

2024년 상반기 K-디지털 트레이닝

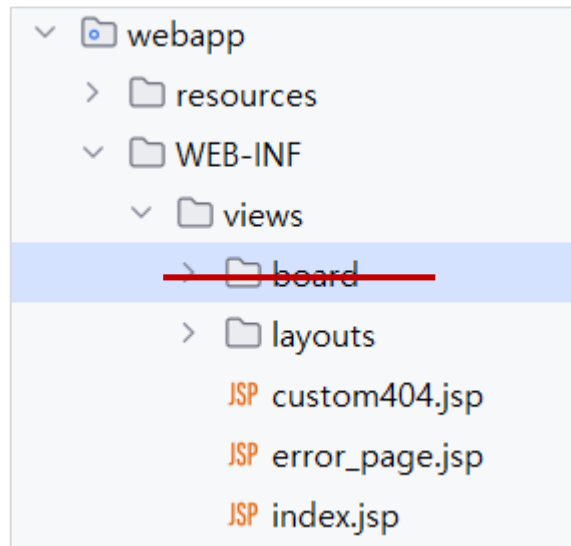
# Rest Controller

---

[KB] IT's Your Life

## ✓ 프로젝트 생성

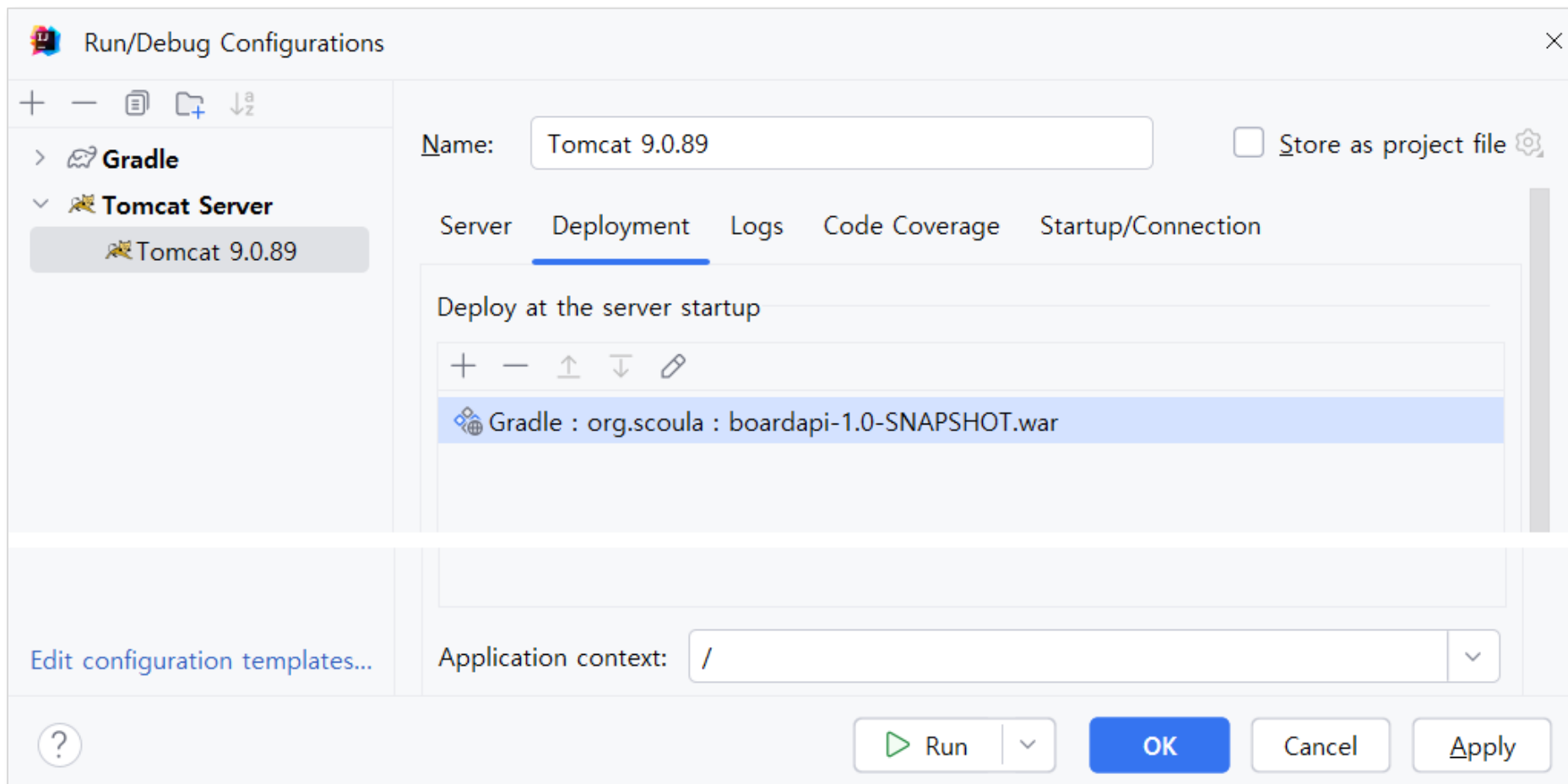
- board 프로젝트를 boardapi로 복사
- settings.gradle  
rootProject.name = "boardapi"
- org.scoula.board.controller.BoardController.java 삭제
- board 뷰 삭제



