- `groupName()`: Docket Bean이 한 개일 경우 생략해도 상관없으나, 둘 이상일 경우 충돌을 방지하기 위해 설정
- `apis()`: API 스펙이 작성되어 있는 패키지를 지정한다. 즉, 컨트롤러가 존재하는 패키지를 basePackage로 지정하여 해당 패키지에 존재하는 API를 문서화한다.
- `RequestHandlerSelectors.basePackage(String packageName)`: Swagger를 적용할 클래스의 패키지명
- `paths()`: `apis()`로 선택되어진 API 중 특정 path 조건에 맞는 API들을 다시 필터링하여 문서화한다. `PathSelectors.any()`로 설정하면 패키지 안에 모든 API를 한 번에 볼 수 있다.
- `useDefaultResponseMessages()`: false로 설정하면 Swagger에서 제공해주는 응답코드(200 등)에 대한 기본 메시지를 제거해줌
- `ParameterBuilder`와 `globalOperationParameters()`를 이용하여 Swagger의 전체 API에서 보여줄 파라미터를 설정할 수도 있음
- `globalResponseMessage()`로 공통 응답 메시지를 작성할 수도 있음

Controller

- `@Api`: 해당 클래스가 Swagger 리소스라는 것을 명시해줌
  - `value`: 사용자 지정 이름. `tags` 사용시 무시됨
  - `tags`: 여러 개의 태그를 정의할 수 있음
- `@ApiOperation`: 한 개의 Operation을 선언
  - `value`: API에 대한 요약 작성

    - `notes`: API에 대한 자세한 설명
- `@ApiParam`: 파라미터에 대한 정보 명시
    - `name`: 파라미터 이름 작성
    - `value`: 파라미터 설명 작성
    - `required`: 필수 파라미터면 `true`, 아니면 `false`
- `@ApiResponse`: 해당 메서드의 응답에 대한 설명 작성(응답 코드, 메시지)

```
@ApiOperation(value = "this is test!", notes = "note!!!!!")
@PostMapping("/test")
public String test(@ApiParam(name = "first param", value = "first value", required = true) String input,
                   @ApiParam(name = "second param", value = "second value", required = false) String input2) {
    return "test";
}
```

DTO

- `@ApiModel`: 해당 모델을 설명할 때 사용. VO, DTO, Entity 등에서 사용
    - `description`: 모델 설명
- `@ApiModelProperty`: 모델 내의 필드를 설명할 때 사용
    - `required`: 필수 여부
    - `value`: 필드 이름
    - `example`: 필드 예시
    - `hidden`: 필드를 숨기는 여부