KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

Rest Controller (심화1)

[KB] IT's Your Life

KB 국민은행

Rest Controller

- BoardController를 Rest Controller로 정의하세요.

Rest Controller

BoardController.java

```
package org.scoula.board.controller;

import org.scoula.board.service.BoardService;
import org.scoula.board.domain.BoardDTO;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/board")
@RequiredArgsConstructor
@Log4j2
public class BoardController {

    private final BoardService service;

}
```

Rest Controller

- 게시글 목록보기를 정의하고, Talend API Tester로 결과를 확인하세요.

Rest Controller

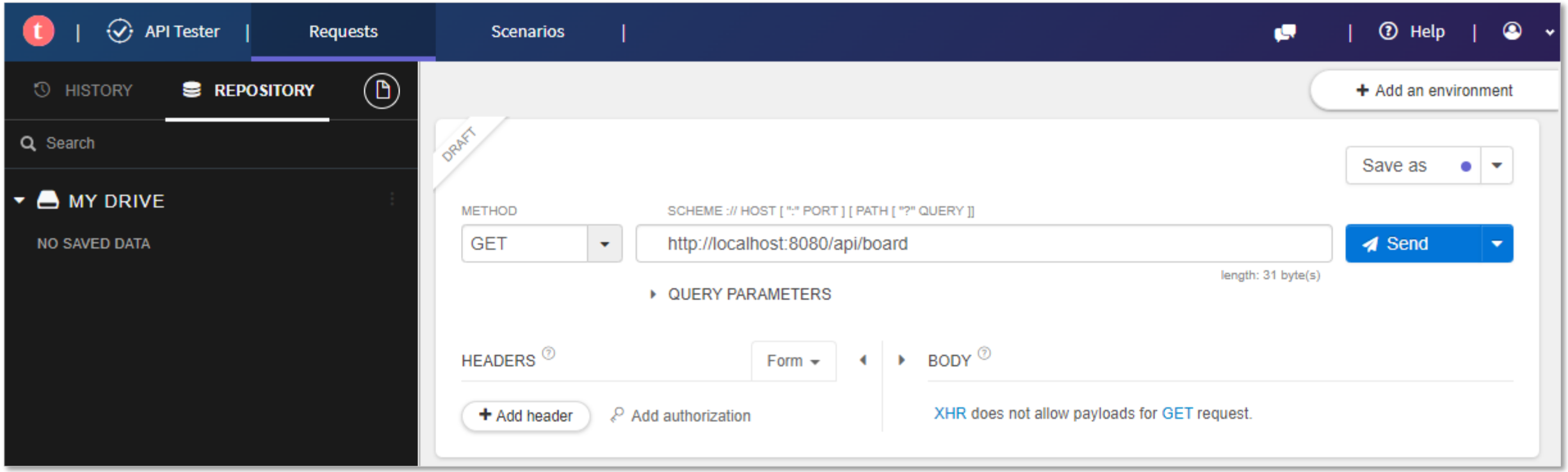
BoardController.java

```
@GetMapping("")
public List<BoardDTO> getList() {
    return service.getList();
}
```