## ✓ 실행 설정

### ○ Deployment

- ArtifactId 추가
- Application context: /



## HomeController.java

```
package org.scoula.controller;

import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
@Slf4j
public class HomeController {

    @GetMapping("/")
    public String home() {
        log.info("=====> HomController /");
        return "index";    // View의 이름
    }

}
```

## ✓ API 서버에서 CUD(Create, Update, Delete)

### ○ 처리한 객체를 리턴

- 생성된 객체
- 업데이트된 객체
- 삭제된 객체

→ BoardService 수정

## BoardService.java

```
public interface BoardService {  
    public List<BoardDTO> getList();  
  
    public BoardDTO get(Long no);  
  
    public BoardDTO create(BoardDTO board);  
  
    public BoardDTO update(BoardDTO board);  
  
    public BoardDTO delete(Long no);  
  
    public BoardAttachmentVO getAttachment(Long no);  
  
    public boolean deleteAttachment(Long no);  
}
```

## BoardServiceImpl.java

```
public class BoardServiceImpl implements BoardService {  
    ...  
    @Transactional // 2개 이상의 insert 문이 실행될 수 있으므로 트랜잭션 처리 필요  
    @Override  
    public BoardDTO create(BoardDTO board) {  
        log.info("create....." + board);  
  
        BoardVO boardVO= board.toVo();  
        mapper.create(boardVO);  
  
        // 파일 업로드 처리  
        List<MultipartFile> files = board.GetFiles();  
        if(files != null && !files.isEmpty()) {  
            upload(boardVO.getNo(), files);  
        }  
  
        return get(boardVO.getNo());  
    }  
}
```

## BoardServiceImpl.java

```
...

@Override
public BoardDTO update(BoardDTO board) {
    log.info("update....." + board);
    mapper.update(board.toVo());

    return get(board.getNo());
}

@Override
public BoardDTO delete(Long no) {
    log.info("delete...." + no);
    BoardDTO board = get(no);

    mapper.delete(no);
    return board;
}
...
```



# 1 Rest Controller

## ✓ Rest Api용 컨트롤러

### ○ @RestController

- 모든 메서드에 @ResponseBody를 자동으로 추가
  - 응답 헤더에 content-type을 application/json 타입으로 설정
  - 메서드가 객체를 리턴하면 자동으로 Json 문자열로 변환
  - jackson 라이브러리가 담당
- 요청의 body가 application/json 인코딩인 경우, 매개변수 앞에 @RequestBody 사용

### ○ 매핑 어노테이션

- @GetMapping(url)
- @PostMapping(url)
- @PutMapping(url)
- @DeleteMapping(url)

## 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
@Slf4j
public class BoardController {

    private final BoardService service;

