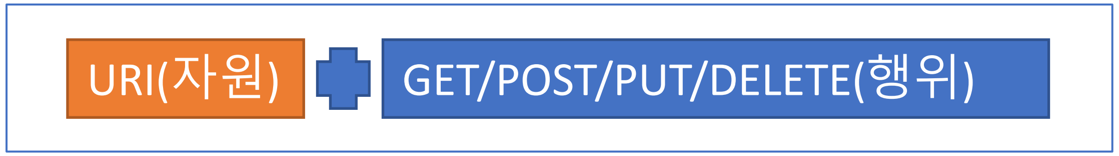
PART6 – Ajax와 JSON

REST 방식의 서비스

- 과거의 웹은 경로 자체가 특정한 행위를 의미하고 추가적인 데이터는 파라미터를 이용해서 처리하는 방식으로 설계
- **URI/URL**의 고유한 의미에 대한 재인식
  - **URL**자체가 하나의 고유한 목적지이면서 자원에 대한 의미
  - 행위에 대한 부분은 **GET/POST**등을 활용해서 처리



Ajax와 JSON

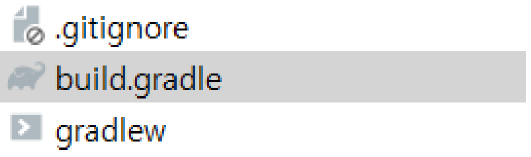
- Asynchronous JavaScript And XML – 비동기로 서버와의 통신 처리
- JavaScript에서 많은 역할을 수행하고 서버에서는 XML이나 JSON과 같이 순수한 데이터만 처리하는 방식
- 개발의 무게 중심 변화
- JSON(JavaScript Object Notation)

REST방식의 구현

- Ajax를 이용해서 클라이언트와 서버 사이의 통신이 가능하다.
- Ajax는 처음에는 브라우저에서 서버를 호출하는데 사용되었지만 모바일에서도 유용하다.
- Ajax로 데이터를 주고 받을 때는 문자열이 가장 적합하다.
- 문자열로 구조화된 데이터를 표현하기 위해서 JSON이라는 포맷이 사용된다.
- JSON은 ‘키:값’과 ‘{ }’를 이용해서 객체의 구조를 표현할 수 있다.

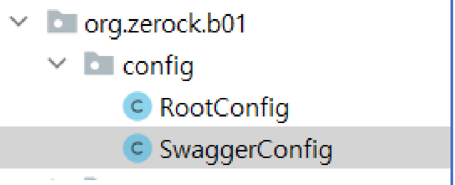
Swagger UI

- REST api document tool
- 약간의 설정으로 api 테스트 환경과 문서 작업을 처리할 수 있음



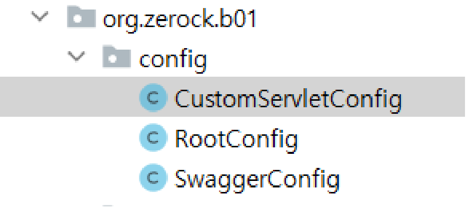
.gitignore
build.gradle
gradlew

```
dependencies {
    ...

    implementation 'io.springfox:springfox-boot-starter:3.0.0'
    implementation 'io.springfox:springfox-swagger-ui:3.0.0'

}
```



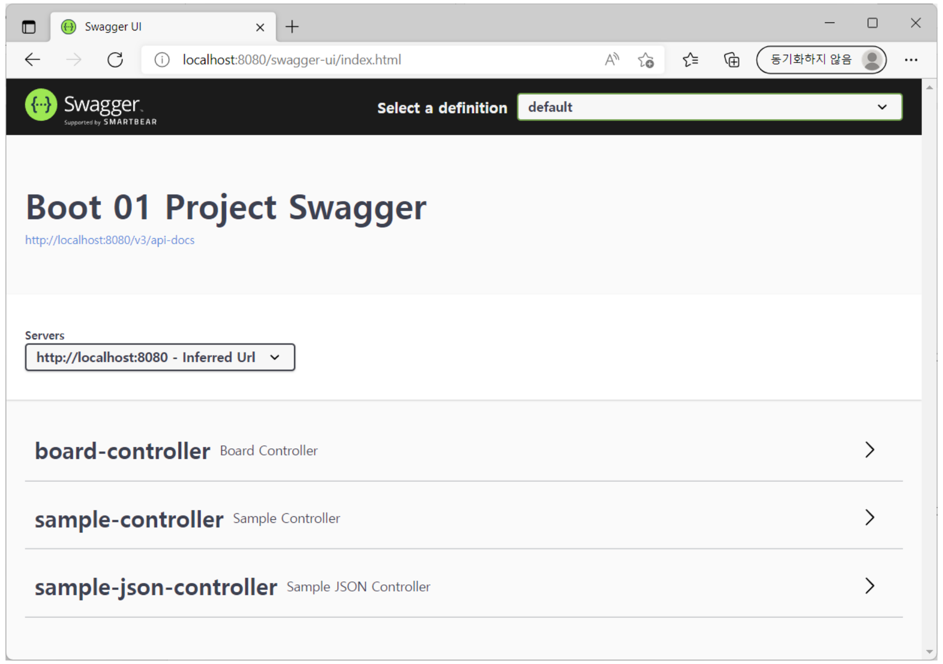
org.zerock.b01
config
RootConfig
SwaggerConfig

```
package org.zerock.b01.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;

@Configuration

public class SwaggerConfig {

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.OAS_30)
                .useDefaultResponseMessages(false)
                .select()

.apis(RequestHandlerSelectors.basePackage("org.zerock.b01.controller"))
                .paths(PathSelectors.any())
                .build()
                .apiInfo(apiInfo());
```



```
org.zerock.b01
  config
    CustomServletConfig
    RootConfig
    SwaggerConfig
```

```java
package org.zerock.b01.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;

@Configuration
@EnableWebMvc
public class CustomServletConfig {

}
```



Swagger UI

localhost:8080/swagger-ui/index.html

동기화하지 않음

Swagger

Select a definition default

Boot 01 Project Swagger

http://localhost:8080/v3/api-docs

Servers

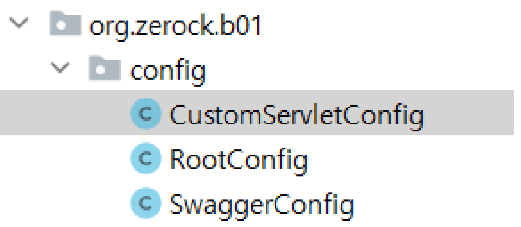
http://localhost:8080 - Inferred Url

board-controller Board Controller

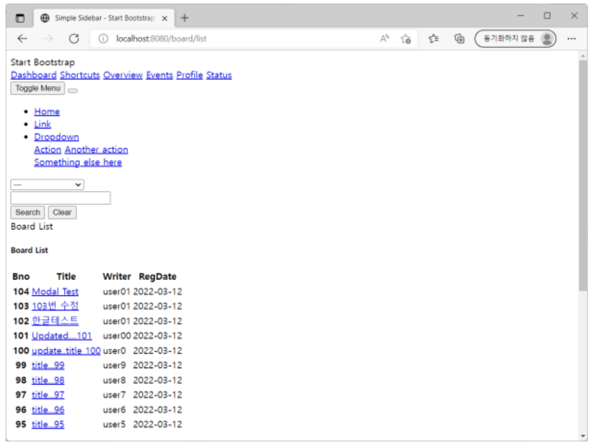
sample-controller Sample Controller

sample-json-controller Sample JSON Controller

정적 파일을 위한 Swagger UI 설정



```
org.zerock.b01
  config
    CustomServletConfig
    RootConfig
    SwaggerConfig
```



Start Bootstrap
Dashboard Shortcuts Overview Events Profile Status
Toggle Menu

- Home
- Link
- Dropdown
  Action Another action
  Something else here

Search Clear
Board List

Board List

Bno	Title	Writer	RegDate
104	Modal Test	user01	2022-03-12
103	103번 수정	user01	2022-03-12
102	한글테스트	user01	2022-03-12
101	Updated....101	user00	2022-03-12
100	update..title 100	user0	2022-03-12
99	title...99	user9	2022-03-12
98	title...98	user8	2022-03-12
97	title...97	user7	2022-03-12
96	title...96	user6	2022-03-12
95	title...95	user5	2022-03-12

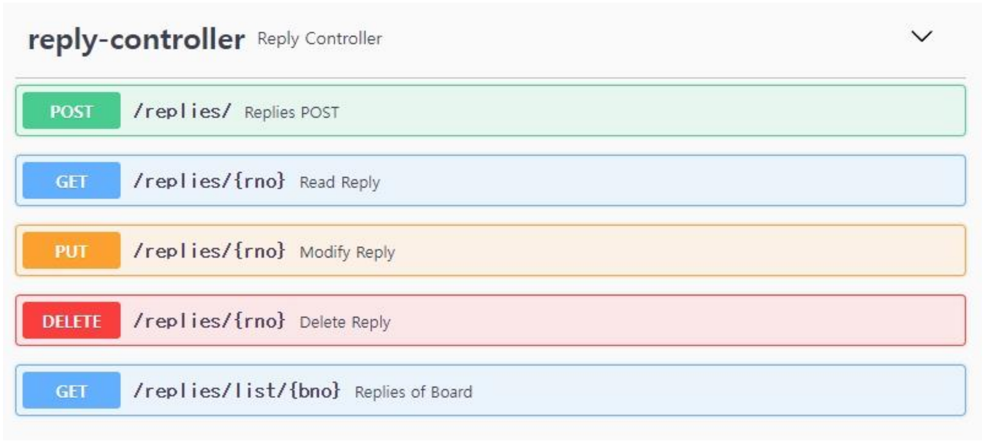
```java
package org.zerock.b01.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
@EnableWebMvc
public class CustomServletConfig implements WebMvcConfigurer {

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {

        registry.addResourceHandler("/js/**")
                .addResourceLocations("classpath:/static/js/");
        registry.addResourceHandler("/fonts/**")
                .addResourceLocations("classpath:/static/fonts/");
        registry.addResourceHandler("/css/**")
                .addResourceLocations("classpath:/static/css/");
        registry.addResourceHandler("/assets/**").
                addResourceLocations("classpath:/static/assets/");

    }
}
```

REST방식의 댓글 처리 준비

- 구현 단계
  - URL의 설계와 데이터 포맷 결정
  - 컨트롤러의 JSON/XML 처리
  - 동작 확인
  - JavaScript를 통한 화면 처리

URL설계와 DTO 설계

URL(Method)	설명	반환 데이터
/replies (POST)	특정한 게시물의 댓글 추가	{'rno':11 } – 생성된 댓글의 번호
/replies/list/:bno (GET)	특정 게시물(bno)의 댓글의 목록 '?' 뒤에 페이지 번호 등을 추가해서 댓글 페이징 처리	PageResponseDTO를 JSON으로 처리
/replies/:rno (PUT)	특정한 번호의 댓글 수정	{'rno':11} -수정된 댓글의 번호
/replies/:rno (DELETE)	특정한 번호의 댓글 삭제	{'rno':11} – 삭제된 댓글의 번호
/replies/:rno (GET)	특정한 번호의 댓글 조회	댓글 객체를 JSON으로 변환한 문자열


reply-controller Reply Controller

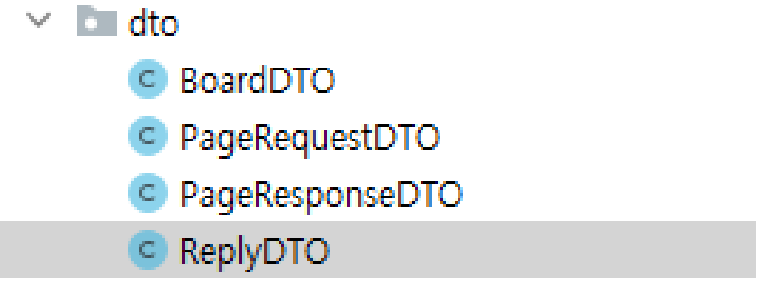
POST /replies/ Replies POST

GET /replies/{rno} Read Reply

PUT /replies/{rno} Modify Reply

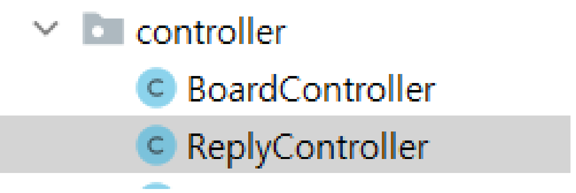
DELETE /replies/{rno} Delete Reply

GET /replies/list/{bno} Replies of Board



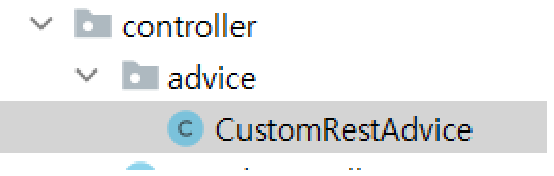
```java
package org.zerock.b01.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDateTime;



@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class ReplyDTO {

    private Long rno;

    private Long bno;

    private String replyText;

    private String replyer;

    private LocalDateTime regDate, modDate;

}
```

ReplyController의 준비



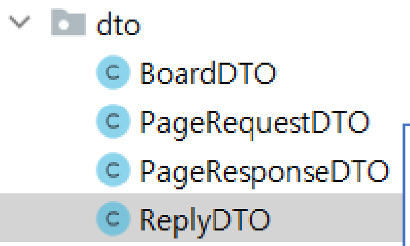
```java
package org.zerock.b01.controller;

import io.swagger.annotations.ApiOperation;
import lombok.extern.log4j.Log4j2;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.zerock.b01.dto.ReplyDTO;

import java.util.HashMap;
import java.util.Map;

@RestController
@RequestMapping("/replies")
@Log4j2
public class ReplyController {

    @ApiOperation(value = "Replies POST", notes = "POST 방식으로 댓글 등록")
    @PostMapping(value = "/", consumes = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<Map<String,Long>> register(@RequestBody ReplyDTO replyDTO){

        log.info(replyDTO);

        Map<String, Long> resultMap = Map.of("rno", 111L);

        return ResponseEntity.ok(resultMap);
    }

}
```

@Valid와 @RestControllerAdvice


controller
advice
CustomRestAdvice

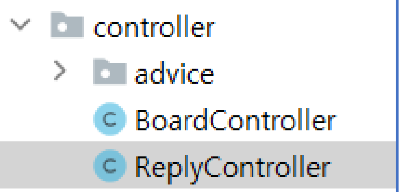
```
package org.zerock.b01.controller.advice;


import lombok.extern.log4j.Log4j2;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindException;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestControllerAdvice;

import java.util.HashMap;
import java.util.Map;

@RestControllerAdvice
@Log4j2
public class CustomRestAdvice {

    @ExceptionHandler(BindException.class)
    @ResponseStatus(HttpStatus.EXPECTATION_FAILED)
    public ResponseEntity<Map<String, String>> handleBindException(BindException e) {

        log.error(e);

        Map<String, String> errorMap = new HashMap<>();

        if(e.hasErrors()){

            BindingResult bindingResult = e.getBindingResult();

            bindingResult.getFieldErrors().forEach(fieldError -> {
                errorMap.put(fieldError.getField(), fieldError.getCode());
            });
        }

        return ResponseEntity.badRequest().body(errorMap);
    }
}
```

댓글 등록의 @Valid



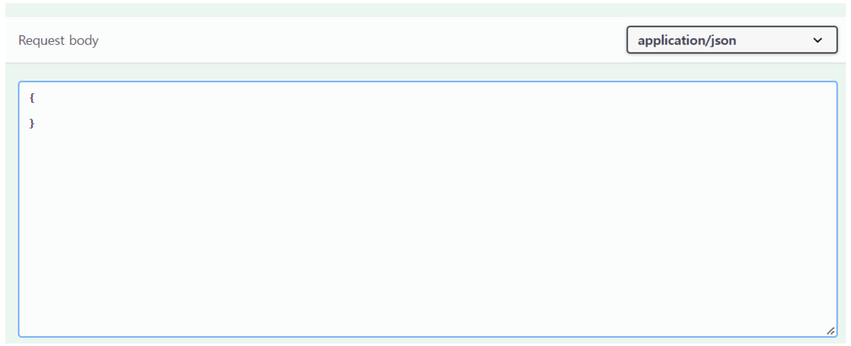
dto
- BoardDTO
- PageRequestDTO
- PageResponseDTO
- ReplyDTO

```java
package org.zerock.b01.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
import java.time.LocalDateTime;



@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class ReplyDTO {

    private Long rno;

    @NotNull
    private Long bno;

    @NotEmpty
    private String replyText;

    @NotEmpty
    private String replyer;

    private LocalDateTime regDate, modDate;

}
```