- GET::http://localhost:8080/api/board

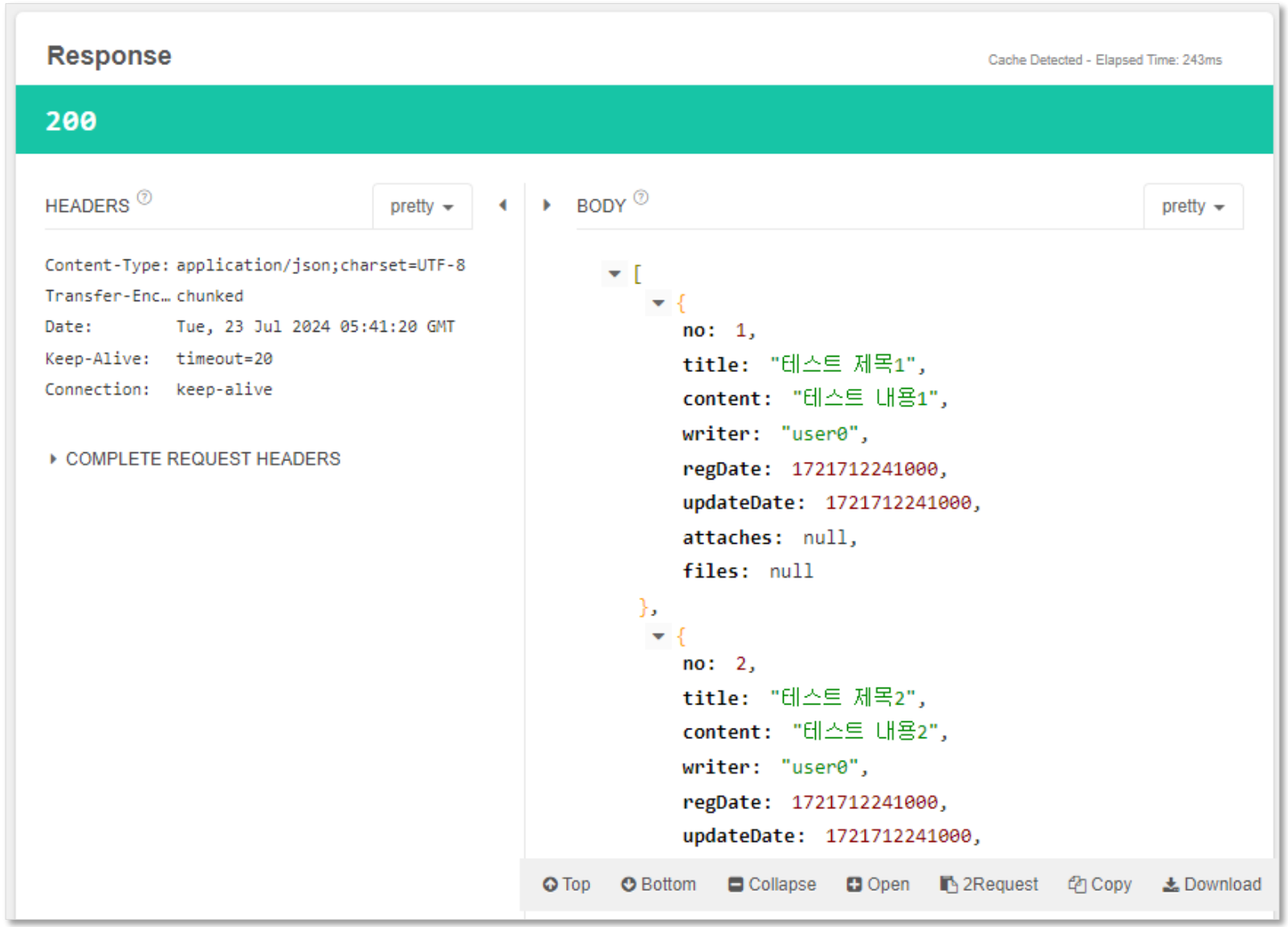
Rest Controller

- GET :: http://localhost:8080/api/board



Rest Controller

- GET :: http://localhost:8080/api/board



Rest Controller

- 게시글 목록보기를 ResponseEntity를 이용하여 수정하고, Talend API Tester로 결과를 확인하세요.

Rest Controller

BoardController.java

```
@GetMapping("")
public ResponseEntity<List<BoardDTO>> getList() {
    return ResponseEntity.ok(service.getList());
}
```

- GET::http://localhost:8080/api/board

Rest Controller

- 게시글 보기를 ResponseEntity를 이용하여 정의하고, Talend API Tester로 결과를 확인하세요.