}
```

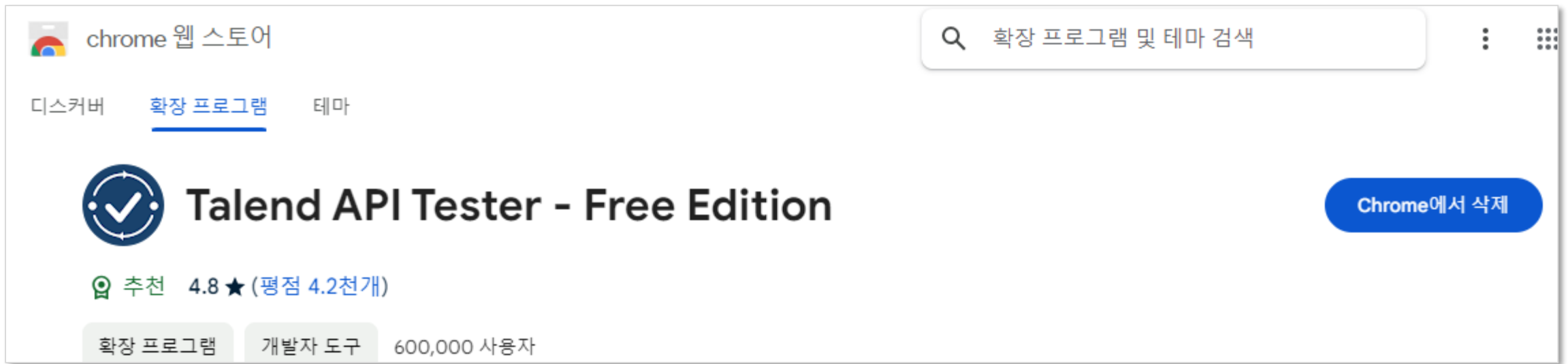
## BoardController.java

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

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

## ☑ Talend API Tester

- Rest API 호출 테스트를 할 수 있는 크롬 확장 프로그램
- 구글: "talend api 확장" 검색



✓ GET :: `http://localhost:8080/api/board`

The screenshot displays the API Tester application interface. The top navigation bar includes a home icon, a checkmark icon, and the text "API Tester". Below this, the "Requests" tab is selected, with "Scenarios" also visible. On the left sidebar, there are sections for "HISTORY", "REPOSITORY" (with a search bar), and "MY DRIVE" (showing "NO SAVED DATA"). The main workspace is titled "DRAFT" and contains a request configuration form. The form has a "METHOD" dropdown set to "GET" and a URL input field containing "http://localhost:8080/api/board". A "Send" button is located to the right of the URL field. Below the URL field, there is a "QUERY PARAMETERS" section. At the bottom, there are tabs for "HEADERS" and "BODY". The "HEADERS" tab is active, showing a "Form" dropdown and buttons for "Add header" and "Add authorization". The "BODY" tab is also visible, with a message stating "XHR does not allow payloads for GET request." A "Save as" button is located in the top right corner of the workspace.

API Tester | Requests | Scenarios | Help | User

+ Add an environment

DRAFT

METHOD: GET | SCHEME :// HOST [ ":" PORT ] [ PATH [ "?" QUERY ] ] | http://localhost:8080/api/board | length: 31 byte(s)

QUERY PARAMETERS

HEADERS ? | Form | BODY ?

+ Add header | Add authorization

XHR does not allow payloads for GET request.

Save as

Send

## ✓ GET :: http://localhost:8080/api/board

### Response

Cache Detected - Elapsed Time: 243ms

## 200

#### HEADERS

Content-Type: application/json;charset=UTF-8  
Transfer-Enc... chunked  
Date: Tue, 23 Jul 2024 05:41:20 GMT  
Keep-Alive: timeout=20  
Connection: keep-alive

▶ COMPLETE REQUEST HEADERS

#### BODY

```
[
  {
    no: 1,
    title: "테스트 제목1",
    content: "테스트 내용1",
    writer: "user0",
    regDate: 1721712241000,
    updateDate: 1721712241000,
    attaches: null,
    files: null
  },
  {
    no: 2,
    title: "테스트 제목2",
    content: "테스트 내용2",
    writer: "user0",
    regDate: 1721712241000,
    updateDate: 1721712241000,
  }
]
```

Top Bottom Collapse Open 2Request Copy Download

## ✓ ResponseEntity<T>

- Rest Controller가 객체를 리턴하는 경우 상태 코드/헤더 설정하기 힘들
- ResponseEntity<T>를 사용하면 상태 코드, 응답 헤더, body를 설정할 수 있음.

