

2025년 상반기 K-디지털 트레이닝

# Swagger-ui를 활용한 문서 자동화

---

[KB] IT's Your Life

## Swagger-ui를 활용한 문서 자동화

### ✓ Swagger란 ?

- 개발한 Rest API를 문서화
- 문서화된 내용을 통해 관리 & API 호출을 통한 테스트 가능
- API Test할 때 많이 사용되는 PostMan, Talend API Tester와 비슷

## Swagger-ui를 활용한 문서 자동화

### ✓ Swagger 라이브러리의 종류 2가지

- Spring-Fox\*
- Spring-Doc

### ✓ 의존성

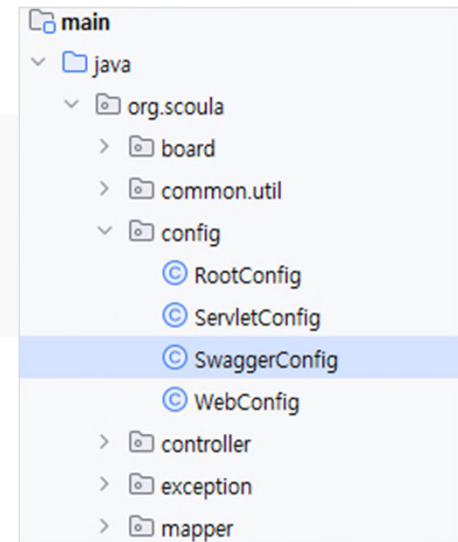
#### ○ build.gradle

implementation 'io.springfox:springfox-swagger2:2.9.2'

implementation 'io.springfox:springfox-swagger-ui:2.9.2'

## config.SwaggerConfig.java

```
@Configuration  
@EnableSwagger2  
public class SwaggerConfig {  
  
}
```



## config.SwaggerConfig.java

```
@Configuration
@EnableSwagger2
public class SwaggerConfig {
    private final String API_NAME = "Board API";
    private final String API_VERSION = "1.0";
    private final String API_DESCRIPTION = "Board API 명세서";

    private ApiInfo apiInfo() {
        return new ApiInfoBuilder()
            .title(API_NAME)
            .description(API_DESCRIPTION)
            .version(API_VERSION)
            .build();
    }

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .select()
            .apis(RequestHandlerSelectors.withClassAnnotation(RestController.class))
            .paths(PathSelectors.any())
            .build()
            .apiInfo(apiInfo());
    }
}
```

@RestController가 붙은 모든 컨트롤러를 대상으로 함

## config.WebConfig.java

```
public class WebConfig extends AbstractAnnotationConfigDispatcherServletInitializer {
    ...

    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class[] { ServletConfig.class, SwaggerConfig.class };
    }

    // 스프링의 FrontController인 DispatcherServlet이 담당할 url 매핑 패턴, /: 모든 요청에 대해 매핑
    @Override
    protected String[] getServletMappings() {
        return new String[]{
            "/",
            "/swagger-ui.html",
            "/swagger-resources/**",
            "/v2/api-docs",
            "/webjars/**"
        };
    }
    ...
}
```

## config.ServletConfig.java

```
public class ServletConfig implements WebMvcConfigurer {

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry
            .addResourceHandler("/resources/**") // url이 /resources/로 시작하는 모든 경로
            .addResourceLocations("/resources/"); // webapp/resources/경로로 매핑

        // Swagger UI 리소스를 위한 핸들러 설정
        registry.addResourceHandler("/swagger-ui.html")
            .addResourceLocations("classpath:/META-INF/resources/");

        // Swagger WebJar 리소스 설정
        registry.addResourceHandler("/webjars/**")
            .addResourceLocations("classpath:/META-INF/resources/webjars/");

        // Swagger 리소스 설정
        registry.addResourceHandler("/swagger-resources/**")
            .addResourceLocations("classpath:/META-INF/resources/");

        registry.addResourceHandler("/v2/api-docs")
            .addResourceLocations("classpath:/META-INF/resources/");
    }
    ...
}
```



## Swagger-ui를 활용한 문서 자동화

✓ <http://localhost:8080/swagger-ui.html>

The screenshot shows the Swagger UI interface for the Board API. At the top, there is a green header with the Swagger logo and a dropdown menu labeled "Select a spec" with "default" selected. Below the header, the title "Board API" is displayed with a version badge "1.0". Underneath, the base URL is shown as "[ Base URL: localhost:8080/ ]" and a link to the API docs is provided: <http://localhost:8080/v2/api-docs>. A description "Board API 명세서" is also present. The main section is titled "board-controller Board Controller" and lists five API endpoints:

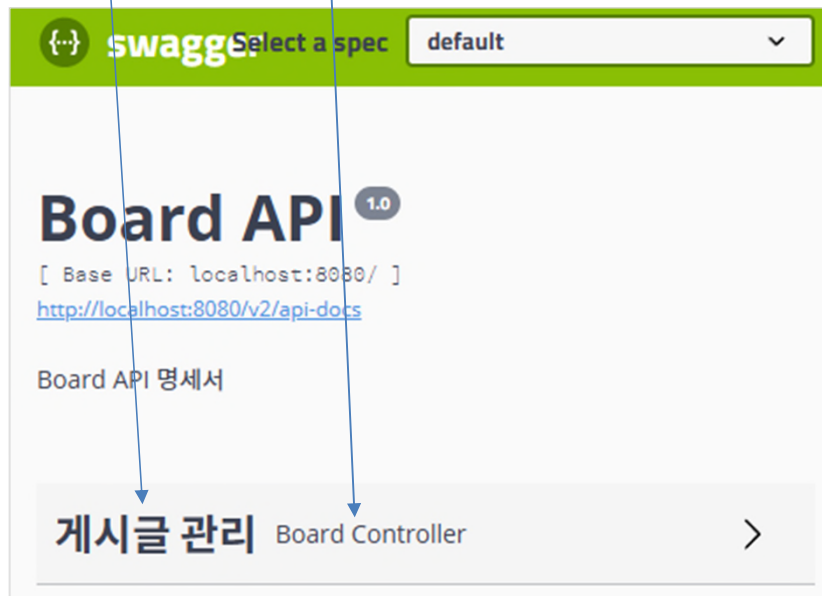
- GET** /api/board getList
- POST** /api/board create
- GET** /api/board/{no} get
- PUT** /api/board/{no} update
- DELETE** /api/board/{no} delete

### ✓ RestController에 정보 설정하기

- @Api(tags="API 타이틀")

@Api(tags = "게시글 관리")

public class BoardController { ... }



### controller.BoardController.java

```
@RestController
@RequestMapping("/api/board")
@RequiredArgsConstructor
@Log4j2
@Api(tags = "게시글 관리")
public class BoardController {
    private final BoardService service;
    ...
}
```

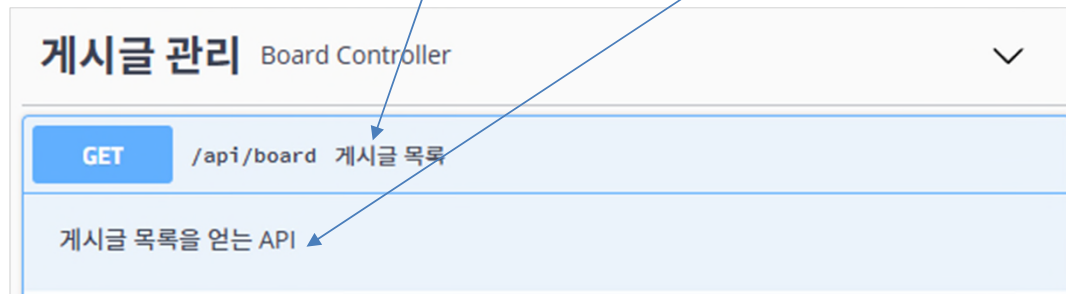
### ✓ API 엔드포인트 설명

- @ApiOperation(value = "api 명", notes = "설명")

@ApiOperation(value = "게시글 목록", notes = "게시글 목록을 얻는 API")

@GetMapping("")

public ResponseEntity<List<BoardDTO>> getList() { ... }



### controller.BoardController.java

```
...
@Api(tags = "게시글 관리")
public class BoardController {
    private final BoardService service;

    @ApiOperation(value = "게시글 목록", notes = "게시글 목록을 얻는 API")
    @GetMapping("")
    public ResponseEntity<List<BoardDTO>> getList() {
        return ResponseEntity.ok(service.getList());
    }
    ...
}
```

### ✓ API 엔드포인트 상세 설명

#### ○ @ApiParam


- 엔드포인트 파라미터 설명
- @PathVariable, @RequestBody 앞에 설정
- 주요 속성
  - value 속성: 엔드포인트의 간략한 설명,
  - required 속성: 필수 여부
  - example 속성: 파라미터의 예시 값 제공

## Swagger-ui를 활용한 문서 자동화

### controller.BoardController.java

```
@ApiOperation(value = "상세정보 얻기", notes = "게시글 상세 정보를 얻는 API")
@GetMapping("/{no}")
public ResponseEntity<BoardDTO> get(
    @ApiParam(value = "게시글 ID", required = true, example = "1")
    @PathVariable Long no) {
    return ResponseEntity.ok(service.get(no));
}
```