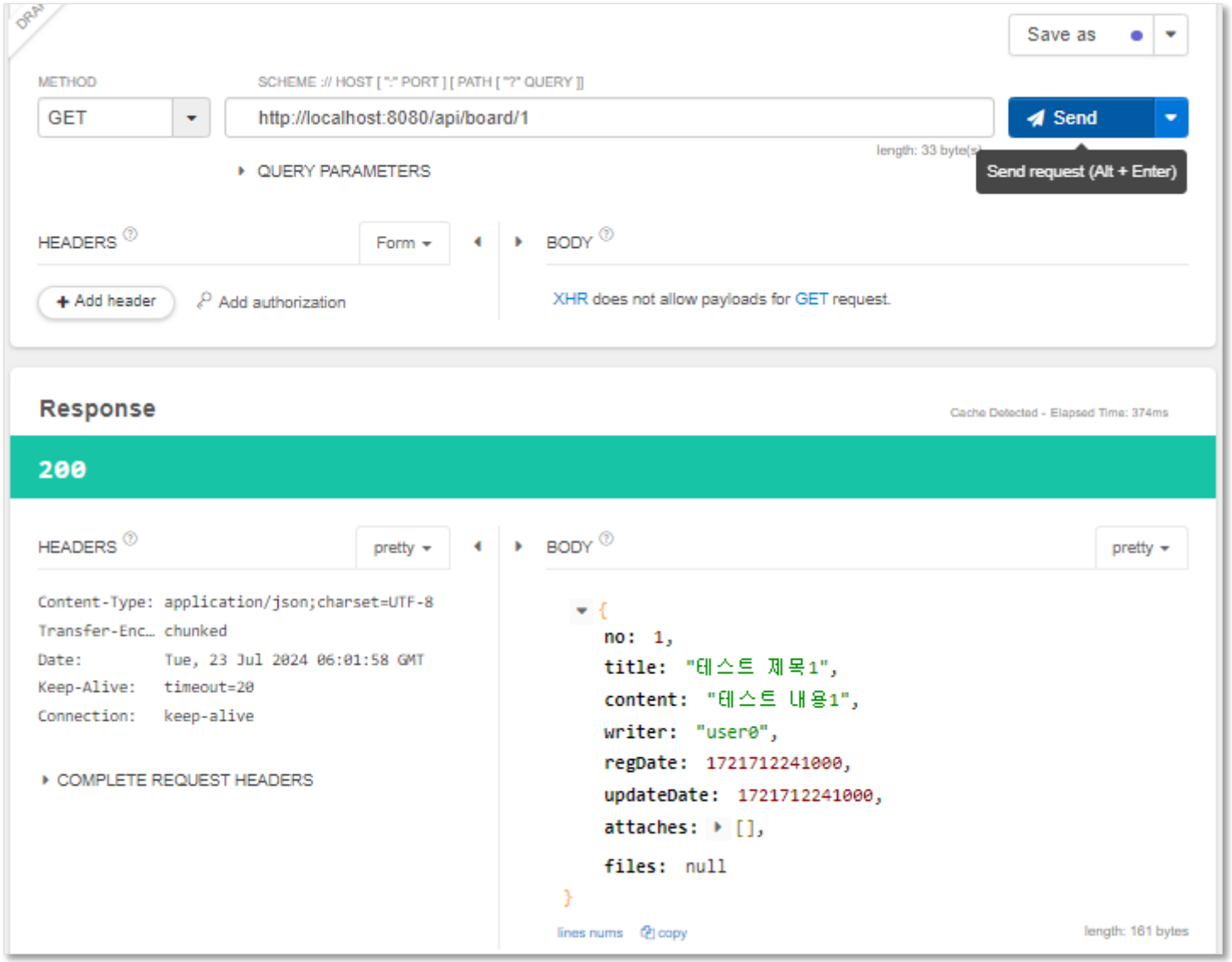
BoardController.java

```
@GetMapping("/{no}")
public ResponseEntity<BoardDTO> get(@PathVariable Long no) {
    return ResponseEntity.ok(service.get(no));
}
```

- GET::http://localhost:8080/api/board/1

Rest Controller

GET :: http://localhost:8080/api/board/1



Rest Controller

- 게시글 등록을 ResponseEntity를 이용하여 정의하고, Talend API Tester로 결과를 확인하세요.

