개발한 rest api를 편리하게 문서화 해주고, 이를 통해서 관리 및 제3의 사용자가 편리하게 api를 호출해보고 테스트할 수 있는 프로젝트 🡺 swagger

일반적인 문서화는 도구 > javaDoc 생성.

알아야할 키워드 : @Api, @ApiOperation, @ApiParam, @ApiResponses, @ApiResponse, @ApiModel, @ApiModelProperty



**필요 구성 및 준비 :**

Build.gradle //spring-doc말고 spring-fox  
implementation ‘io.springfox:springfox-swagger2:2.9.2’  
implementation ‘io.springfox:springfox-swagger-ui:2.9.2’

Src>main>java>org.scoula>config>SwaggerConfig.java//스웨거 전용 config  
@Configuration @EnableSwagger2  
public class SwaggerConfig{  
 private final String API\_NAME = "Board API"; //api에 대한 기본 정보들  
 private final String API\_VERSION = "1.0";  
 private final String API\_DESCRIPTION = "Board API 명세서";  
  
 private ApiInfo apiInfo( ) { //api에 대한 기본 정보 지정  
 return new ApiInfoBuilder( )  
 .title(API\_NAME)  
 .description(API\_DESCRIPTION)  
 .version(API\_VERSION)  
 .build( );  
 }  
 @Bean //api라고하는 빈을 등록  
 public Docket api( ) {  
 return new Docket(DocumentationType.SWAGGER\_2)  
 .select( ) //@RestController가 붙은 모든 컨트롤러를 대상으로해서 api를 만들겠다.  
 .apis(RequestHandlerSelectors.withClassAnnotation(RestController.class))  
 .paths(PathSelectors.any( )) //제한을 두지 않겠다.  
 .build( )  
 .apiInfo(apiInfo( )); //api에 대한 기본 정보를 구성  
 }  
}

WebConfig.java  
…  
@Override protected Class<?>[ ] getServletConfigClasses( ){  
 return new Class[ ] {ServletConfig.class, SwaggerConfig.class };//SwaggerConfig.class 추가  
}  
//스프링의 dispatcher servlet이 담당할 url 매핑 패턴. 요청 경로 추가.  
@Override protected String[ ] getServletMappings( ){  
 return new String[ ]{  
 “/”,  
 "/swagger-ui.html", //문서를 보기 위한 인덱스 페이지  
 "/swagger-resources/\*\*", //스웨거 정적파일들  
 "/v2/api-docs", //관련 경로  
 "/webjars/\*\*" //관련 경로  
 };  
}

ServletConfig.java //스웨거의 정적 리소스 4가지 경로 추가  
…  
@Override  
public void addResourceHandlers(ResourceHandlerRegistry registry) {  
…  
 registry.addResourceHandler("/swagger-ui.html")   
 .addResourceLocations("classpath:/META-INF/resources/"); // Swagger UI 리소스를 위한 핸들러 설정  
 registry.addResourceHandler("/webjars/\*\*")  
 .addResourceLocations("classpath:/META-INF/resources/webjars/");// Swagger WebJar 리소스 설정  
 registry.addResourceHandler("/swagger-resources/\*\*")  
 .addResourceLocations("classpath:/META-INF/resources/");// Swagger 리소스 설정  
 registry.addResourceHandler("/v2/api-docs")  
 .addResourceLocations("classpath:/META-INF/resources/");  
}

**사용법 :**

컨트롤러의 전체 설명  
@RestController @RequestMapping(“/api/board”) @RequiredArgsConstructor @Log4j2  
@Api(tags=”해당 api(컨트롤러) 전체 타이틀 및 설명”)  
public class ???Controller{ … }

컨트롤러의 메서드 == api 엔드포인트에 대한 설명  
@ApiOperation(value=”api 엔드포인트(기능) 이름”, notes=”설명”)  
@GetMapping(“”)//엔드포인트 url 경로  
public ResponseEntity< List<BoardDTO> > getList( ){ … }

API엔드 포인트에 대한 상세 설명. 매개변수와 반환.  
@ApiOperation(value=”api 엔드포인트(기능) 이름”, notes=”설명”)  
@ApiResponses( { //반환될 수 있는 경우들에 대해서 설명  
 @ApiResponse(code=응답코드, message=”설명”, response=???DTO.class),//단일이든 컬렉션이든  
 @ApiResponse(code=응답코드2, message=”설명2”)  
} )  
@GetMapping(“”)//엔드포인트 url 경로  
public ResponseEntity< List<BoardDTO> > getList( //@PathVariable, @RequestBody앞에서 설정 가능  
 @ApiParam(value=”매개변수에 대한 설명”, required=true, example=”예시값”)   
 @PathVariable Long no,  
 @ApiParam(value=”매개변수에 대한 설명2”, required=false, example=”예시값2”)   
 @RequestBody BoardDTO board)  
 ){ … }

DTO모델에 대한 문서화  
@Data @NoArgsConstructor @AllArgsConstructor @Builder  
@ApiModel( description=”게시글DTO자체에 대한 설명”)  
public class BoardDTO{  
 @ApiModelProperty(value”필드에 대한 설명”)  
 private Long no;  
}