숫자 타입인 경우 example을 숫자값으로 지정해줘야 함  
지정하지 않은 경우 NumberFormatException 예외발생



GET /api/board/{no} 상세정보 얻기

게시글 상세 정보를 얻는 API

Parameters Try it out

Name	Description
no * required integer (\$int64) (path)	게시글 ID

# Swagger-ui를 활용한 문서 자동화

## controller.BoardController.java

```
@ApiOperation(value = "게시글 생성", notes = "게시글 생성 API")
@PostMapping("")
public ResponseEntity<BoardDTO> create(
    @ApiParam(value = "게시글 객체", required = true)
    @RequestBody BoardDTO board) {
    return ResponseEntity.ok(service.create(board));
}
```

**POST** /api/board 게시글 생성

게시글 생성 API

**Parameters** [Try it out](#)

Name	Description
<b>board</b> * required (body)	게시글 객체

Example Value

Model

```
{
  "attaches": [
    {
      "bno": 0,
      "contentType": "string",
      "fileSize": "string",
      "filename": "string",
      "no": 0,
      "path": "string",
      "regDate": "2025-01-20T04:08:46.797Z",
      "size": 0
    }
  ],
  "content": "string",
  "files": [
    null
  ],
  "no": 0,
  "regDate": "2025-01-20T04:08:46.797Z",
  "title": "string",
  "updateDate": "2025-01-20T04:08:46.797Z",
  "writer": "string"
}
```

Parameter content type

application/json



### ✓ API 엔드포인트 상세 설명

#### ○ @ApiResponses(value = {}) :

- @ApiResponse의 배열
  - code 속성: 응답 코드
  - message 속성: 값의 의미설명
  - response 속성: 응답 객체 class 정보

```
@ApiResponses(value = {  
    @ApiResponse(code = 200, message = "성공적으로 요청이 처리되었습니다.", response = BoardDTO.class),  
    @ApiResponse(code = 400, message = "잘못된 요청입니다."),  
    @ApiResponse(code = 500, message = "서버에서 오류가 발생했습니다.")  
})
```

### controller.BoardController.java

```
@ApiOperation(value = "게시글 목록", notes = "게시글 목록을 얻는 API")
@ApiResponses(value = {
    @ApiResponse(code = 200, message = "성공적으로 요청이 처리되었습니다.", response = BoardDTO.class),
    @ApiResponse(code = 400, message = "잘못된 요청입니다."),
    @ApiResponse(code = 500, message = "서버에서 오류가 발생했습니다.")
})
@GetMapping("")
public ResponseEntity<List<BoardDTO>> getList() {
    return ResponseEntity.ok(service.getList());
}
```

# Swagger-ui를 활용한 문서 자동화

Responses Response content type \*/\*

Code	Description
200	성공적으로 요청이 처리되었습니다.

Example Value | Model

```
{
  "attaches": [
    {
      "bno": 0,
      "contentType": "string",
      "fileSize": "string",
      "filename": "string",
      "no": 0,
      "path": "string",
      "regDate": "2025-01-20T03:55:06.154Z",
      "size": 0
    }
  ],
  "content": "string",
  "files": [
    null
  ],
  "no": 0,
  "regDate": "2025-01-20T03:55:06.154Z",
  "title": "string",
  "updateDate": "2025-01-20T03:55:06.154Z",
  "writer": "string"
}
```

400	잘못된 요청입니다.
401	Unauthorized
403	Forbidden
404	Not Found
500	서버에서 오류가 발생했습니다.

### controller.BoardController.java

```
@ApiOperation(value = "상세정보 얻기", notes = "게시글 상세 정보를 얻는 API")
@ApiResponses(value = {
    @ApiResponse(code = 200, message = "성공적으로 요청이 처리되었습니다.", response = BoardDTO.class),
    @ApiResponse(code = 400, message = "잘못된 요청입니다."),
    @ApiResponse(code = 500, message = "서버에서 오류가 발생했습니다.")
})
@GetMapping("/{no}")
public ResponseEntity<BoardDTO> get(
    @ApiParam(value = "게시글 ID", required = true, example = "1")
    @PathVariable Long no) {
    return ResponseEntity.ok(service.get(no));
}
```

Responses Response content type \*/\*

Code	Description
200	OK

Example Value | Model

```
{
  "attaches": [
    {
      "bno": 0,
      "contentType": "string",
      "fileSize": "string",
      "filename": "string",
      "no": 0,
      "path": "string",
      "regDate": "2025-01-20T04:08:56.492Z",
      "size": 0
    }
  ],
  "content": "string",
  "files": [
    null
  ],
  "no": 0,
  "regDate": "2025-01-20T04:08:56.492Z",
  "title": "string",
  "updateDate": "2025-01-20T04:08:56.492Z",
  "writer": "string"
}
```