Rest Controller

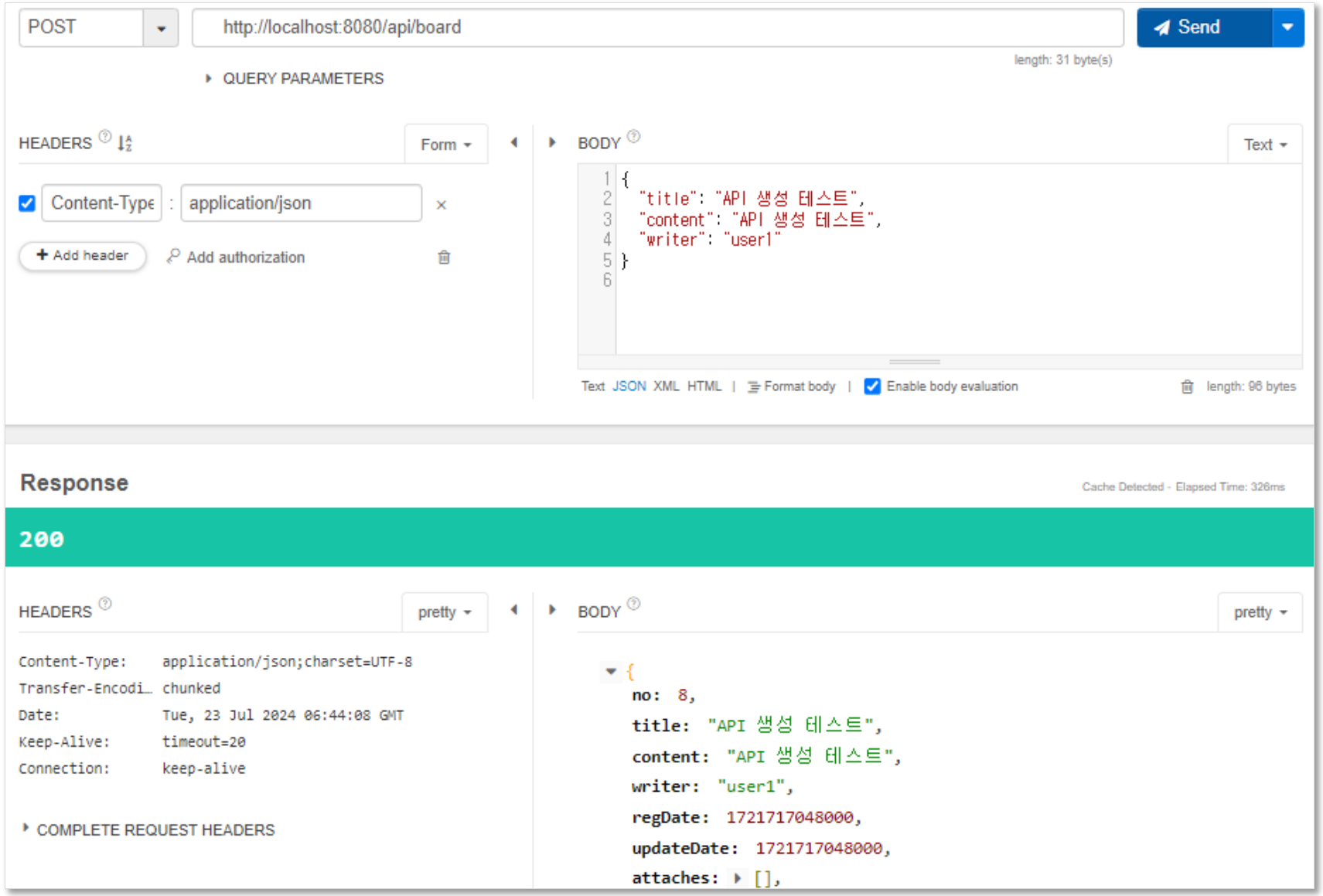
BoardController.java

```
@PostMapping("")
public ResponseEntity<Board> create(@RequestBody BoardDTO board) {
    return ResponseEntity.ok(service.create(board));
}
```

- POST::http://localhost:8080/api/board
  - BODY
    - Board 내용을 JSON 인코딩

Rest Controller

- POST :: http://localhost:8080/api/board



Rest Controller

- 게시글 수정하기를 ResponseEntity를 이용하여 정의하고, Talend API Tester로 결과를 확인하세요.