```
public class HttpEntity<T> {  
  
    private final HttpHeaders headers;  
  
    @Nullable  
    private final T body;  
}
```

```
public class RequestEntity<T> extends HttpEntity<T>
```

```
public class ResponseEntity<T> extends HttpEntity<T>
```

## ✓ ResponseEntity<T> 생성자

```
public ResponseEntity(HttpStatus status) {  
    this(null, null, status);  
}
```

```
public ResponseEntity(@Nullable T body, HttpStatus status) {  
    this(body, null, status);  
}
```

```
public ResponseEntity(@Nullable T body, @Nullable MultiValueMap<String, String> headers, HttpStatus status) {  
    super(body, headers);  
    Assert.notNull(status, "HttpStatus must not be null");  
    this.status = status;  
}
```



## ✓ ResponseEntity<T> 생성

### ○ 생성자 패턴(비권장)

```
return new ResponseEntity(body, headers, HttpStatus.valueOf(200));
```

### ○ 빌더 패턴 (권장)

```
return ResponseEntity.ok().build(); // 200 코드만 구성
```

```
return ResponseEntity.ok(body); // 200 코드 + body 구성
```

```
return ResponseEntity.ok() // 200 코드 + 헤더 + body 구성
    .headers(headers)
    .body(body);
```

```
return ResponseEntity.status(상태코드).build(); // 상태 코드만 구성
```

```
return ResponseEntity.status(상태코드).body(body); // 상태 코드 + body 구성
```

```
return ResponseEntity.status(상태코드).headers(headers).body(body); // 상태 코드 + header + body 구성
```

## BoardController.java

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

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

## BoardController.java

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

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



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

The screenshot displays a REST client interface with the following details:

- Request Section:**
  - METHOD:** GET
  - URL:** http://localhost:8080/api/board/1 (length: 33 bytes)
  - Buttons:** Save as, Send, Send request (Alt + Enter)
  - QUERY PARAMETERS:** (empty)
  - HEADERS:** Form view, + Add header, Add authorization
  - BODY:** XHR does not allow payloads for GET request.
- Response Section:**
  - Status:** 200 (Cache Detected - Elapsed Time: 374ms)
  - HEADERS:** pretty view, Content-Type: application/json; charset=UTF-8, Transfer-Encoding: chunked, Date: Tue, 23 Jul 2024 06:01:58 GMT, Keep-Alive: timeout=20, Connection: keep-alive, COMPLETE REQUEST HEADERS
  - BODY:** pretty view, JSON response:

```
{
  no: 1,
  title: "테스트 제목1",
  content: "테스트 내용1",
  writer: "user0",
  regDate: 1721712241000,
  updateDate: 1721712241000,
  attaches: [],
  files: null
}
```

 (length: 161 bytes)

## BoardController.java

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

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

## ✓ POST :: http://localhost:8080/api/board

The screenshot displays a REST client interface with a POST request to `http://localhost:8080/api/board`. The request body is a JSON object with the following content:

```
{
  "title": "API 생성 테스트",
  "content": "API 생성 테스트",
  "writer": "user1"
}
```

The response is a 200 status code with the following headers:

- Content-Type: application/json; charset=UTF-8
- Transfer-Encoding: chunked
- Date: Tue, 23 Jul 2024 06:44:08 GMT
- Keep-Alive: timeout=20
- Connection: keep-alive

The response body is a JSON object with the following content:

```
{
  no: 8,
  title: "API 생성 테스트",
  content: "API 생성 테스트",
  writer: "user1",
  regDate: 1721717048000,
  updateDate: 1721717048000,
  attaches: []
}
```

## BoardController.java

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

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

## ✓ PUT::http://localhost:8080/api/board/8

The screenshot displays a REST client interface with a PUT request configured for the URL `http://localhost:8080/api/board/8`. The request body is a JSON object. The response is a 200 status code with a JSON body.

**Request Details:**

- METHOD:** PUT
- URL:** `http://localhost:8080/api/board/8`
- Content-Type:** `application/json`
- Body:**

```
1 {  
2   "no": 8,  
3   "title": "API 생성 테스트 수정",  
4   "content": "API 생성 테스트 수정",  
5   "writer": "user1",  
6   "regDate": 1721717048000,  
7   "updateDate": 1721717048000  
8 }
```

**Response Details:**

- Status:** 200
- Content-Type:** `application/json; charset=UTF-8`
- Body:**