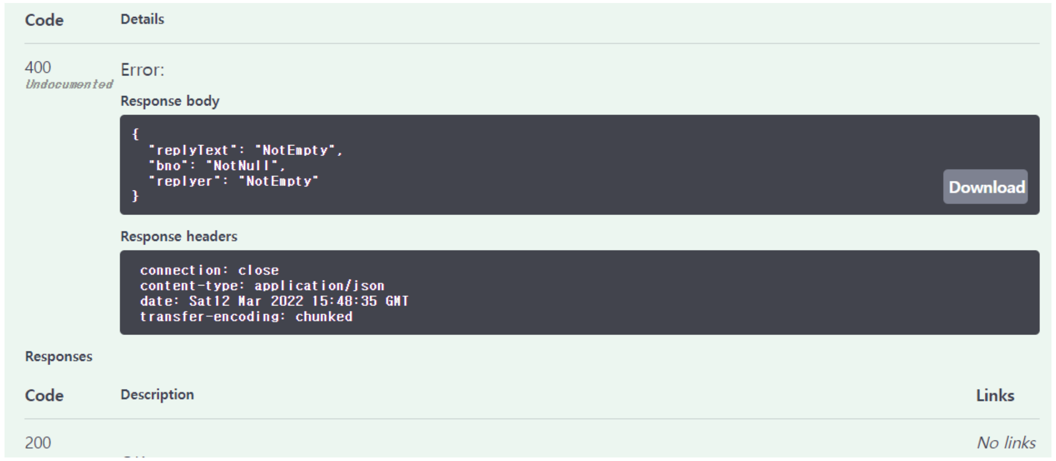
controller
- advice
- BoardController
- ReplyController

```java
package org.zerock.b01.controller;

import io.swagger.annotations.ApiOperation;
import lombok.extern.log4j.Log4j2;
import org.springframework.http.MediaType;
import org.springframework.validation.BindException;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.zerock.b01.dto.ReplyDTO;

import javax.validation.Valid;
import java.util.HashMap;
import java.util.Map;

@RestController
@RequestMapping("/replies")
@Log4j2
public class ReplyController {

    @ApiOperation(value = "Replies POST", notes = "POST 방식으로 댓글 등록")
    @PostMapping(value = "/", consumes = MediaType.APPLICATION_JSON_VALUE)
    public Map<String,Long> register(
            @Valid @RequestBody ReplyDTO replyDTO,
            BindingResult bindingResult)throws BindException{

        log.info(replyDTO);

        if(bindingResult.hasErrors()){
            throw new BindException(bindingResult);
        }

        Map<String, Long> resultMap = new HashMap<>();
        resultMap.put("rno",111L);

        return resultMap;
    }

}
```


Request body

application/json

```
{
}
```


Code | Details

400
Undocumented

Error:

Response body

```
{
  "replyText": "NotEmpty",
  "bno": "NotNull",
  "replyer": "NotEmpty"
}
```

Download

Response headers

```
connection: close
content-type: application/json
date: Sat12 Mar 2022 15:48:35 GMT
transfer-encoding: chunked
```

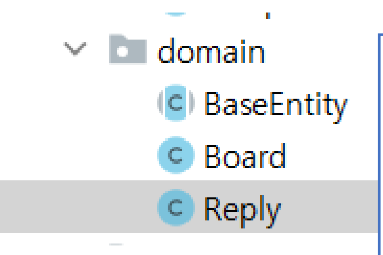
Responses

Code	Description	Links
200		*No links*

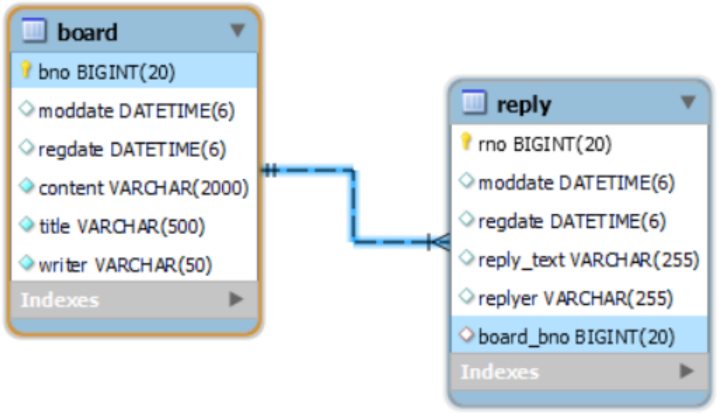
다대일(ManyToOne) 연관관계

- 연관관계의 기준 결정
  - 연관관계의 기준은 항상 변화가 많은 쪽을 기준으로 결정
  - ERD의 FK를 기준으로 결정
  - 단방향과 양방향
    - 데이터베이스의 연관관계는 PK/FK만으로 구성되지만 객체지향에서는 양방향 참조/단방향 참조의 형태가 가능
    - 양방향의 경우 복잡성이 증가하므로 주의

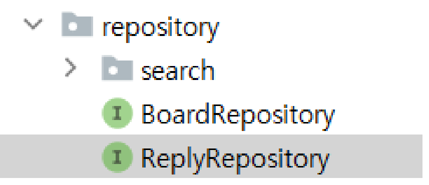
다대일 연관 관계의 구현



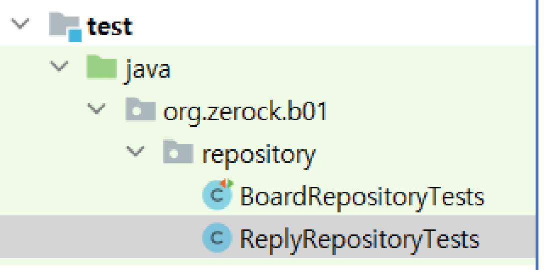
```java
package org.zerock.b01.domain;


import lombok.*;

import javax.persistence.*;

@Entity
@Getter
@Builder
@AllArgsConstructor
@NoArgsConstructor
@ToString(exclude = "board")
public class Reply extends BaseEntity{

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long rno;

    @ManyToOne(fetch = FetchType.LAZY)
    private Board board;

    private String replyText;

    private String replyer;

}
```



ReplyRepository의 생성과 테스트



```
repository
  search
  BoardRepository
  ReplyRepository
```

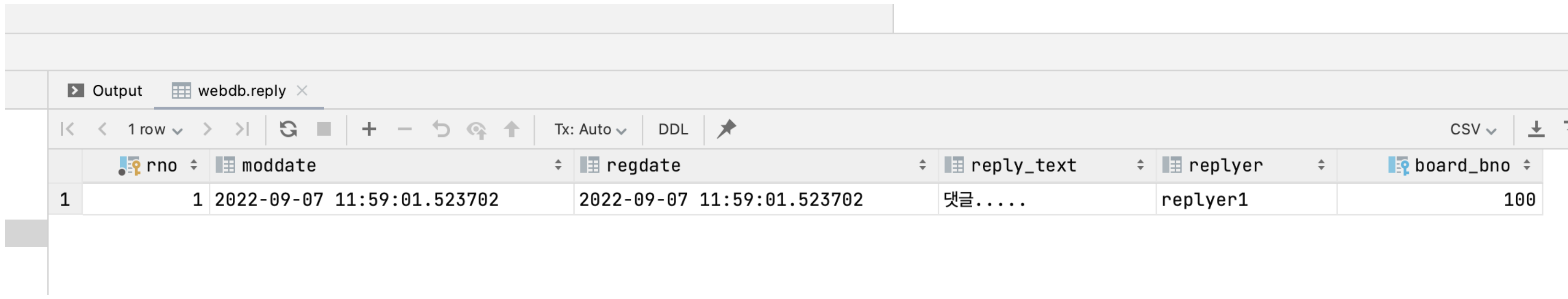


```
test
  java
    org.zerock.b01
      repository
        BoardRepositoryTests
        ReplyRepositoryTests
```

```java
package org.zerock.b01.repository;

import lombok.extern.log4j.Log4j2;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.zerock.b01.domain.Board;
import org.zerock.b01.domain.Reply;

@SpringBootTest
@Log4j2
public class ReplyRepositoryTests {

  @Autowired
  private ReplyRepository replyRepository;

  @Test
  public void testInsert() {

    //실제 DB에 있는 bno
    Long bno  = 100L;

    Board board = Board.builder().bno(bno).build();

    Reply reply = Reply.builder()
            .board(board)
            .replyText("댓글.....")
            .replyer("replyer1")
            .build();

    replyRepository.save(reply);

  }

}
```

```java
package org.zerock.b01.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.zerock.b01.domain.Reply;

public interface ReplyRepository extends JpaRepository<Reply, Long> {
}
```

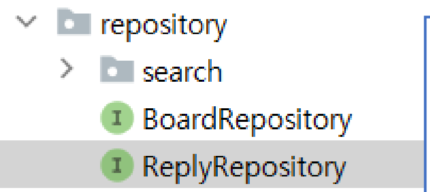
```
insert
into
    reply
    (moddate, regdate, board_bno, reply_text, replyer)
values
    (?, ?, ?, ?, ?)
```



	rno	moddate	regdate	reply_text	replyer	board_bno
1	1	2022-09-07 11:59:01.523702	2022-09-07 11:59:01.523702	댓글.....	replyer1	100

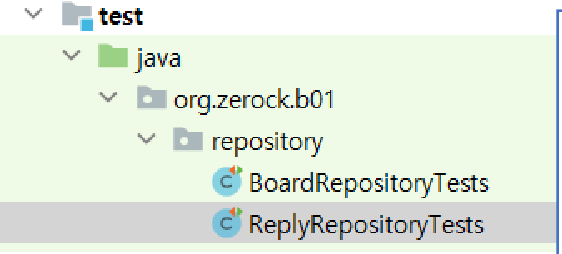
특정 게시물의 댓글을 위한 인덱스 생성

```java
@Entity
@Table(name = "Reply", indexes = {
        @Index(name = "idx_reply_board_bno", columnList = "board_bno")
})
@Getter
@Builder
@AllArgsConstructor
@NoArgsConstructor
@ToString(exclude = "board")
public class Reply extends BaseEntity{

    ...

}
```



```
repository
  search
  BoardRepository
  ReplyRepository
```

```java
package org.zerock.b01.repository;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.zerock.b01.domain.Reply;

public interface ReplyRepository extends JpaRepository<Reply, Long> {

    @Query("select r from Reply r where r.board.bno = :bno")
    Page<Reply> listOfBoard(Long bno, Pageable pageable);
}
```



```
@Test
public void testBoardReplies() {

  Long bno = 100L;

  Pageable pageable = PageRequest.of(0,10, Sort.by("rno").descending());

  Page<Reply> result = replyRepository.listOfBoard(bno, pageable);

  result.getContent().forEach(reply -> {
    log.info(reply);
  });
}
```

```
Hibernate:
  select
    reply0_.rno as rno1_1_,
    reply0_.moddate as moddate2_1_,
    reply0_.regdate as regdate3_1_,
    reply0_.board_bno as board_bn6_1_,
    reply0_.reply_text as reply_te4_1_,
    reply0_.replyer as replyer5_1_
  from
    reply reply0_
  where
    reply0_.board_bno=?
  order by
    reply0_.rno desc limit ?
```

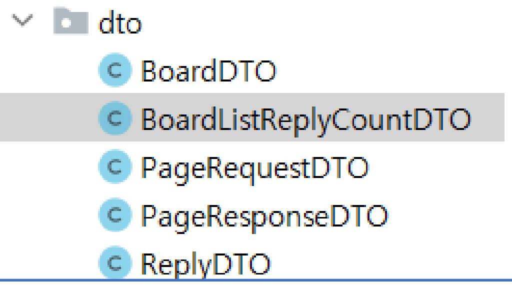
```
Reply(rno=4, replyText=댓글....., replyer=replyer1)
Reply(rno=3, replyText=댓글....., replyer=replyer1)
Reply(rno=2, replyText=댓글....., replyer=replyer1)
Reply(rno=1, replyText=댓글....., replyer=replyer1)
```

Fetch 모드

- FetchType.LAZY
  - 자신이 참조하고 있는 다른 엔티티에 대한 로딩은 최대한 지연한다. 필요한 순간까지 로딩하지 않는다.

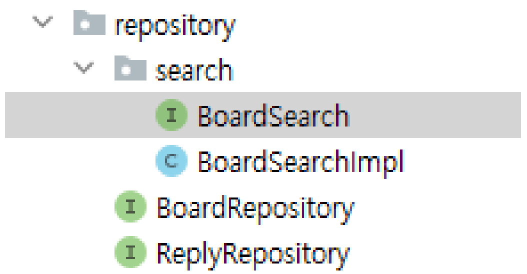
- FetchType.EAGER
  - 엔티티의 로딩 시에 적극적으로 참조관계가 있는 엔티티를 같이 로딩

게시물의 목록과 Projections



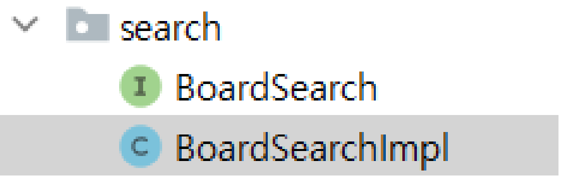
```
dto
    BoardDTO
    BoardListReplyCountDTO
    PageRequestDTO
    PageResponseDTO
    ReplyDTO
```

```java
package org.zerock.b01.dto;

import lombok.Data;

import java.time.LocalDateTime;

@Data
public class BoardListReplyCountDTO {

    private Long bno;

    private String title;

    private String writer;

    private LocalDateTime regDate;

    private Long replyCount;

}
```

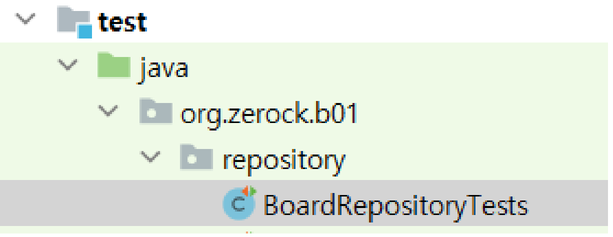


```
repository
    search
        BoardSearch
        BoardSearchImpl
    BoardRepository
    ReplyRepository
```

```java
public interface BoardSearch {

    Page<Board> search1(Pageable pageable);

    Page<Board> searchAll(String[] types, String keyword, Pageable pageable);

    Page<BoardListReplyCountDTO> searchWithReplyCount(String[] types,
                                                     String keyword,
                                                     Pageable pageable);

}
```



```
@Override
public Page<BoardListReplyCountDTO> searchWithReplyCount(String[] types, String keyword, Pageable pageable) {

    QBoard board = QBoard.board;
    QReply reply = QReply.reply;

    JPQLQuery<Board> query = from(board);
    query.leftJoin(reply).on(reply.board.eq(board));

    query.groupBy(board);

    if( (types != null && types.length > 0) && keyword != null ){

        BooleanBuilder booleanBuilder = new BooleanBuilder(); // (

        for(String type: types){

            switch (type){
                case "t":
                    booleanBuilder.or(board.title.contains(keyword));
                    break;
                case "c":
                    booleanBuilder.or(board.content.contains(keyword));
                    break;
                case "w":
                    booleanBuilder.or(board.writer.contains(keyword));
                    break;
            }
        }//end for
        query.where(booleanBuilder);
    }

    //bno > 0
    query.where(board.bno.gt(0L));

    JPQLQuery<BoardListReplyCountDTO> dtoQuery = query.select(Projections.bean(BoardListReplyCountDTO.class,
            board.bno,
            board.title,
            board.writer,
            board.regDate,
            reply.count().as("replyCount")
            ));

    this.getQuerydsl().applyPagination(pageable,dtoQuery);

    List<BoardListReplyCountDTO> dtoList = dtoQuery.fetch();

    long count = dtoQuery.fetchCount();

    return new PageImpl<>(dtoList, pageable, count);
```

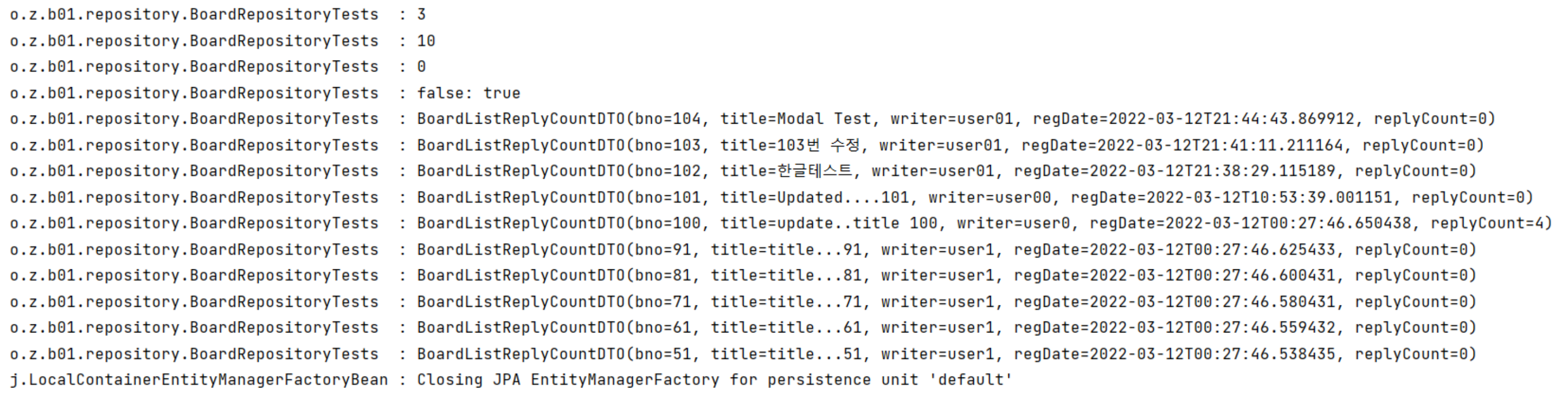


```
test
  java
    org.zerock.b01
      repository
        BoardRepositoryTests
```

```java
@Test
public void testSearchReplyCount() {

    String[] types = {"t","c","w"};

    String keyword = "1";

    Pageable pageable = PageRequest.of(0,10, Sort.by("bno").descending());

    Page<BoardListReplyCountDTO> result =
boardRepository.searchWithReplyCount(types, keyword, pageable );

    //total pages
    log.info(result.getTotalPages());
    //pag size
    log.info(result.getSize());
    //pageNumber
    log.info(result.getNumber());
    //prev next
    log.info(result.hasPrevious() +": " + result.hasNext());

    result.getContent().forEach(board -> log.info(board));
}
```