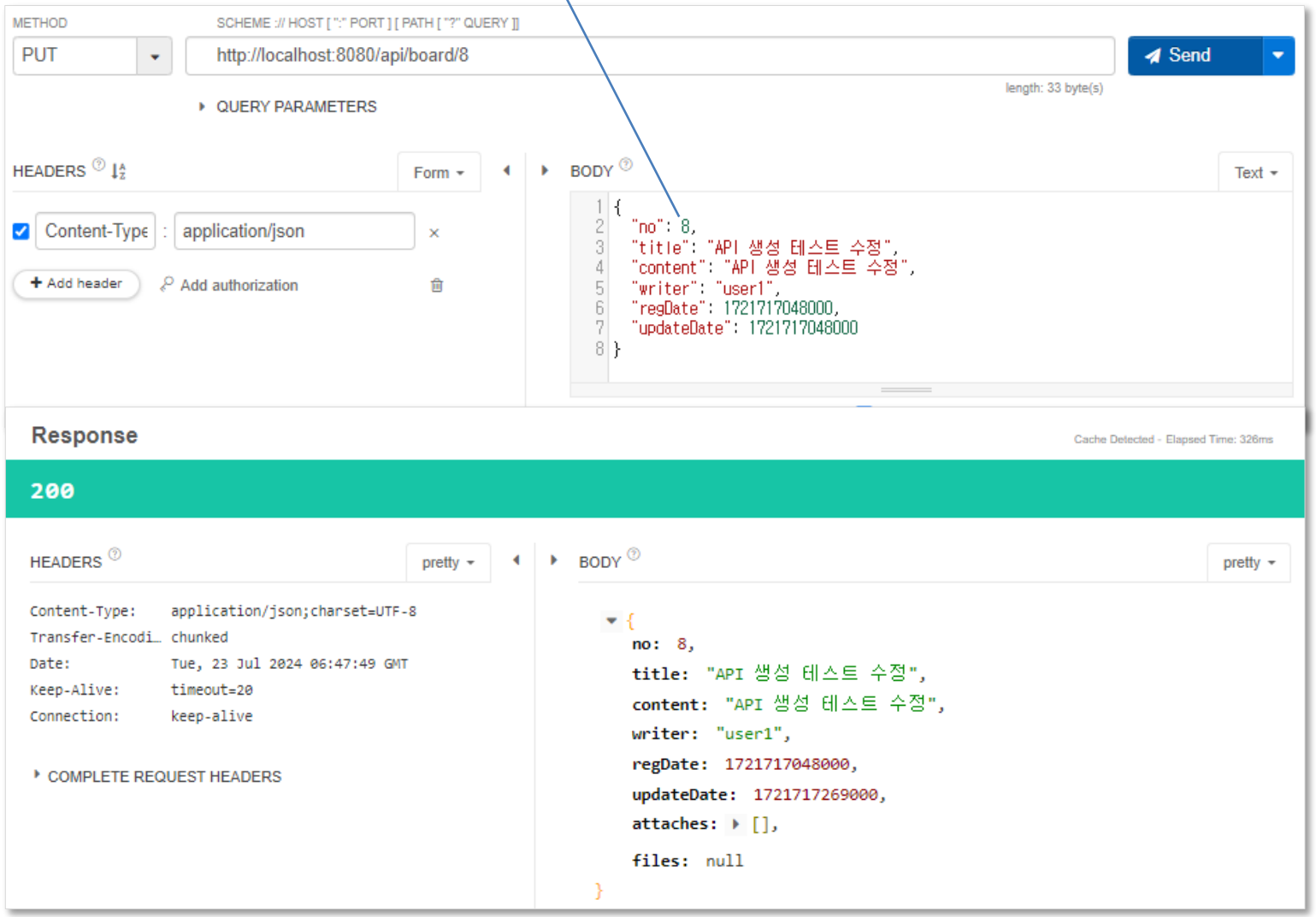
BoardController.java

```
@PutMapping("/{no}")
public ResponseEntity<Board> update(@PathVariable Long no @RequestBody BoardDTO board) {
    return ResponseEntity.ok(service.update(board));
}
```

- PUT::http://localhost:8080/api/board/11
  - BODY
    - Board 내용을 JSON 인코딩

Rest Controller

- PUT::http://localhost:8080/api/board/8



Rest Controller

- 게시글 삭제를 ResponseEntity를 이용하여 정의하고, Talend API Tester로 결과를 확인하세요.