401	Unauthorized
403	Forbidden
404	Not Found

### controller.BoardController.java(전체 코드)

```
@RestController
@RequestMapping("/api/board")
@RequiredArgsConstructor
@Log4j2
@Api(tags = "게시글 관리")
public class BoardController {
    private final BoardService service;

    @ApiOperation(value = "게시글 목록", notes = "게시글 목록을 얻는 API")
    @ApiResponses(value = {
        @ApiResponse(code = 200, message = "성공적으로 요청이 처리되었습니다.", response = BoardDTO.class),
        @ApiResponse(code = 400, message = "잘못된 요청입니다."),
        @ApiResponse(code = 500, message = "서버에서 오류가 발생했습니다.")
    })
    @GetMapping("")
    public ResponseEntity<List<BoardDTO>> getList() {
        return ResponseEntity.ok(service.getList());
    }
}
```

### controller.BoardController.java(전체 코드)

```
@ApiOperation(value = "상세정보 얻기", notes = "게시글 상세 정보를 얻는 API")
@ApiResponses(value = {
    @ApiResponse(code = 200, message = "성공적으로 요청이 처리되었습니다.", response = BoardDTO.class),
    @ApiResponse(code = 400, message = "잘못된 요청입니다."),
    @ApiResponse(code = 500, message = "서버에서 오류가 발생했습니다.")
})
@GetMapping("/{no}")
public ResponseEntity<BoardDTO> get(
    @ApiParam(value = "게시글 ID", required = true, example = "1")
    @PathVariable Long no) {
    return ResponseEntity.ok(service.get(no));
}
```

### controller.BoardController.java(전체 코드)

```
@ApiOperation(value = "게시글 생성", notes = "게시글 생성 API")
@ApiResponses(value = {
    @ApiResponse(code = 200, message = "성공적으로 요청이 처리되었습니다.", response = BoardDTO.class),
    @ApiResponse(code = 400, message = "잘못된 요청입니다."),
    @ApiResponse(code = 500, message = "서버에서 오류가 발생했습니다.")
})
@PostMapping("")
public ResponseEntity<BoardDTO> create(
    @ApiParam(value = "게시글 객체", required = true)
    @RequestBody BoardDTO board) {
    return ResponseEntity.ok(service.create(board));
}
```



### controller.BoardController.java(전체 코드)

```
@ApiOperation(value = "게시글 수정", notes = "게시글 수정 API")
@ApiResponses(value = {
    @ApiResponse(code = 200, message = "성공적으로 요청이 처리되었습니다.", response = BoardDTO.class),
    @ApiResponse(code = 400, message = "잘못된 요청입니다."),
    @ApiResponse(code = 500, message = "서버에서 오류가 발생했습니다.")
})
@PutMapping("/{no}")
public ResponseEntity<BoardDTO> update(
    @ApiParam(value = "게시글 ID", required = true, example = "1")
    @PathVariable Long no,
    @ApiParam(value = "게시글 객체", required = true)
    @RequestBody BoardDTO board) {
    return ResponseEntity.ok(service.update(board));
}
```

### controller.BoardController.java(전체 코드)

```
@ApiOperation(value = "게시글 삭제", notes = "게시글 삭제 API")
@ApiResponses(value = {
    @ApiResponse(code = 200, message = "성공적으로 요청이 처리되었습니다."),
    @ApiResponse(code = 400, message = "잘못된 요청입니다."),
    @ApiResponse(code = 500, message = "서버에서 오류가 발생했습니다.")
})
@DeleteMapping("/{no}")
public ResponseEntity<BoardDTO> delete(
    @ApiParam(value = "게시글 ID", required = true, example = "1")
    @PathVariable
    Long no) {
    return ResponseEntity.ok(service.delete(no));
}
```

### ✓ DTO 모델에 대한 문서화

- `@ApiModelProperty(description = "게시글 DTO")`
  - DTO 클래스에 지정
- `@ApiModelProperty(value = "게시글 ID", example = "1")`
  - DTO 필드에 지정
  - 숫자 형인 경우 반드시 example에 숫자값 지정

## dto.BoardDTO.java

```
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
@ApiModel(description = "게시글 DTO")
public class BoardDTO {
    @ApiModelProperty(value = "업로드 파일 목록")
    List<MultipartFile> files = new ArrayList<>(); // 실제 업로드된 파일(Multipart) 목록
    @ApiModelProperty(value = "게시글 ID", example = "1")
    private Long no;
    @ApiModelProperty(value = "제목")
    private String title;
    @ApiModelProperty(value = "글 본문")
    private String content;
    @ApiModelProperty(value = "작성자")
    private String writer;
    @ApiModelProperty(value = "등록일")
    private Date regDate;
    @ApiModelProperty(value = "수정일")
    private Date updateDate;
    // 첨부 파일
    @ApiModelProperty(value = "첨부파일 목록")
    private List<BoardAttachmentVO> attaches;
```