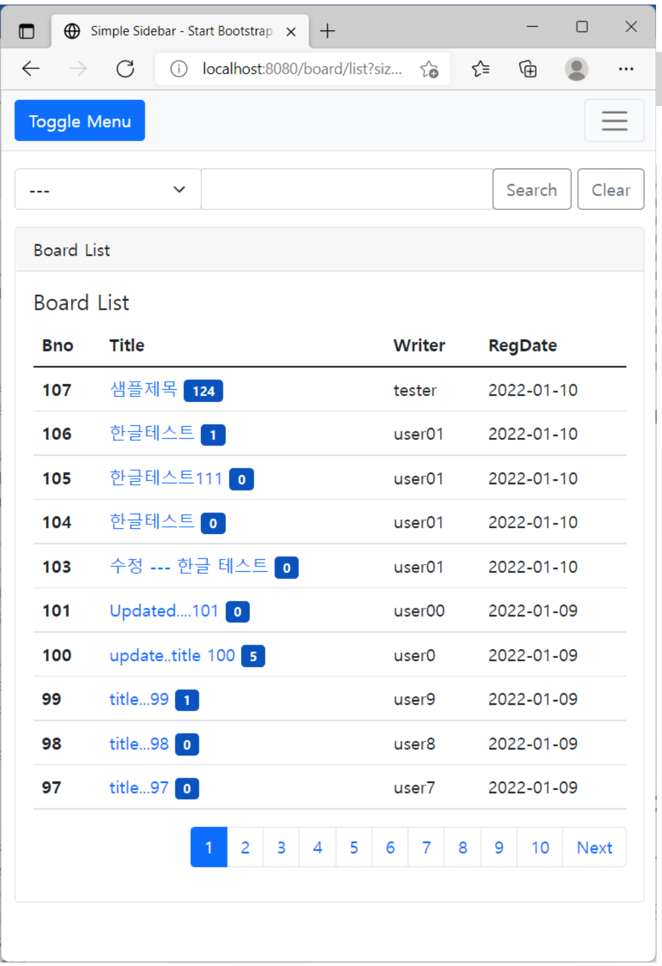
```
Hibernate:
  select
    board0_.bno as col_0_0_,
    board0_.title as col_1_0_,
    board0_.writer as col_2_0_,
    board0_.regdate as col_3_0_,
    count(reply1_.rno) as col_4_0_
  from
    board board0_
  left outer join
    reply reply1_
      on (
        reply1_.board_bno=board0_.bno
      )
  where
    (
      board0_.title like ? escape '!'
      or board0_.content like ? escape '!'
      or board0_.writer like ? escape '!'
    )
    and board0_.bno>?
  group by
    board0_.bno
  order by
    board0_.bno desc limit ?
```

```
select
    count(distinct board0_.bno) as col_0_0_
  from
    board board0_
  left outer join
    reply reply1_
      on (
        reply1_.board_bno=board0_.bno
      )
  where
    (
      board0_.title like ? escape '!'
      or board0_.content like ? escape '!'
      or board0_.writer like ? escape '!'
    )
    and board0_.bno>?
```



```
o.z.b01.repository.BoardRepositoryTests  : 3
o.z.b01.repository.BoardRepositoryTests  : 10
o.z.b01.repository.BoardRepositoryTests  : 0
o.z.b01.repository.BoardRepositoryTests  : false: true
o.z.b01.repository.BoardRepositoryTests  : BoardListReplyCountDTO(bno=104, title=Modal Test, writer=user01, regDate=2022-03-12T21:44:43.869912, replyCount=0)
o.z.b01.repository.BoardRepositoryTests  : BoardListReplyCountDTO(bno=103, title=103번 수정, writer=user01, regDate=2022-03-12T21:41:11.211164, replyCount=0)
o.z.b01.repository.BoardRepositoryTests  : BoardListReplyCountDTO(bno=102, title=한글테스트, writer=user01, regDate=2022-03-12T21:38:29.115189, replyCount=0)
o.z.b01.repository.BoardRepositoryTests  : BoardListReplyCountDTO(bno=101, title=Updated....101, writer=user00, regDate=2022-03-12T10:53:39.001151, replyCount=0)
o.z.b01.repository.BoardRepositoryTests  : BoardListReplyCountDTO(bno=100, title=update..title 100, writer=user0, regDate=2022-03-12T00:27:46.650438, replyCount=4)
o.z.b01.repository.BoardRepositoryTests  : BoardListReplyCountDTO(bno=91, title=title...91, writer=user1, regDate=2022-03-12T00:27:46.625433, replyCount=0)
o.z.b01.repository.BoardRepositoryTests  : BoardListReplyCountDTO(bno=81, title=title...81, writer=user1, regDate=2022-03-12T00:27:46.600431, replyCount=0)
o.z.b01.repository.BoardRepositoryTests  : BoardListReplyCountDTO(bno=71, title=title...71, writer=user1, regDate=2022-03-12T00:27:46.580431, replyCount=0)
o.z.b01.repository.BoardRepositoryTests  : BoardListReplyCountDTO(bno=61, title=title...61, writer=user1, regDate=2022-03-12T00:27:46.559432, replyCount=0)
o.z.b01.repository.BoardRepositoryTests  : BoardListReplyCountDTO(bno=51, title=title...51, writer=user1, regDate=2022-03-12T00:27:46.538435, replyCount=0)
j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
```

게시물의 목록 화면 처리



Simple Sidebar - Start Bootstrap
localhost:8080/board/list?siz...
Toggle Menu

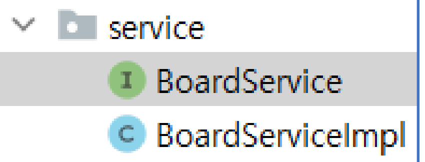
Search
Clear

Board List

Board List

Bno	Title	Writer	RegDate
107	샘플제목 124	tester	2022-01-10
106	한글테스트 1	user01	2022-01-10
105	한글테스트111 0	user01	2022-01-10
104	한글테스트 0	user01	2022-01-10
103	수정 --- 한글 테스트 0	user01	2022-01-10
101	Updated....101 0	user00	2022-01-09
100	update..title 100 5	user0	2022-01-09
99	title...99 1	user9	2022-01-09
98	title...98 0	user8	2022-01-09
97	title...97 0	user7	2022-01-09

1 2 3 4 5 6 7 8 9 10 Next



service
BoardService
BoardServiceImpl

```java
public interface BoardService {

    Long register(BoardDTO boardDTO);

    BoardDTO readOne(Long bno);

    void modify(BoardDTO boardDTO);

    void remove(Long bno);

    PageResponseDTO<BoardDTO> list(PageRequestDTO pageRequestDTO);

    //댓글의 숫자까지 처리
    PageResponseDTO<BoardListReplyCountDTO> listWithReplyCount(PageRequestDTO
pageRequestDTO);
}
```

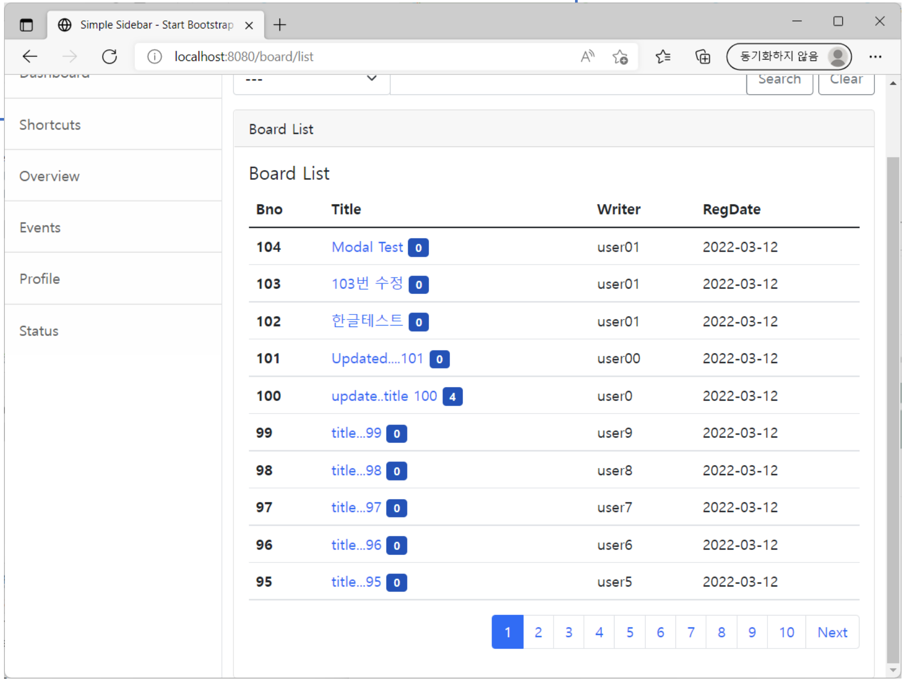
```java
@Override
public PageResponseDTO<BoardListReplyCountDTO> listWithReplyCount(PageRequestDTO
pageRequestDTO) {

    String[] types = pageRequestDTO.getTypes();
    String keyword = pageRequestDTO.getKeyword();
    Pageable pageable = pageRequestDTO.getPageable("bno");

    Page<BoardListReplyCountDTO> result = boardRepository.searchWithReplyCount(types, keyword,
pageable);

    return PageResponseDTO.<BoardListReplyCountDTO>withAll()
            .pageRequestDTO(pageRequestDTO)
            .dtoList(result.getContent())
            .total((int)result.getTotalElements())
            .build();
}
```

```
controller
  > advice
    BoardController
```

```java
@GetMapping("/list")
public void list(PageRequestDTO pageRequestDTO, Model model){

    //PageResponseDTO<BoardDTO> responseDTO = boardService.list(pageRequestDTO);

    PageResponseDTO<BoardListReplyCountDTO> responseDTO =
            boardService.listWithReplyCount(pageRequestDTO);

    log.info(responseDTO);

    model.addAttribute("responseDTO", responseDTO);
}
```

```
templates
  board
    list.html
    modify.html
```

```html
<tbody th:with="link = ${pageRequestDTO.getLink()}">
<tr th:each="dto:${responseDTO.dtoList}"  >
  <th scope="row">[[${dto.bno}]]</th>
  <td>
    <a th:href="|@{/board/read(bno =${dto.bno})}&${link}|" class="text-decoration-none"> [[${dto.title}]] </a>
    <span class="badge progress-bar-success" style="background-color: #0a53be">[[${dto.replyCount}]]</span>
  </td>
  <td>[[${dto.writer}]]</td>
  <td>[[${#temporals.format(dto.regDate, 'yyyy-MM-dd')}]]</td>
</tr>
</tbody>
```

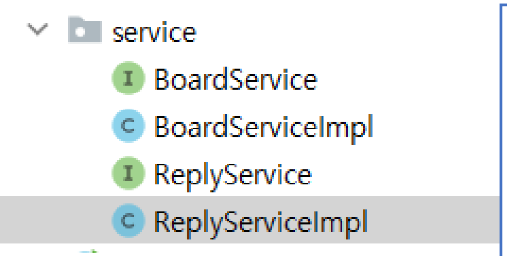


Board List

Bno	Title	Writer	RegDate
104	Modal Test 0	user01	2022-03-12
103	103번 수정 0	user01	2022-03-12
102	한글테스트 0	user01	2022-03-12
101	Updated....101 0	user00	2022-03-12
100	update..title 100 4	user0	2022-03-12
99	title...99 0	user9	2022-03-12
98	title...98 0	user8	2022-03-12
97	title...97 0	user7	2022-03-12
96	title...96 0	user6	2022-03-12
95	title...95 0	user5	2022-03-12

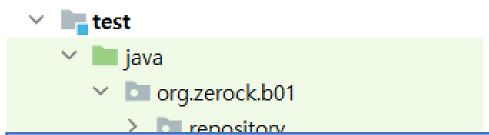
1 2 3 4 5 6 7 8 9 10 Next

댓글 서비스 계층의 구현



```java
package org.zerock.b01.service;

import org.zerock.b01.dto.ReplyDTO;

public interface ReplyService {

    Long register(ReplyDTO replyDTO);
}
```

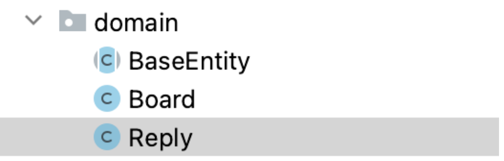
```java
package org.zerock.b01.service;

import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.modelmapper.ModelMapper;
import org.springframework.stereotype.Service;
import org.zerock.b01.domain.Reply;
import org.zerock.b01.dto.ReplyDTO;
import org.zerock.b01.repository.ReplyRepository;


@Service
@RequiredArgsConstructor
@Log4j2
public class ReplyServiceImpl implements ReplyService{

    private final ReplyRepository replyRepository;

    private final ModelMapper modelMapper;

    @Override
    public Long register(ReplyDTO replyDTO) {

        Reply reply = modelMapper.map(replyDTO, Reply.class);

        Long rno = replyRepository.save(reply).getRno();

        return rno;
    }
}
```



```java
package org.zerock.b01.service;

import lombok.extern.log4j.Log4j2;
import org.junit.jupiter.api.Test;
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.boot.test.context.SpringBootTest;
import org.zerock.b01.dto.ReplyDTO;

@SpringBootTest
@Log4j2
public class ReplyServiceTests {

    @Autowired
    private ReplyService replyService;

    @Test
    public void testRegister() {

        ReplyDTO replyDTO = ReplyDTO.builder()
                .replyText("ReplyDTO Text")
                .replyer("replyer")
                .bno(100L)
                .build();

        log.info(replyService.register(replyDTO));
    }

}
```

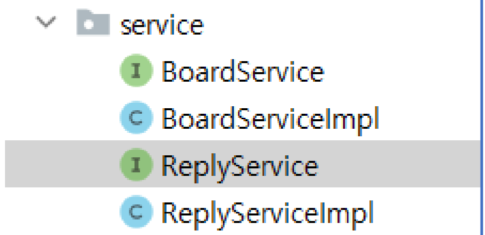
```
Hibernate:
    insert
    into
        reply
        (moddate, regdate, board_bno, reply_text, replyer)
    values
        (?, ?, ?, ?, ?)
```

댓글 조회/수정/삭제/목록



```
domain
    BaseEntity
    Board
    Reply
```

```java
public class Reply extends BaseEntity{

    @Id
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long rno;

    @ManyToOne(fetch = FetchType.LAZY)
    private Board board;

    private String replyText;

    private String replyer;

    public void changeText(String text){
        this.replyText = text;
    }
}
```

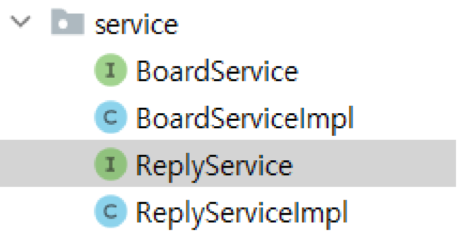


```
service
    BoardService
    BoardServiceImpl
    ReplyService
    ReplyServiceImpl
```

```java
public interface ReplyService {

    Long register(ReplyDTO replyDTO);

    ReplyDTO read(Long rno);

    void modify(ReplyDTO replyDTO);

    void remove(Long rno);
}
```

```java
@Override
public ReplyDTO read(Long rno) {

    Optional<Reply> replyOptional = replyRepository.findById(rno);

    Reply reply = replyOptional.orElseThrow();

    return modelMapper.map(reply, ReplyDTO.class);
}

@Override
public void modify(ReplyDTO replyDTO) {

    Optional<Reply> replyOptional =
replyRepository.findById(replyDTO.getRno());

    Reply reply = replyOptional.orElseThrow();

    reply.changeText(replyDTO.getReplyText());

    replyRepository.save(reply);

}

@Override
public void remove(Long rno) {

    replyRepository.deleteById(rno);
```