```
{  
  no: 8,  
  title: "API 생성 테스트 수정",  
  content: "API 생성 테스트 수정",  
  writer: "user1",  
  regDate: 1721717048000,  
  updateDate: 1721717269000,  
  attaches: [],  
  files: null  
}
```



## BoardController.java

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

○ DELETE :: <http://localhost:8080/api/board/11>

## ✓ DELETE :: http://localhost:8080/api/board/8

The screenshot displays a REST client interface with a DELETE request to `http://localhost:8080/api/board/8`. The request is successfully executed, returning a 200 status code. The response body is a JSON object containing details about the deleted board item.

**Request Details:**

- METHOD:** DELETE
- URL:** http://localhost:8080/api/board/8
- Length:** 33 byte(s)
- QUERY PARAMETERS:** (None)
- HEADERS:** (None)
- BODY:** XHR does not allow payloads for DELETE request.

**Response Details:**

- Status:** 200
- Cache:** Cache Detected - Elapsed Time: 44ms
- HEADERS:**
  - Content-Type: application/json;charset=UTF-8
  - Transfer-Encoding: chunked
  - Date: Tue, 23 Jul 2024 06:55:24 GMT
  - Keep-Alive: timeout=20
  - Connection: keep-alive
- BODY:**

```
{
  "no": 8,
  "title": "API 생성 테스트 수정",
  "content": "API 생성 테스트 수정",
  "writer": "user1",
  "regDate": 1721717671000,
  "updateDate": 1721717671000,
  "attaches": [],
  "files": null
}
```

## ✓ REST API 예외 처리

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

### Response

Cache Detected - Elapsed Time: 324ms

## 200

#### HEADERS ⓘ

pretty ▾

Content-Type: text/html; charset=UTF-8

Content-Language: ko-KR

Content-Length: 5 kilobytes

Date: Wed, 03 Jul 2024 05:53:59 GMT

Keep-Alive: timeout=20

Connection: keep-alive

▶ COMPLETE REQUEST HEADERS

#### BODY ⓘ

pretty ▾

```
<!DOCTYPE html>

<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Insert title here</title>
</head>
<body>
<h4>ID: 100 not found</h4>

<ul>
```

- CommonExceptionHandler가 에러를 처리함
  - jsp로 출력함

## ✓ @RestControllerAdvice

- AOP 기능을 이용하여 @RestController 처리과정에서 발생한 예외를 핸들링

- 기본 구조

**@RestControllerAdvice**

```
public class ApiExceptionAdvice {
```

```
    @ExceptionHandler({처리할 예외 클래스.class, ... })
```

```
    public ResponseEntity<T> 핸들러(HttpServletRequest request, 예외클래스 e) {  
        // ResponseEntity 구성 후 리턴
```

```
    }
```

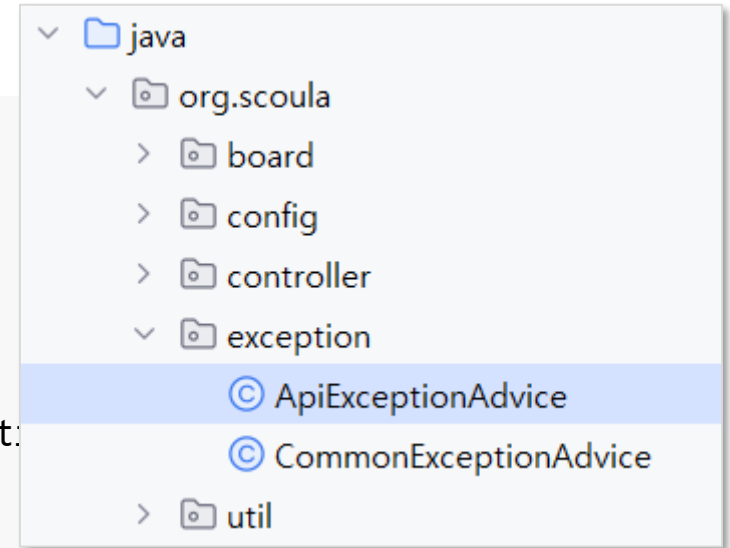
```
    ...
```

```
}
```

## 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());
    }
}
```



## ✓ GET::http://localhost:8080/api/board/100

METHOD

GET

SCHEME :// HOST [ ":" PORT ][ PATH [ "?" QUERY ]]

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

length: 35 byte(s)

Send

QUERY PARAMETERS

HEADERS

Form

+ Add header

Add authorization

BODY

XHR does not allow payloads for GET request.

Response

Cache Detected - Elapsed Time: 140ms

404

HEADERS

pretty

Content-Type: text/plain;charset=UTF-8

Content-Length: 36 bytes

Date: Tue, 23 Jul 2024 07:11:52 GMT

Keep-Alive: timeout=20

Connection: keep-alive

COMPLETE REQUEST HEADERS

BODY

raw

해당 ID의 요소가 없습니다.

copy

length: 36 bytes