Rest Controller

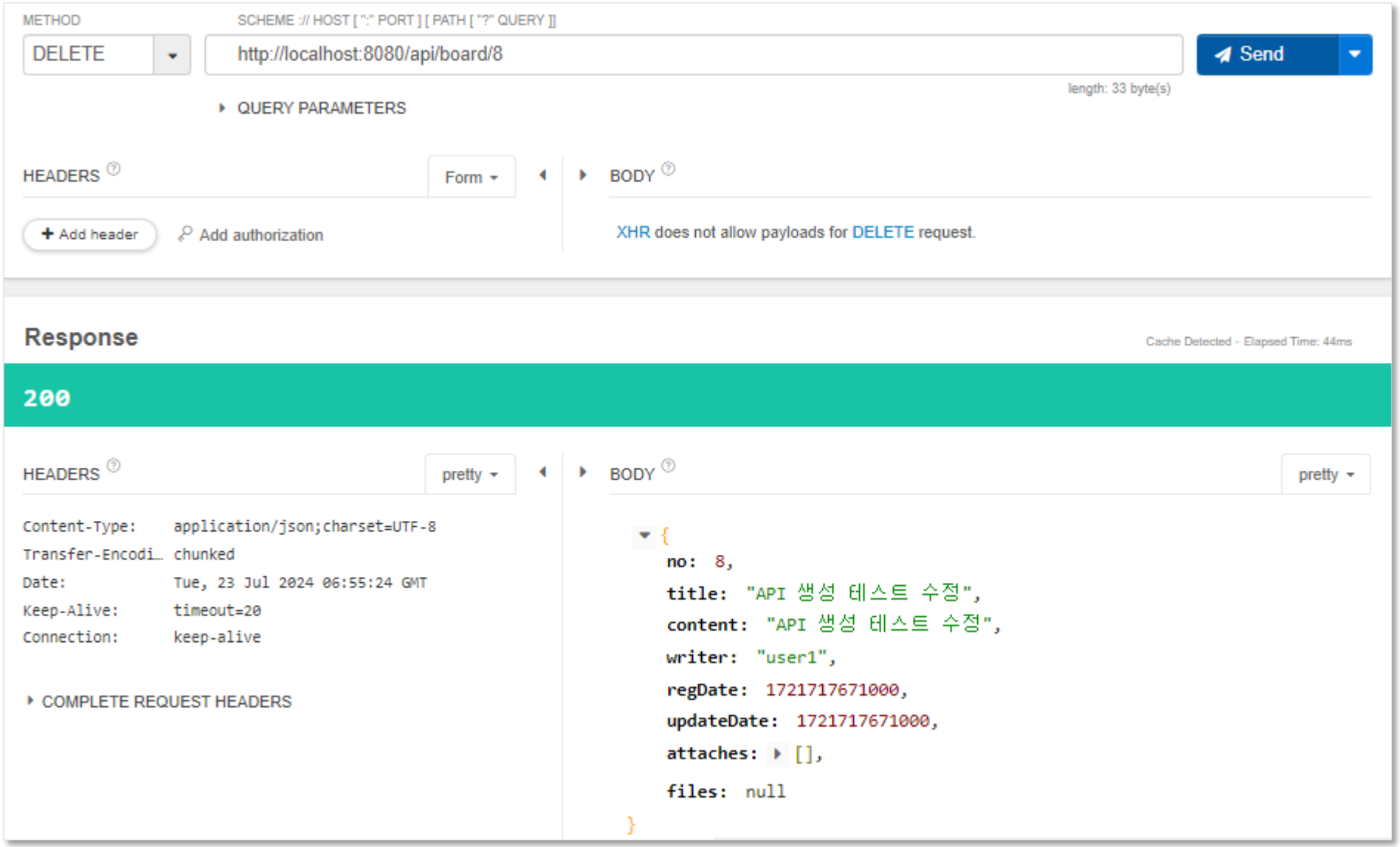
BoardController.java

```
@DeleteMapping("/{no}")
public ResponseEntity<BoardDTO> delete(@PathVariable Long no) {
    return ResponseEntity.ok(service.delete(no));
}
```