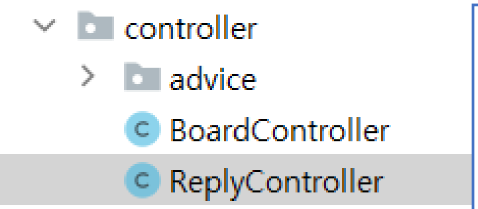
특정 게시물의 목록 처리



```
public interface ReplyService {

    ...

    PageResponseDTO<ReplyDTO> getListOfBoard(Long bno,
                                            PageRequestDTO pageRequestDTO);
}
```

```
@Override
public PageResponseDTO<ReplyDTO> getListOfBoard(Long bno, PageRequestDTO pageRequestDTO) {

    Pageable pageable = PageRequest.of(pageRequestDTO.getPage() <=0? 0: pageRequestDTO.getPage()
-1,
            pageRequestDTO.getSize(),
            Sort.by("rno").ascending());

    Page<Reply> result = replyRepository.listOfBoard(bno, pageable);

    List<ReplyDTO> dtoList =
            result.getContent().stream().map(reply -> modelMapper.map(reply, ReplyDTO.class))
                    .collect(Collectors.toList());

    return PageResponseDTO.<ReplyDTO>withAll()
            .pageRequestDTO(pageRequestDTO)
            .dtoList(dtoList)
            .total((int)result.getTotalElements())
            .build();
}
```

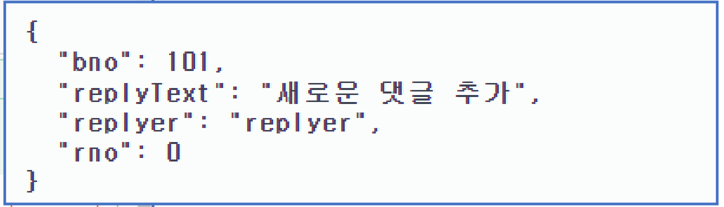
컨트롤러 계층의 처리



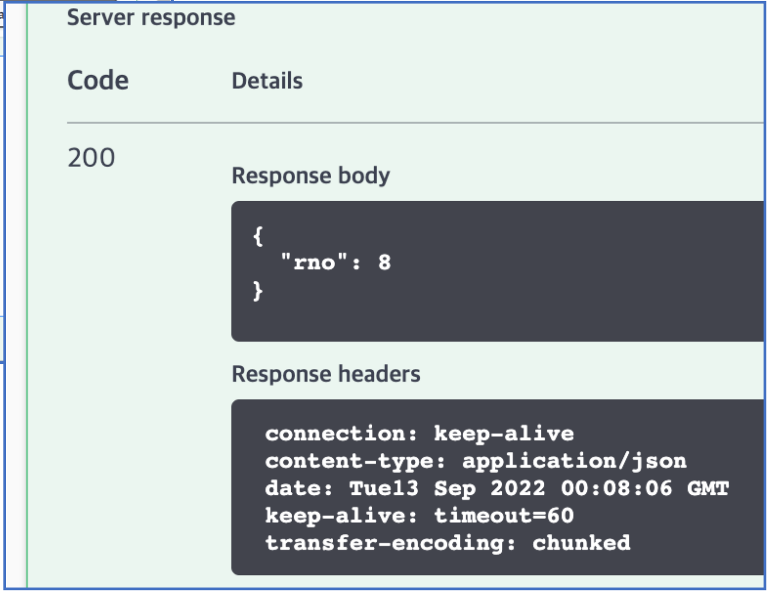
```
@RestController
@RequestMapping("/replies")
@Log4j2
@RequiredArgsConstructor   //의존성 주입을 위한
public class ReplyController {

    private final ReplyService replyService;



    ...
}
```

등록 기능의 확인

```java
@ApiOperation(value = "Replies POST", notes = "POST 방식으로 댓글 등록")
@PostMapping(value = "/", consumes = MediaType.APPLICATION_JSON_VALUE)
public Map<String,Long> register(
      @Valid @RequestBody ReplyDTO replyDTO,
      BindingResult bindingResult)throws BindException{

   log.info(replyDTO);

   if(bindingResult.hasErrors()){
      throw new BindException(bindingResult);
   }

   Map<String, Long> resultMap = new HashMap<>();

   Long rno = replyService.register(replyDTO);

   resultMap.put("rno",rno);

   return resultMap;
}
```



```json
{
  "bno": 101,
  "replyText": "새로운 댓글 추가",
  "replyer": "replyer",
  "rno": 0
}
```



Server response

Code | Details

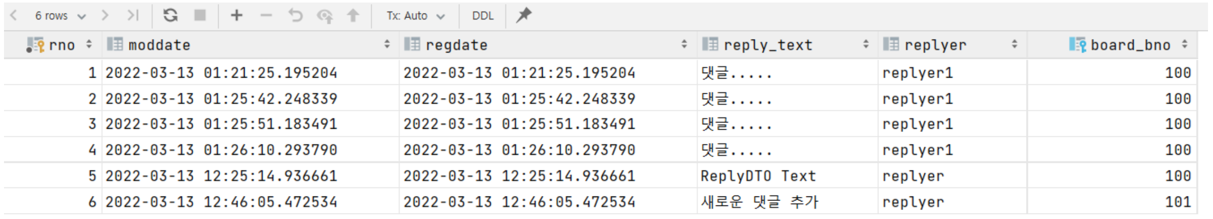
200

Response body

```
{
  "rno": 8
}
```

Response headers

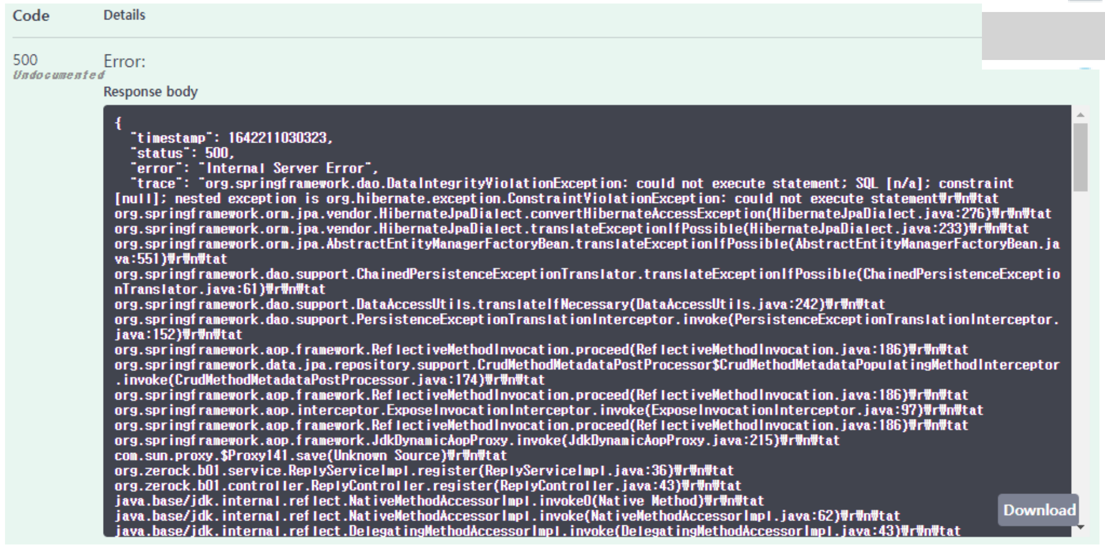
```
connection: keep-alive
content-type: application/json
date: Tue13 Sep 2022 00:08:06 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```



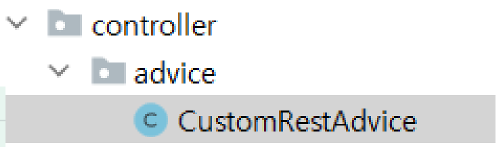
rno	moddate	regdate	reply_text	replyer	board_bno
1	2022-03-13 01:21:25.195204	2022-03-13 01:21:25.195204	댓글.....	replyer1	100
2	2022-03-13 01:25:42.248339	2022-03-13 01:25:42.248339	댓글.....	replyer1	100
3	2022-03-13 01:25:51.183491	2022-03-13 01:25:51.183491	댓글.....	replyer1	100
4	2022-03-13 01:26:10.293790	2022-03-13 01:26:10.293790	댓글.....	replyer1	100
5	2022-03-13 12:25:14.936661	2022-03-13 12:25:14.936661	ReplyDTO Text	replyer	100
6	2022-03-13 12:46:05.472534	2022-03-13 12:46:05.472534	새로운 댓글 추가	replyer	101

잘못된 상황에 대한 처리

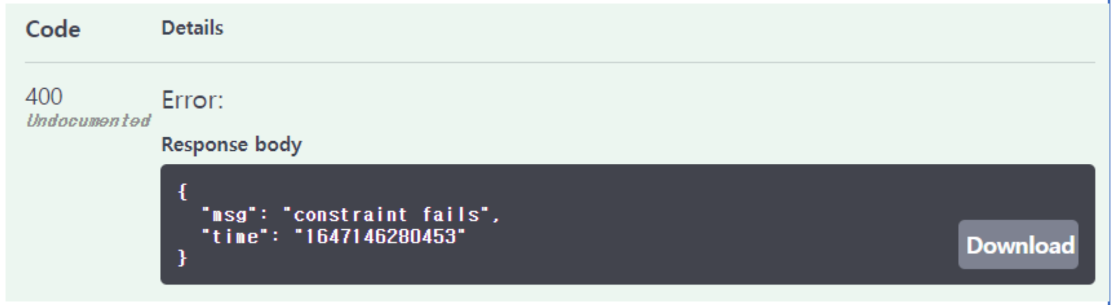
- @Valid의 검증 결과 예외 처리



```
{
  "timestamp": 1642211030323,
  "status": 500,
  "error": "Internal Server Error",
  "trace": "org.springframework.dao.DataIntegrityViolationException: could not execute statement; SQL [n/a]; constraint
[null]; nested exception is org.hibernate.exception.ConstraintViolationException: could not execute statement₩r₩n₩tat
org.springframework.orm.jpa.vendor.HibernateJpaDialect.convertHibernateAccessException(HibernateJpaDialect.java:276)₩r₩n₩tat
org.springframework.orm.jpa.vendor.HibernateJpaDialect.translateExceptionIfPossible(HibernateJpaDialect.java:233)₩r₩n₩tat
org.springframework.orm.jpa.AbstractEntityManagerFactoryBean.translateExceptionIfPossible(AbstractEntityManagerFactoryBean.ja
va:551)₩r₩n₩tat
org.springframework.dao.support.ChainedPersistenceExceptionTranslator.translateExceptionIfPossible(ChainedPersistenceExceptio
nTranslator.java:61)₩r₩n₩tat
org.springframework.dao.support.DataAccessUtils.translateIfNecessary(DataAccessUtils.java:242)₩r₩n₩tat
org.springframework.dao.support.PersistenceExceptionTranslationInterceptor.invoke(PersistenceExceptionTranslationInterceptor.
java:152)₩r₩n₩tat
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)₩r₩n₩tat
org.springframework.data.jpa.repository.support.CrudMethodMetadataPostProcessor$CrudMethodMetadataPopulatingMethodInterceptor
.invoke(CrudMethodMetadataPostProcessor.java:174)₩r₩n₩tat
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)₩r₩n₩tat
org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(ExposeInvocationInterceptor.java:97)₩r₩n₩tat
org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)₩r₩n₩tat
org.springframework.aop.framework.JdkDynamicAopProxy.invoke(JdkDynamicAopProxy.java:215)₩r₩n₩tat
com.sun.proxy.$Proxy141.save(Unknown Source)₩r₩n₩tat
org.zerock.b01.service.ReplyServiceImpl.register(ReplyServiceImpl.java:36)₩r₩n₩tat
org.zerock.b01.controller.ReplyController.register(ReplyController.java:43)₩r₩n₩tat
java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)₩r₩n₩tat
java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)₩r₩n₩tat
java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)₩r₩n₩tat
```

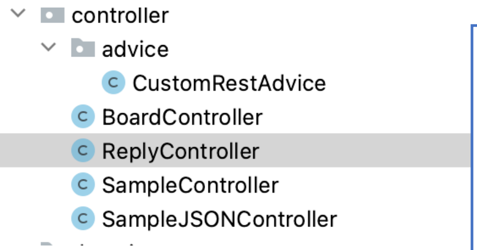


```java
package org.zerock.b01.controller.advice;

...

@RestControllerAdvice
@Log4j2
public class CustomRestAdvice {

    @ExceptionHandler(BindException.class)
    @ResponseStatus(HttpStatus.EXPECTATION_FAILED)
    public ResponseEntity<Map<String, String>>
handleBindException(BindException e) {

        log.error(e);

        Map<String, String> errorMap = new HashMap<>();

        if(e.hasErrors()){

            BindingResult bindingResult = e.getBindingResult();

            bindingResult.getFieldErrors().forEach(fieldError -> {
                errorMap.put(fieldError.getField(), fieldError.getCode());
            });
        }

        return ResponseEntity.badRequest().body(errorMap);
    }

    @ExceptionHandler(DataIntegrityViolationException.class)
    @ResponseStatus(HttpStatus.EXPECTATION_FAILED)
    public ResponseEntity<Map<String, String>>
handleFKException(Exception e) {

        log.error(e);

        Map<String, String> errorMap = new HashMap<>();

        errorMap.put("time", ""+System.currentTimeMillis());
        errorMap.put("msg",  "constraint fails");
        return ResponseEntity.badRequest().body(errorMap);
    }

}
```



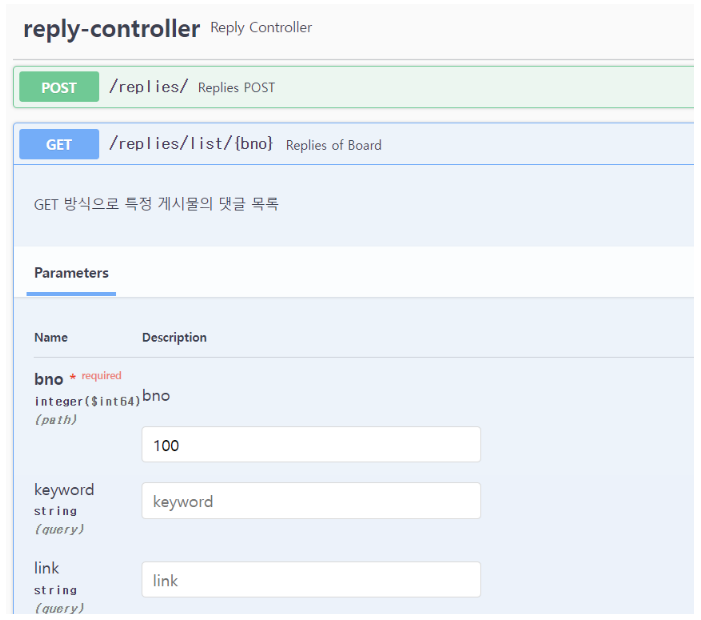
```
{
  "msg": "constraint fails",
  "time": "1647146280453"
}
```

특정 게시물의 댓글 목록



controller
advice
CustomRestAdvice
BoardController
ReplyController
SampleController
SampleJSONController

```java
@ApiOperation(value = "Replies of Board", notes = "GET 방식으로 특정 게시물의 댓글 목록")
@GetMapping(value = "/list/{bno}")
public PageResponseDTO<ReplyDTO> getList(@PathVariable("bno") Long bno,
                                         PageRequestDTO pageRequestDTO){

    PageResponseDTO<ReplyDTO> responseDTO = replyService.getListOfBoard(bno,
pageRequestDTO);

    return responseDTO;
}
```



reply-controller Reply Controller

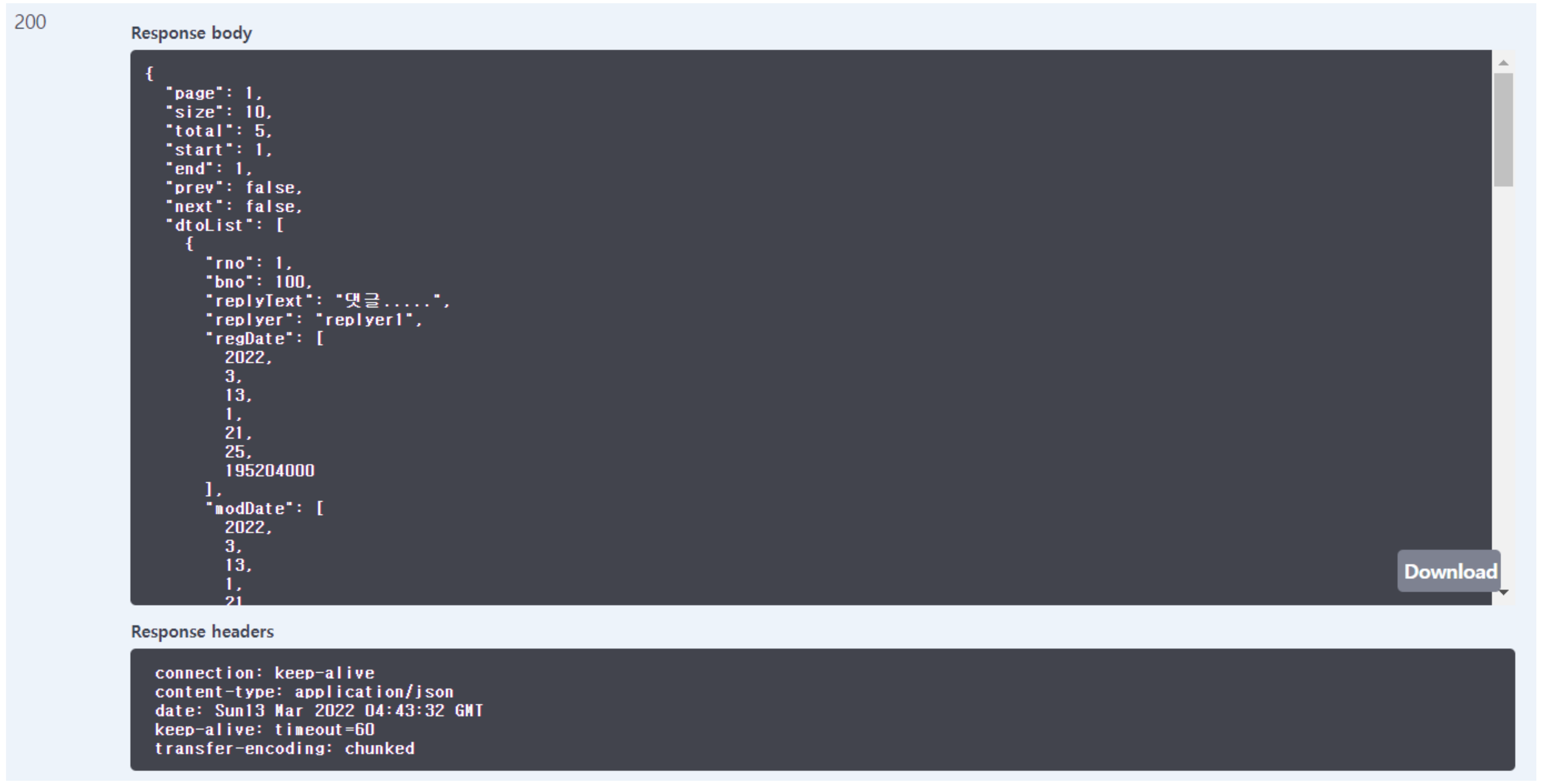
POST /replies/ Replies POST

GET /replies/list/{bno} Replies of Board

GET 방식으로 특정 게시물의 댓글 목록

Parameters

Name	Description
bno * required integer($int64) (path)	bno 100
keyword string (query)	keyword
link string (query)	link



200

Response body

```
{
  "page": 1,
  "size": 10,
  "total": 5,
  "start": 1,
  "end": 1,
  "prev": false,
  "next": false,
  "dtoList": [
    {
      "rno": 1,
      "bno": 100,
      "replyText": "댓글.....",
      "replyer": "replyer1",
      "regDate": [
        2022,
        3,
        13,
        1,
        21,
        25,
        195204000
      ],
      "modDate": [
        2022,
        3,
        13,
        1,
        21
```

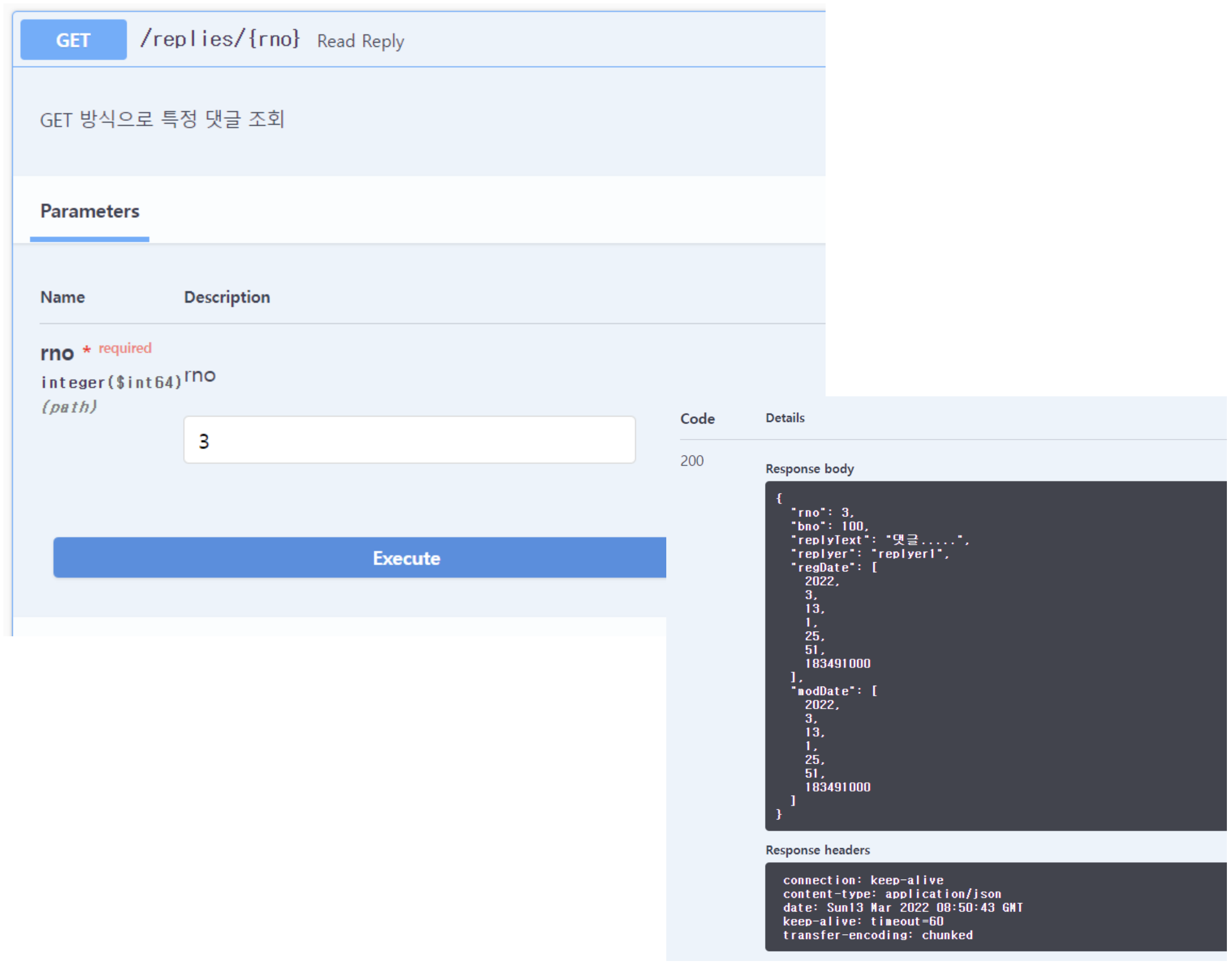
Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Sun13 Mar 2022 04:43:32 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

특정 댓글의 조회

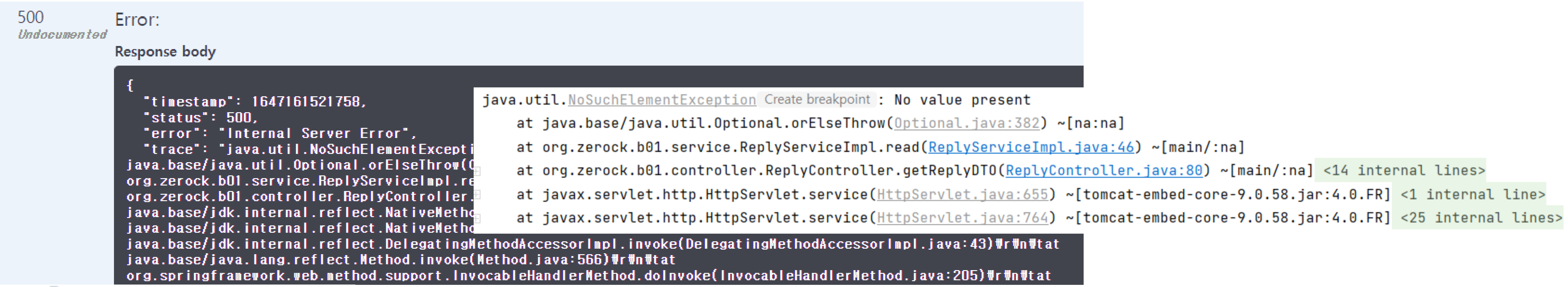
```java
@ApiOperation(value = "Read Reply", notes = "GET 방식으로 특정 댓글 조회")
@GetMapping("/{rno}")
public ReplyDTO getReplyDTO( @PathVariable("rno") Long rno ){

    ReplyDTO replyDTO = replyService.read(rno);

    return replyDTO;
}
```



```
{
  "rno": 3,
  "bno": 100,
  "replyText": "댓글.....",
  "replyer": "replyer1",
  "regDate": [
    2022,
    3,
    13,
    1,
    25,
    51,
    183491000
  ],
  "modDate": [
    2022,
    3,
    13,
    1,
    25,
    51,
    183491000
  ]
}
```

```
connection: keep-alive
content-type: application/json
date: Sun13 Mar 2022 08:50:43 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

데이터가 존재하지 않는 경우의 처리



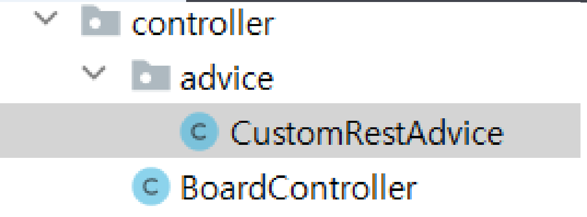
500
Undocumented

Error:

Response body

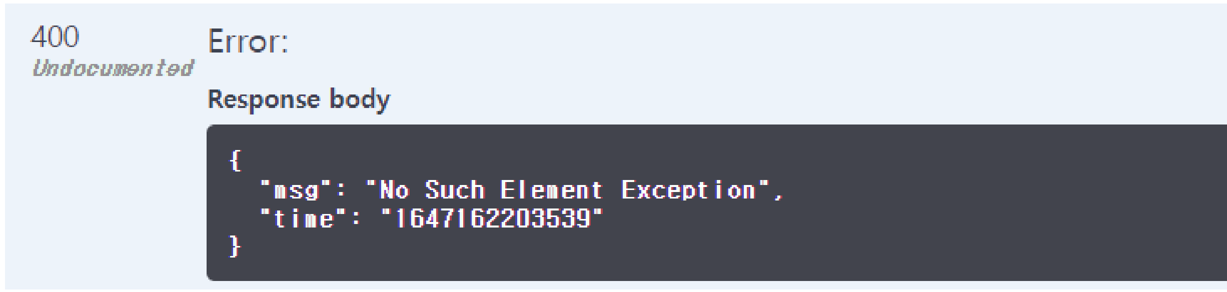
```
{
  "timestamp": 1647161521758,
  "status": 500,
  "error": "Internal Server Error",
  "trace": "java.util.NoSuchElementExcepti
java.base/java.util.Optional.orElseThrow(O
org.zerock.b01.service.ReplyServiceImpl.re
org.zerock.b01.controller.ReplyController.
java.base/jdk.internal.reflect.NativeMetho
java.base/jdk.internal.reflect.NativeMetho
java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)₩r₩n₩tat
java.base/java.lang.reflect.Method.invoke(Method.java:566)₩r₩n₩tat
org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:205)₩r₩n₩tat
```

```
java.util.NoSuchElementException Create breakpoint : No value present
	at java.base/java.util.Optional.orElseThrow(Optional.java:382) ~[na:na]
	at org.zerock.b01.service.ReplyServiceImpl.read(ReplyServiceImpl.java:46) ~[main/:na]
	at org.zerock.b01.controller.ReplyController.getReplyDTO(ReplyController.java:80) ~[main/:na] <14 internal lines>
	at javax.servlet.http.HttpServlet.service(HttpServlet.java:655) ~[tomcat-embed-core-9.0.58.jar:4.0.FR] <1 internal line>
	at javax.servlet.http.HttpServlet.service(HttpServlet.java:764) ~[tomcat-embed-core-9.0.58.jar:4.0.FR] <25 internal lines>
```



```
controller
  advice
    CustomRestAdvice
  BoardController
```

```java
@ExceptionHandler(NoSuchElementException.class)
@ResponseStatus(HttpStatus.EXPECTATION_FAILED)
public ResponseEntity<Map<String, String>> handleNoSuchElement(Exception
e) {

    log.error(e);

    Map<String, String> errorMap = new HashMap<>();

    errorMap.put("time", ""+System.currentTimeMillis());
    errorMap.put("msg",  "No Such Element Exception");
    return ResponseEntity.badRequest().body(errorMap);
}
```



400
Undocumented

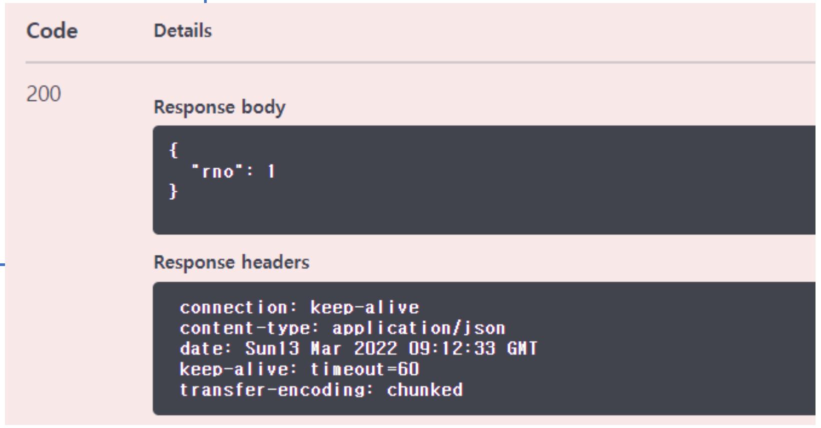
Error:

Response body

```
{
  "msg": "No Such Element Exception",
  "time": "1647162203539"
}
```

특정 댓글의 삭제

```java
@ApiOperation(value = "Delete Reply", notes = "DELETE 방식으로 특정 댓글 삭제")
@DeleteMapping("/{rno}")
public Map<String,Long> remove( @PathVariable("rno") Long rno ){

    replyService.remove(rno);

    Map<String, Long> resultMap = new HashMap<>();

    resultMap.put("rno", rno);

    return resultMap;
}
```



Code Details

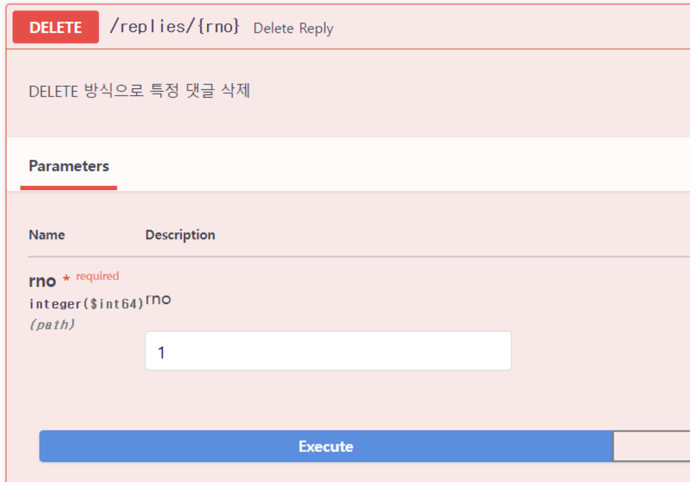
200

Response body

```
{
  "rno": 1
}
```

Response headers

```
 connection: keep-alive
 content-type: application/json
 date: Sun13 Mar 2022 09:12:33 GMT
 keep-alive: timeout=60
 transfer-encoding: chunked
```