- DELETE :: http://localhost:8080/api/board/11

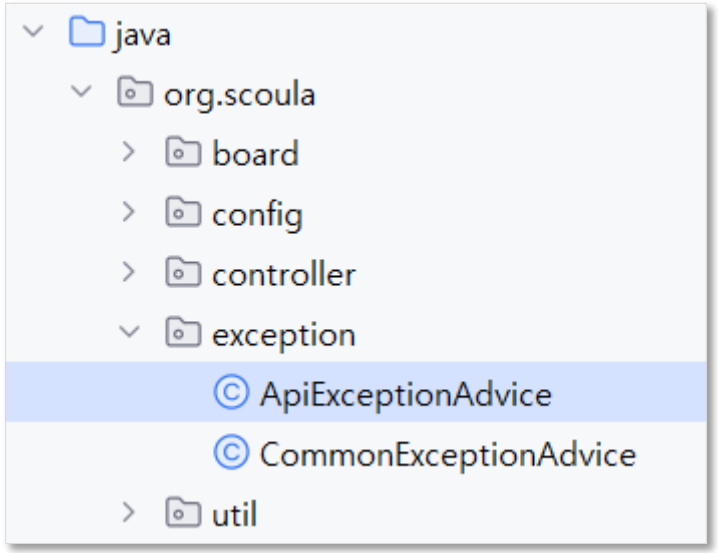
Rest Controller

- DELETE :: http://localhost:8080/api/board/8



Rest Controller

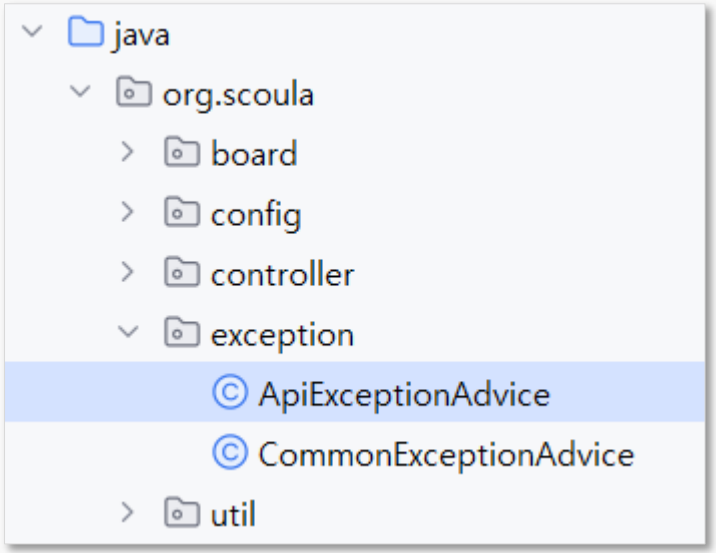
- **API 처리 때 예외가 발생한 경우 이를 일괄 처리하는 ApiExceptionAdvice 클래스를 정의하세요.**
  - 404에러 및 500 에러 처리



ApiExceptionAdvice.java

```
package org.scoula.exception;
...
@RestControllerAdvice
public class ApiExceptionAdvice {
    // 404 에러
    @ExceptionHandler(NoSuchElementException.class)
    protected ResponseEntity<String> handleIllegalArgumentException(NoSuchElementException e) {
        return ResponseEntity
                .status(HttpStatus.NOT_FOUND)
                .header("Content-Type", "text/plain;charset=UTF-8")
                .body("해당 ID의 요소가 없습니다.");
    }

    // 500 에러
    @ExceptionHandler(Exception.class)
    protected ResponseEntity<String> handleException(Exception e) {
        return ResponseEntity
                .status(HttpStatus.INTERNAL_SERVER_ERROR)
                .header("Content-Type", "text/plain;charset=UTF-8")
                .body(e.getMessage());
    }
}
```



Rest Controller

- GET::http://localhost:8080/api/board/100

KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

Swagger-ui를 활용한 문서 자동화 (심화 2)

[KB] IT's Your Life

KB 국민은행

Swagger-ui를 활용한 문서 자동화

- board api를 swagger-fox를 이용해서 문서화 하세요.

Swagger-ui를 활용한 문서 자동화

build.gradle

```
implementation 'io.springfox:springfox-swagger2:2.9.2'
implementation 'io.springfox:springfox-swagger-ui:2.9.2'
```

Swagger-ui를 활용한 문서 자동화

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());
    }
}
```

Swagger-ui를 활용한 문서 자동화

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/**"
        };
    }

    ...
}
```

Swagger-ui를 활용한 문서 자동화

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/");
    }
...
```

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 상제 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));
    }

}
```

Swagger-ui를 활용한 문서 자동화

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;
```