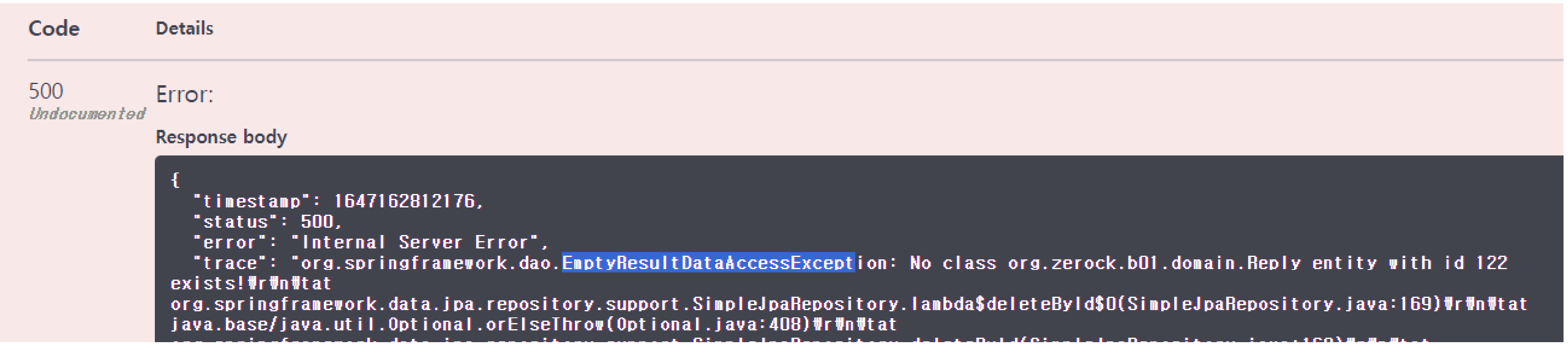
DELETE /replies/{rno} Delete Reply

DELETE 방식으로 특정 댓글 삭제

Parameters

Name	Description
rno * required integer($int64) (path)	rno 1

존재하지 않는 댓글의 삭제 문제

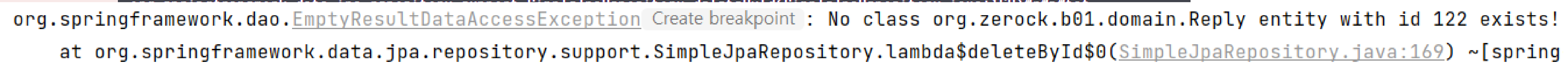


Code Details

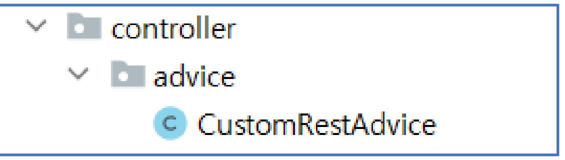
500 *Undocumented* Error:

Response body

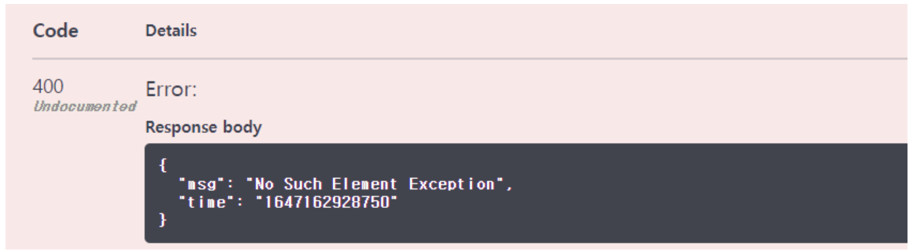
```
{
  "timestamp": 1647162812176,
  "status": 500,
  "error": "Internal Server Error",
  "trace": "org.springframework.dao.EmptyResultDataAccessException: No class org.zerock.b01.domain.Reply entity with id 122
exists!\r\n\tat
org.springframework.data.jpa.repository.support.SimpleJpaRepository.lambda$deleteById$0(SimpleJpaRepository.java:169)\r\n\tat
java.base/java.util.Optional.orElseThrow(Optional.java:408)\r\n\tat
```



```
org.springframework.dao.EmptyResultDataAccessException Create breakpoint : No class org.zerock.b01.domain.Reply entity with id 122 exists!
    at org.springframework.data.jpa.repository.support.SimpleJpaRepository.lambda$deleteById$0(SimpleJpaRepository.java:169) ~[spring
```



```java
@ExceptionHandler({
        NoSuchElementException.class,
        EmptyResultDataAccessException.class }) //추가
@ResponseStatus(HttpStatus.EXPECTATION_FAILED)
public ResponseEntity<Map<String, String>> handleNoSuchElement(Exception e) {

    log.error(e);

    Map<String, String> errorMap = new HashMap<>();

    errorMap.put("time", ""+System.currentTimeMillis());
    errorMap.put("msg",  "No Such Element Exception");
    return ResponseEntity.badRequest().body(errorMap);
}
```



Code Details

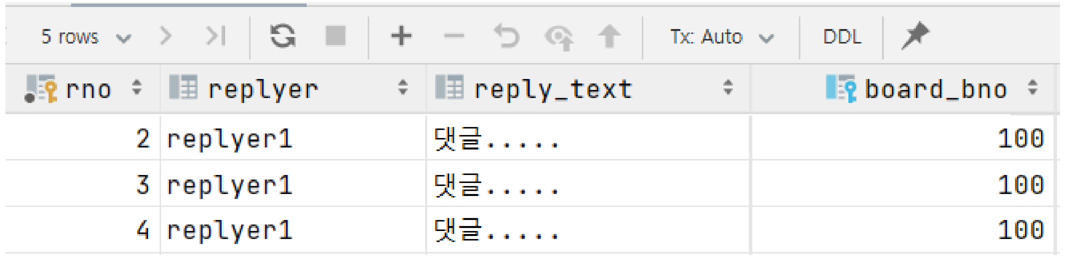
400 *Undocumented* Error:

Response body

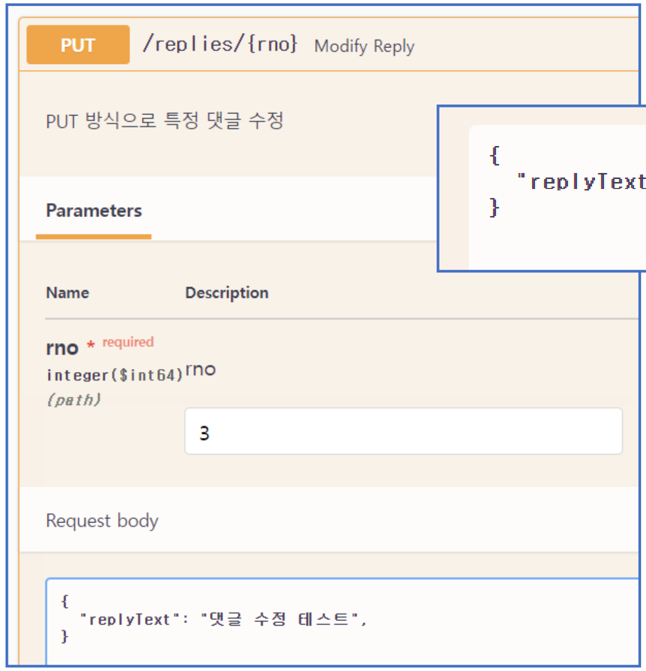
```
{
  "msg": "No Such Element Exception",
  "time": "1647162928750"
}
```

특정 댓글의 수정

```
@ApiOperation(value = "Modify Reply", notes = "PUT 방식으로 특정 댓글 수정")
@PutMapping(value = "/{rno}", consumes = MediaType.APPLICATION_JSON_VALUE )
public Map<String,Long> remove( @PathVariable("rno") Long rno, @RequestBody
ReplyDTO replyDTO ){

    replyDTO.setRno(rno); //번호를 일치시킴

    replyService.modify(replyDTO);

    Map<String, Long> resultMap = new HashMap<>();

    resultMap.put("rno", rno);

    return resultMap;
}
```



rno	replyer	reply_text	board_bno
2	replyer1	댓글.....	100
3	replyer1	댓글.....	100
4	replyer1	댓글.....	100



PUT /replies/{rno} Modify Reply

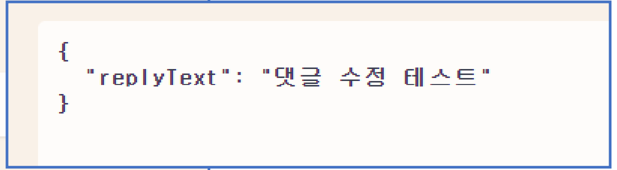
PUT 방식으로 특정 댓글 수정

Parameters

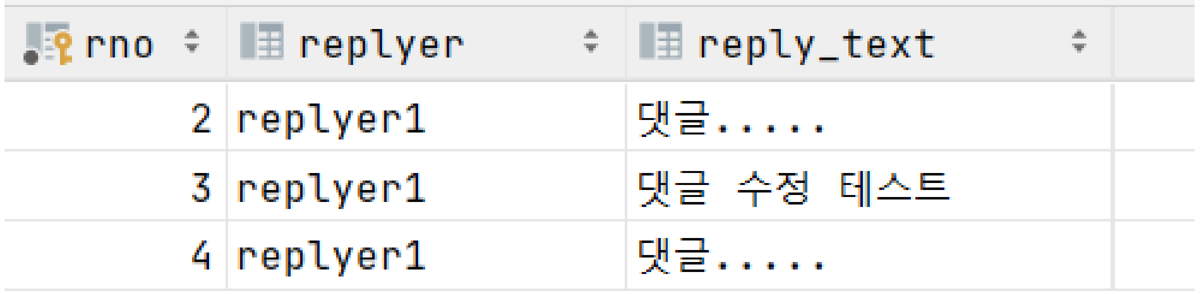
Name	Description
rno * required integer($int64) (path)	rno 3

Request body

```
{
  "replyText": "댓글 수정 테스트",
}
```



```
{
  "replyText": "댓글 수정 테스트"
}
```



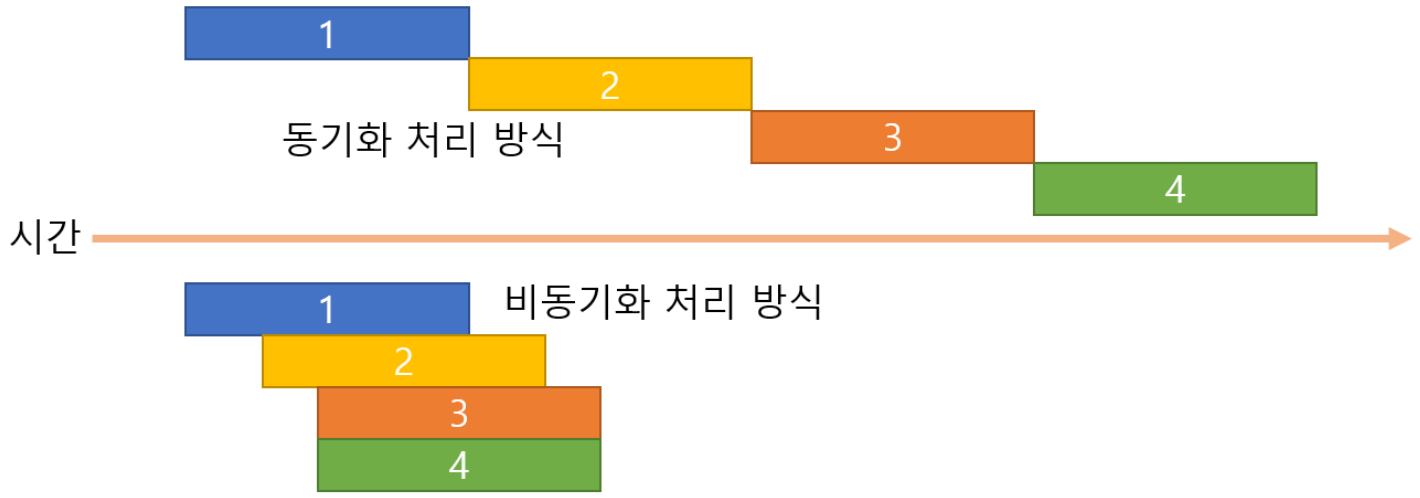
rno	replyer	reply_text
2	replyer1	댓글.....
3	replyer1	댓글 수정 테스트
4	replyer1	댓글.....

댓글의 JavaScript처리

- Ajax를 이용해서 GET/POST/PUT/DELETE방식등을 호출
- Axios라이브러리를 이용
- JavaScript를 이용한 동적 처리

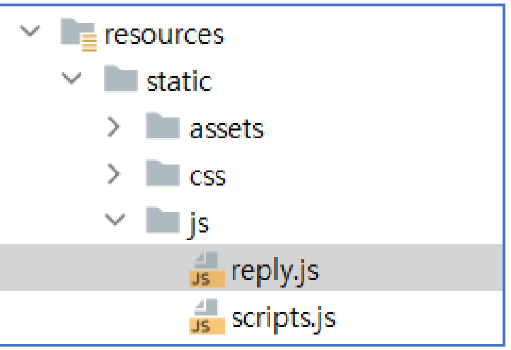
비동기 처리와 Ajax

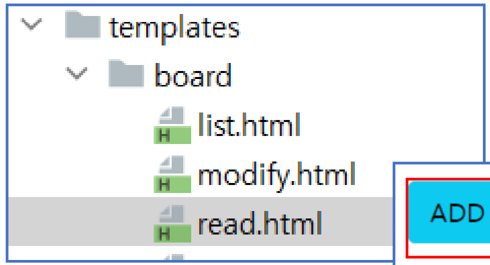
- 동기화 vs 비동기 처리
  - 콜백 패턴(callback)

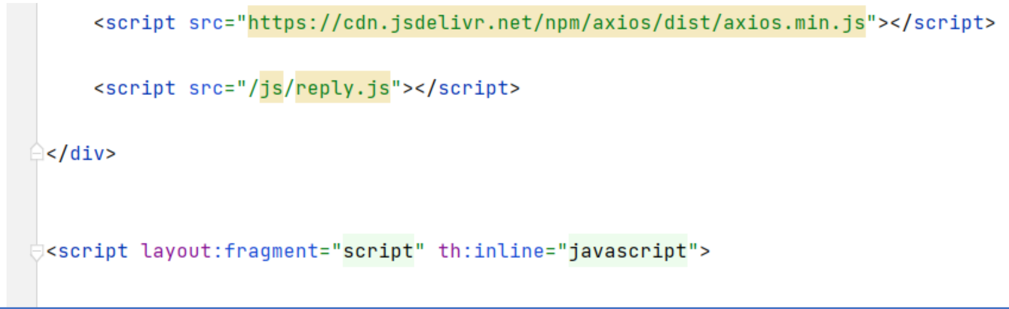


Axios

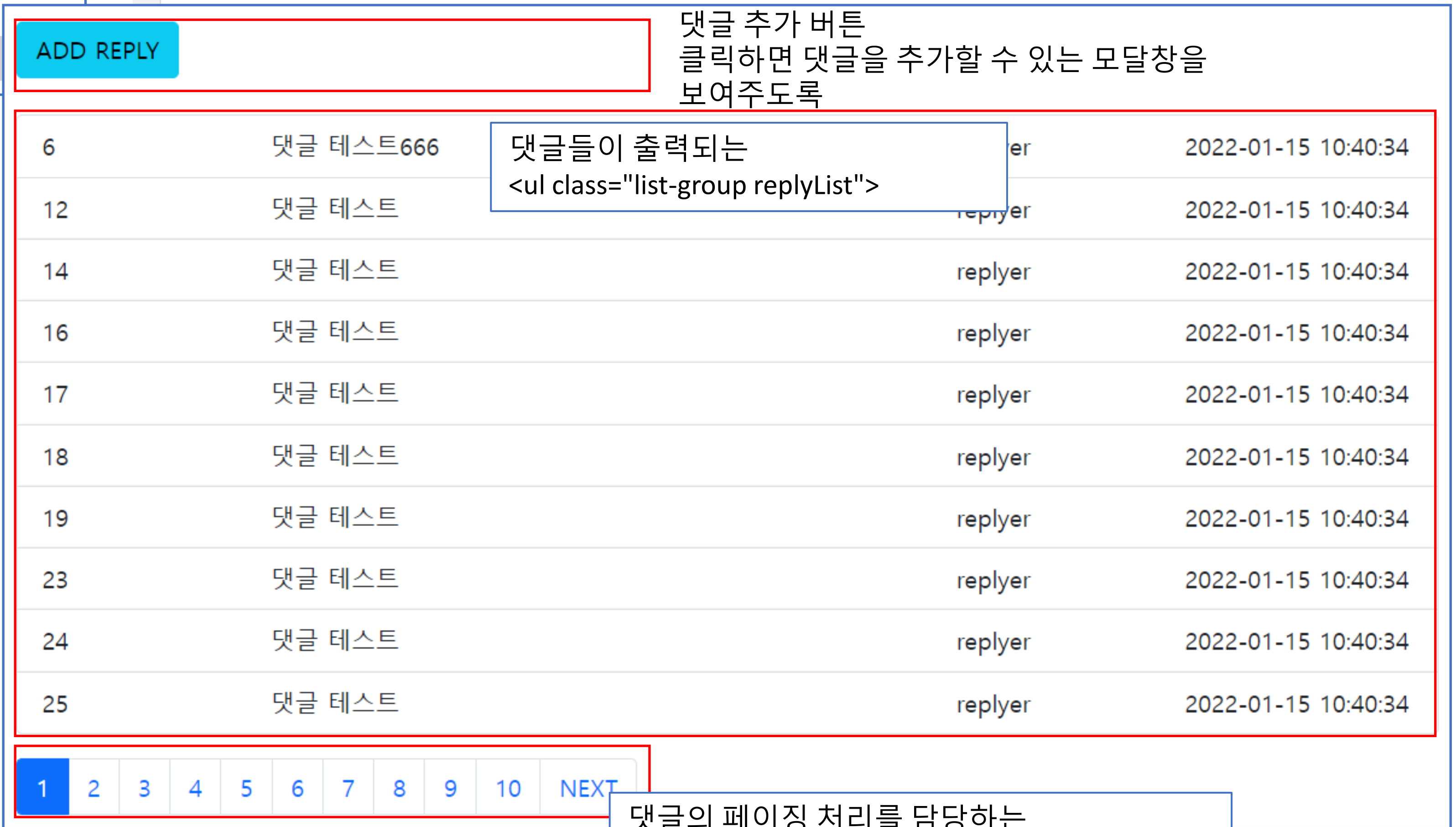
비동기 처리를 위한 라이브러리
Promise기반







```
    <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

    <script src="/js/reply.js"></script>

</div>

<script layout:fragment="script" th:inline="javascript">
```



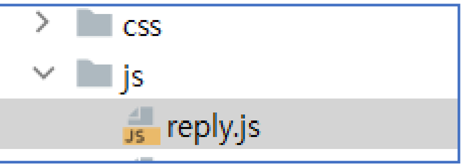
댓글 추가 버튼
클릭하면 댓글을 추가할 수 있는 모달창을 보여주도록

댓글들이 출력되는
<ul class="list-group replyList">

6	댓글 테스트666	er	2022-01-15 10:40:34
12	댓글 테스트	replyer	2022-01-15 10:40:34
14	댓글 테스트	replyer	2022-01-15 10:40:34
16	댓글 테스트	replyer	2022-01-15 10:40:34
17	댓글 테스트	replyer	2022-01-15 10:40:34
18	댓글 테스트	replyer	2022-01-15 10:40:34
19	댓글 테스트	replyer	2022-01-15 10:40:34
23	댓글 테스트	replyer	2022-01-15 10:40:34
24	댓글 테스트	replyer	2022-01-15 10:40:34
25	댓글 테스트	replyer	2022-01-15 10:40:34

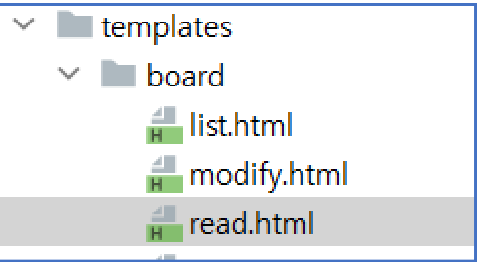
댓글의 페이징 처리를 담당하는

Axios호출해 보기



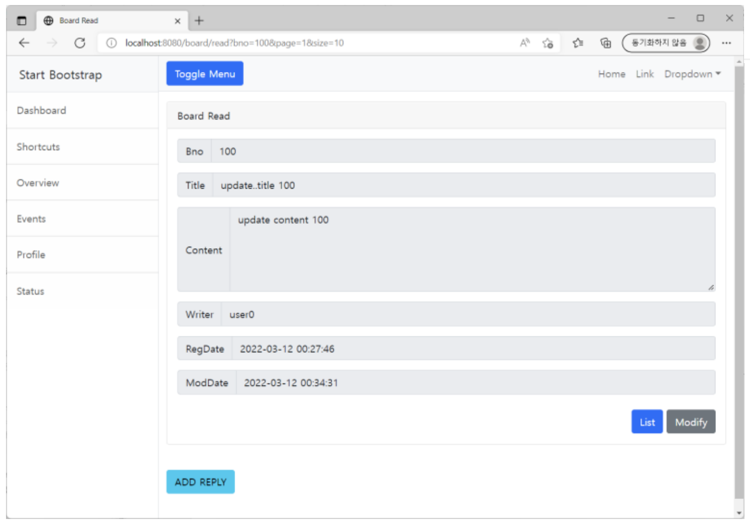
```
> css
v js
    reply.js
```

```
async function get1(bno) {

    const result = await axios.get(`/replies/list/${bno}`)

    console.log(result)

}
```



```
v templates
  v board
      list.html
      modify.html
      read.html
```

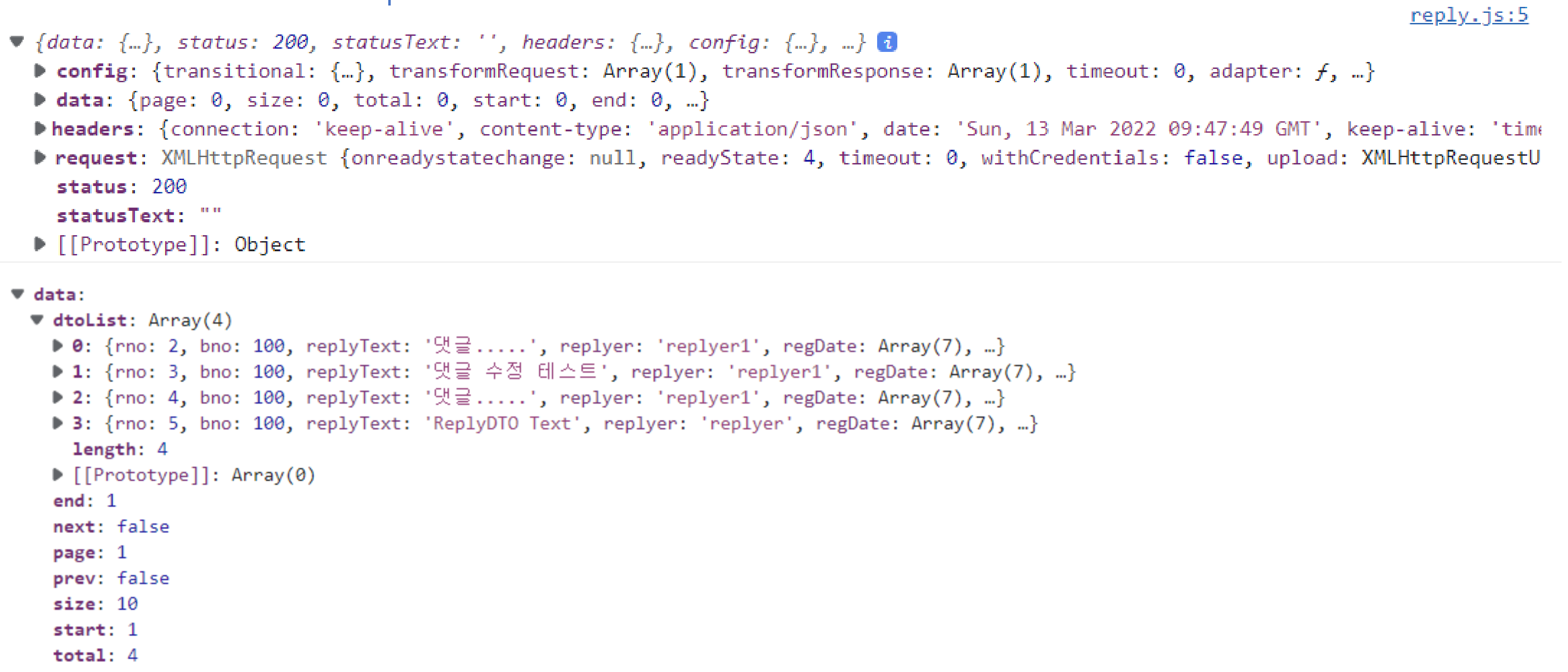
```
<script layout:fragment="script" th:inline="javascript">

    const bno = [[${dto.bno}]]

    get1(bno)

</script>
```



Board Read

localhost:8080/board/read?bno=100&page=1&size=10

Bno	100
Title	update..title 100
Content	update content 100
Writer	user0
RegDate	2022-03-12 00:27:46
ModDate	2022-03-12 00:34:31



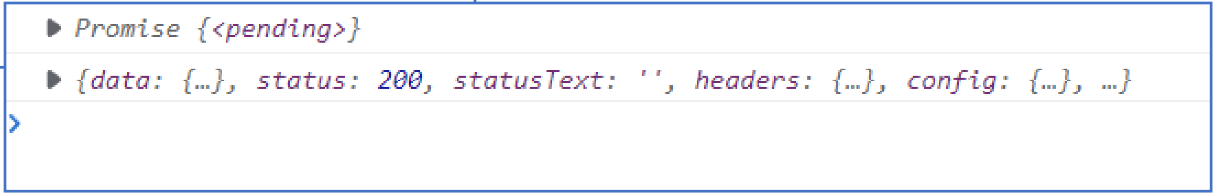
```
                                                                                                   reply.js:5
▼ {data: {…}, status: 200, statusText: '', headers: {…}, config: {…}, …} ℹ
  ▶ config: {transitional: {…}, transformRequest: Array(1), transformResponse: Array(1), timeout: 0, adapter: ƒ, …}
  ▶ data: {page: 0, size: 0, total: 0, start: 0, end: 0, …}
  ▶ headers: {connection: 'keep-alive', content-type: 'application/json', date: 'Sun, 13 Mar 2022 09:47:49 GMT', keep-alive: 'time
  ▶ request: XMLHttpRequest {onreadystatechange: null, readyState: 4, timeout: 0, withCredentials: false, upload: XMLHttpRequestU
    status: 200
    statusText: ""
  ▶ [[Prototype]]: Object

▼ data:
  ▼ dtoList: Array(4)
    ▶ 0: {rno: 2, bno: 100, replyText: '댓글.....', replyer: 'replyer1', regDate: Array(7), …}
    ▶ 1: {rno: 3, bno: 100, replyText: '댓글 수정 테스트', replyer: 'replyer1', regDate: Array(7), …}
    ▶ 2: {rno: 4, bno: 100, replyText: '댓글.....', replyer: 'replyer1', regDate: Array(7), …}
    ▶ 3: {rno: 5, bno: 100, replyText: 'ReplyDTO Text', replyer: 'replyer', regDate: Array(7), …}
      length: 4
    ▶ [[Prototype]]: Array(0)
    end: 1
    next: false
    page: 1
    prev: false
    size: 10
    start: 1
    total: 4
```

비동기 함수의 반환

- Axios의 경우 비동기 함수의 반환은 항상 Promise

```
async function get1(bno) {

    const result = await axios.get(`/replies/list/${bno}`)


    return result.data
}
```

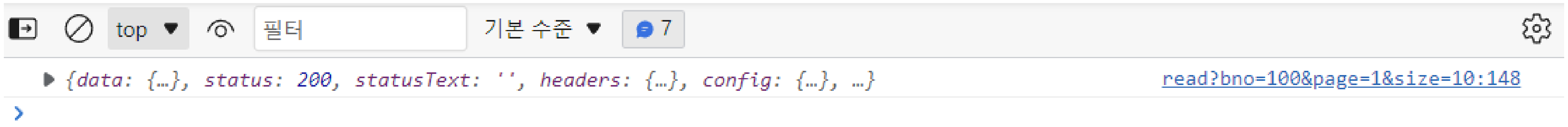
```
<script layout:fragment="script" th:inline="javascript">

    const bno = [[${dto.bno}]]

    console.log(get1(bno))

</script>
```



```
▶ Promise {<pending>}
▶ {data: {…}, status: 200, statusText: '', headers: {…}, config: {…}, …}
>
```

```
async function get1(bno) {

    const result = await axios.get(`/replies/list/${bno}`)

    //console.log(result)

    return result;
}
```

```
<script layout:fragment="script" th:inline="javascript">

    const bno = [[${dto.bno}]]

    get1(bno).then(data => {
        console.log(data)
    }).catch(e => {
        console.error(e)
    })

</script>
```

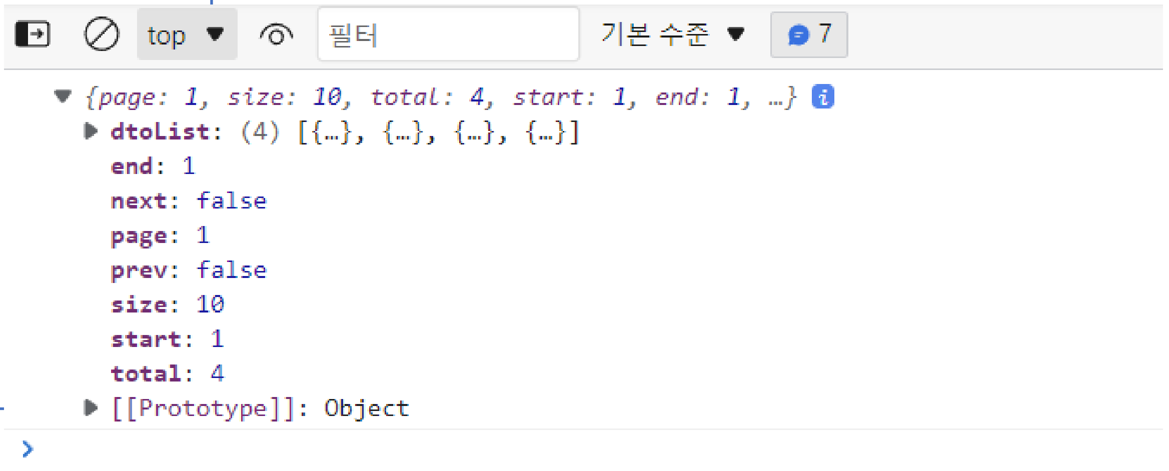


```
▶ {data: {…}, status: 200, statusText: '', headers: {…}, config: {…}, …}        read?bno=100&page=1&size=10:148
>
```

댓글 처리 JavaScript – 댓글 목록 처리

```
async function getList({bno, page, size, goLast}){

    const result = await axios.get(`/replies/list/${bno}`, {params: {page, size}})

    return result.data
}
```

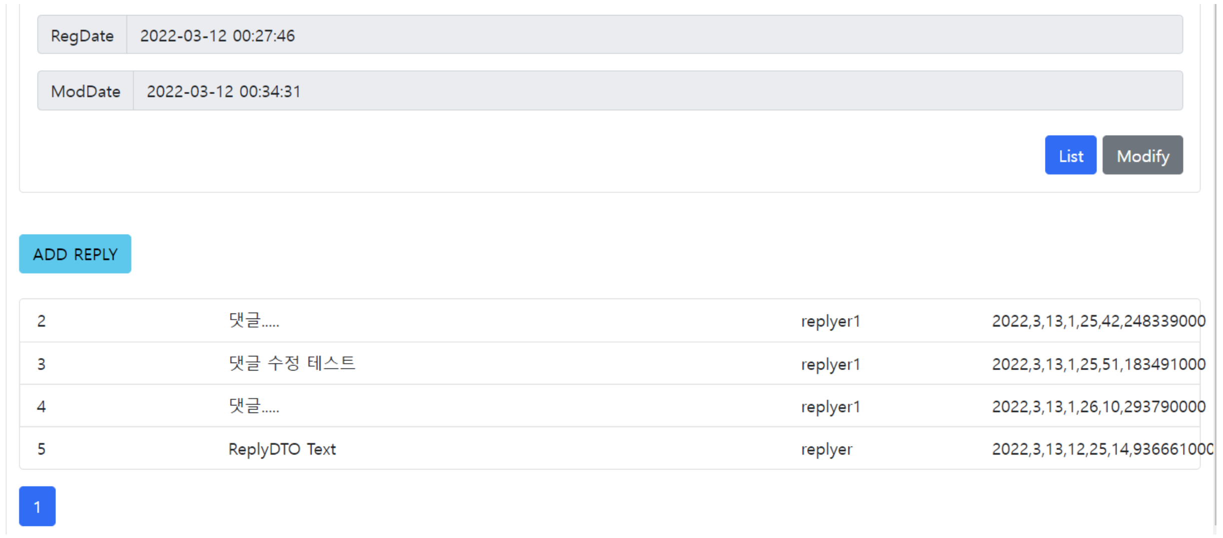
```
<script layout:fragment="script" th:inline="javascript">

    const bno = [[${dto.bno}]]

    function printReplies(page,size,goLast){

        getList({bno, page,size, goLast}).then(
            data => {console.log(data)}
        ).catch(e => {
            console.error(e)
        })
    }

    printReplies(1,10)

</script>
```



```
▼ {page: 1, size: 10, total: 4, start: 1, end: 1, …}
  ▶ dtoList: (4) [{…}, {…}, {…}, {…}]
    end: 1
    next: false
    page: 1
    prev: false
    size: 10
    start: 1
    total: 4
  ▶ [[Prototype]]: Object
>
```

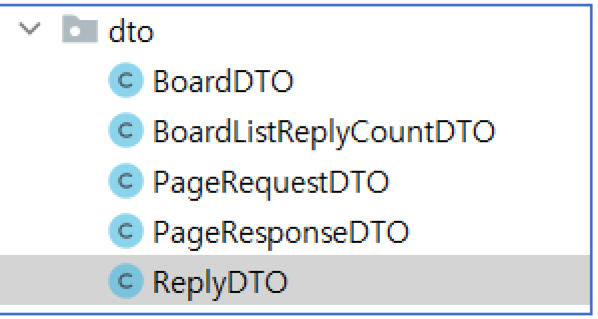
```
const bno = [[${dto.bno}]]

const replyList = document.querySelector('.replyList') //댓글 목록 DOM
const replyPaging = document.querySelector('.replyPaging') //페이지 목록 DOM

function printList(dtoList){ //댓글 목록 출력
    let str = '';

    if(dtoList && dtoList.length > 0){

        for (const dto of dtoList) {

            str += `<li class="list-group-item d-flex replyItem">
                  <span class="col-2">${dto.rno}</span>
                  <span class="col-6" data-rno="${dto.rno}">${dto.replyText}</span>
                  <span class="col-2">${dto.replyer}</span>
                  <span class="col-2">${dto.regDate} </span>
                </li>`
        }
    }
    replyList.innerHTML = str
}

function printPages(data){ //페이지 목록 출력

    //pagination
    let pageStr = '';

    if(data.prev) {
        pageStr +=`<li class="page-item"><a class="page-link" data-page="${data.start-
1}">PREV</a></li>`
    }

    for(let i = data.start; i <= data.end; i++){
        pageStr +=`<li class="page-item ${i == data.page?"active":""} "><a class="page-link" data-
page="${i}">${i}</a></li>`
    }

    if(data.next) {
        pageStr +=`<li class="page-item"><a class="page-link" data-page="${data.end
+1}">NEXT</a></li>`
    }
    replyPaging.innerHTML = pageStr
}
```

```
function printReplies(page,size,goLast){

    getList({bno, page,size, goLast}).then(
        data => {
            printList(data.dtoList) //목록 처리
            printPages(data) //페이지 처리
        }
    ).catch(e => {
        console.error(e)
    })
}

printReplies(1,10)
```



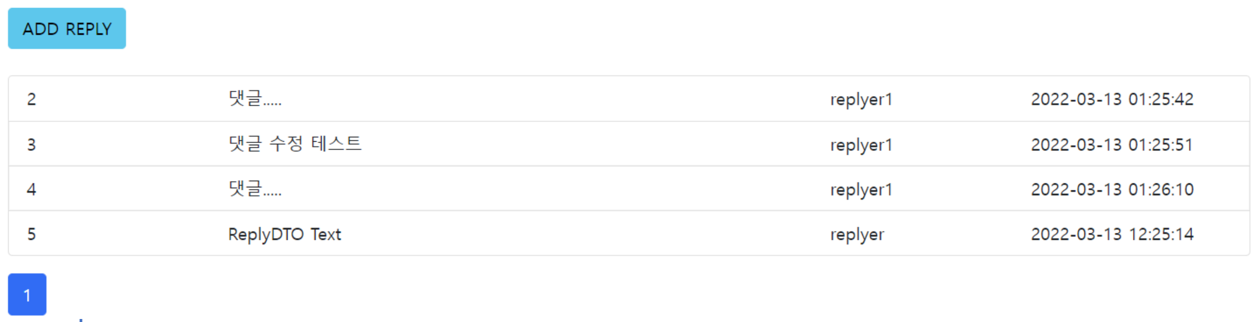
2	댓글.....	replyer1	2022,3,13,1,25,42,248339000
3	댓글 수정 테스트	replyer1	2022,3,13,1,25,51,183491000
4	댓글.....	replyer1	2022,3,13,1,26,10,293790000
5	ReplyDTO Text	replyer	2022,3,13,12,25,14,93666100C

@JsonFormat, @JsonIgnore



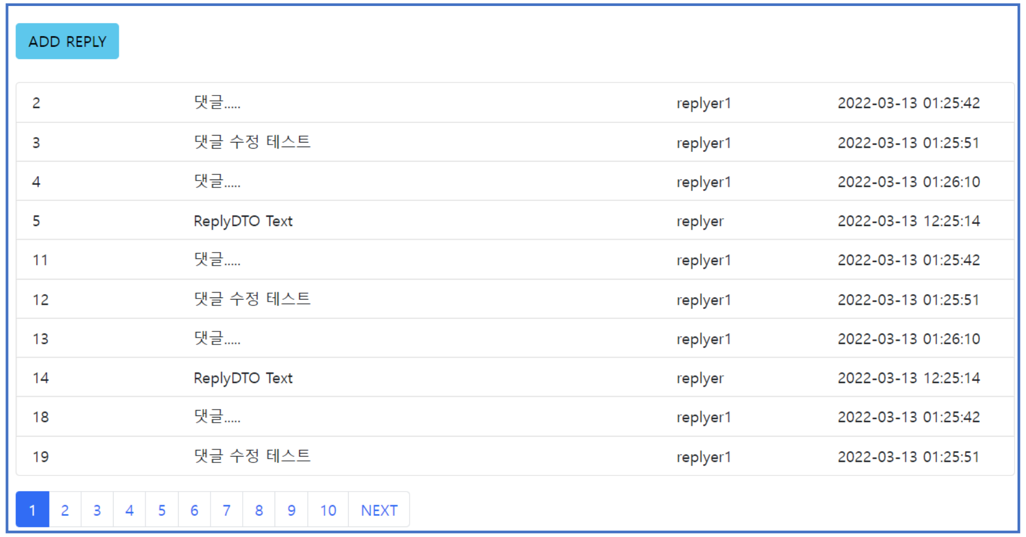
- dto
  - BoardDTO
  - BoardListReplyCountDTO
  - PageRequestDTO
  - PageResponseDTO
  - ReplyDTO

```java
package org.zerock.b01.dto;

import com.fasterxml.jackson.annotation.JsonFormat;
import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
import java.time.LocalDateTime;



@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class ReplyDTO {

    private Long rno;

    @NotNull
    private Long bno;

    @NotEmpty
    private String replyText;

    @NotEmpty
    private String replyer;

    @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss")
    private LocalDateTime regDate;

    @JsonIgnore
    private LocalDateTime modDate;

}
```



2	댓글.....	replyer1	2022-03-13 01:25:42
3	댓글 수정 테스트	replyer1	2022-03-13 01:25:51
4	댓글.....	replyer1	2022-03-13 01:26:10
5	ReplyDTO Text	replyer	2022-03-13 12:25:14

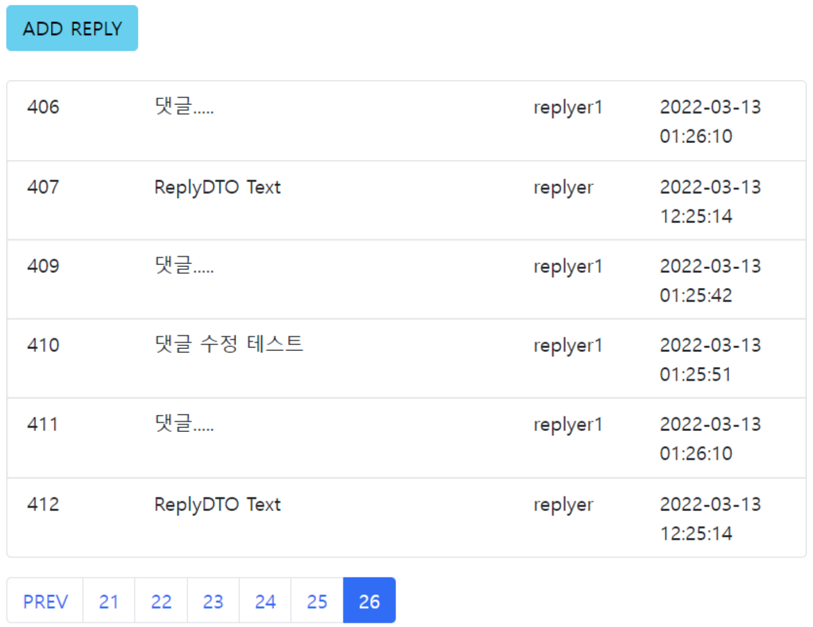
마지막 페이지로 이동


ADD REPLY

2	댓글.....	replyer1	2022-03-13 01:25:42
3	댓글 수정 테스트	replyer1	2022-03-13 01:25:51
4	댓글.....	replyer1	2022-03-13 01:26:10
5	ReplyDTO Text	replyer	2022-03-13 12:25:14
11	댓글.....	replyer1	2022-03-13 01:25:42
12	댓글 수정 테스트	replyer1	2022-03-13 01:25:51
13	댓글.....	replyer1	2022-03-13 01:26:10
14	ReplyDTO Text	replyer	2022-03-13 12:25:14
18	댓글.....	replyer1	2022-03-13 01:25:42
19	댓글 수정 테스트	replyer1	2022-03-13 01:25:51

1 2 3 4 5 6 7 8 9 10 NEXT

```
async function getList({bno, page, size, goLast}){

    const result = await axios.get(`/replies/list/${bno}`, {params: {page, size}})

    if(goLast){
        const total = result.data.total
        const lastPage = parseInt(Math.ceil(total/size))

        return getList({bno:bno, page:lastPage, size:size})

    }

    return result.data
}
```


ADD REPLY

406	댓글.....	replyer1	2022-03-13 01:26:10
407	ReplyDTO Text	replyer	2022-03-13 12:25:14
409	댓글.....	replyer1	2022-03-13 01:25:42
410	댓글 수정 테스트	replyer1	2022-03-13 01:25:51
411	댓글.....	replyer1	2022-03-13 01:26:10
412	ReplyDTO Text	replyer	2022-03-13 12:25:14

PREV 21 22 23 24 25 26

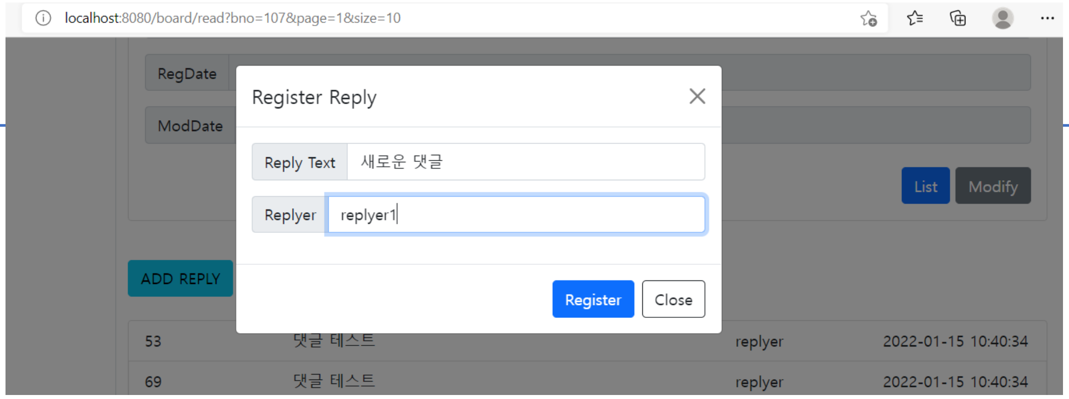
```
printReplies(1,10,true)
```

댓글의 등록

js
reply.js

```
async function addReply(replyObj) {
    const response = await axios.post(`/replies/`,replyObj)
    return response.data
}
```

```
<div class="modal registerModal" tabindex="-1">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h5 class="modal-title">Register Reply</h5>
                <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
            </div>
            <div class="modal-body">
                <div class="input-group mb-3">
                    <span class="input-group-text">Reply Text</span>
                    <input type="text" class="form-control replyText" >
                </div>
                <div class="input-group mb-3">
                    <span class="input-group-text">Replyer</span>
                    <input type="text" class="form-control replyer" >
                </div>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-primary registerBtn">Register</button>
                <button type="button" class="btn btn-outline-dark closeRegisterBtn" >Close</button>
            </div>
        </div>
    </div>
</div>
```



localhost:8080/board/read?bno=107&page=1&size=10

RegDate
ModDate

Register Reply

Reply Text 새로운 댓글
Replyer replyer1

Register Close

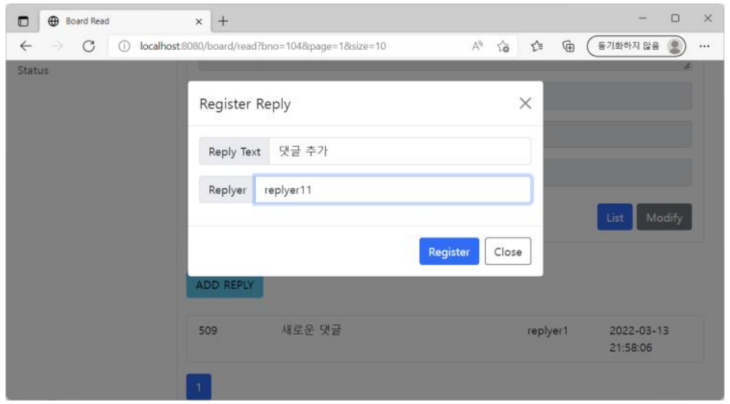
List Modify

ADD REPLY

53	댓글 테스트	replyer	2022-01-15 10:40:34
69	댓글 테스트	replyer	2022-01-15 10:40:34

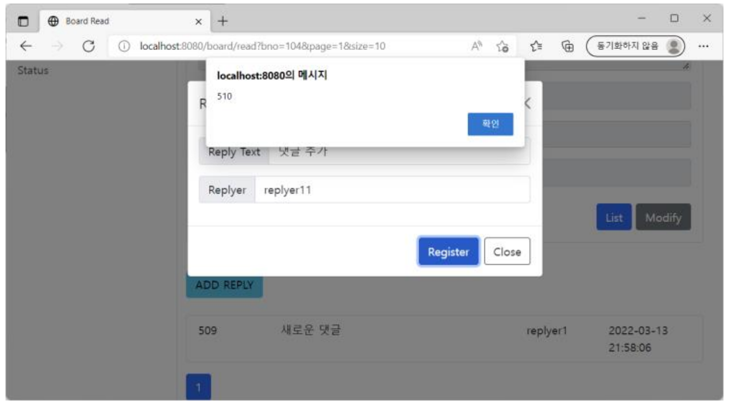
```
//댓글 등록 모달
const registerModal = new bootstrap.Modal(document.querySelector(".registerModal"))
//registerModel
const registerBtn = document.querySelector(".registerBtn")
const replyText = document.querySelector(".replyText")
const replyer = document.querySelector(".replyer")
const closeRegisterBtn = document.querySelector(".closeRegisterBtn")

document.querySelector(".addReplyBtn").addEventListener("click", function (e){
   registerModal.show()
},false)

closeRegisterBtn.addEventListener("click", function (e){
   registerModal.hide()
},false)
```

```
registerBtn.addEventListener("click", function(e){
    const replyObj = {
        bno:bno,
        replyText:replyText.value,
        replyer:replyer.value}

    addReply(replyObj).then(result => {
        alert(result.rno)
        registerModal.hide()
        replyText.value = ''
        replyer.value =''
        printReplies(1,10, true) //댓글 목록 갱신
    }).catch(e => {
        alert("Exception...")
    })
}, false)
```



Reply Text	댓글 추가
Replyer	replyer11

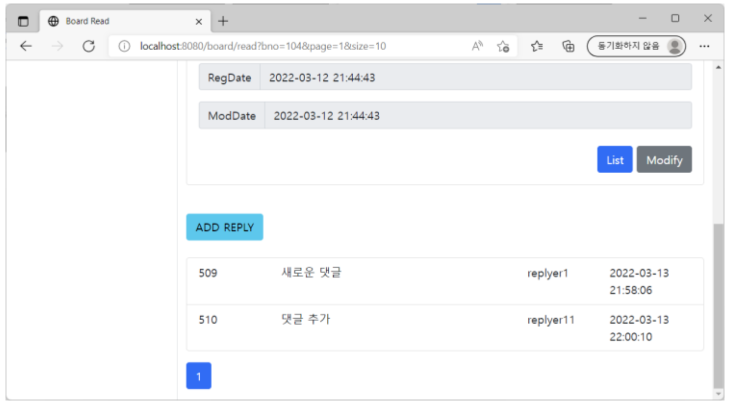
```
509    새로운 댓글    replyer1    2022-03-13
                                  21:58:06
```



localhost:8080의 메시지
510

Reply Text	댓글 추가
Replyer	replyer11

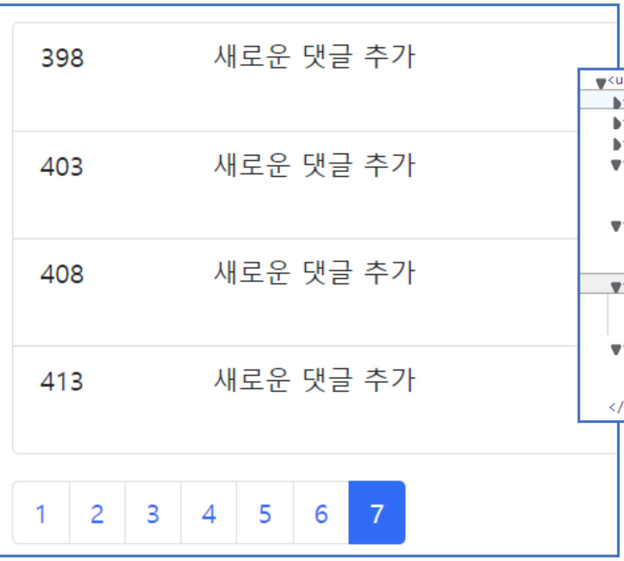
```
509    새로운 댓글    replyer1    2022-03-13
                                  21:58:06
```



RegDate	2022-03-12 21:44:43
ModDate	2022-03-12 21:44:43

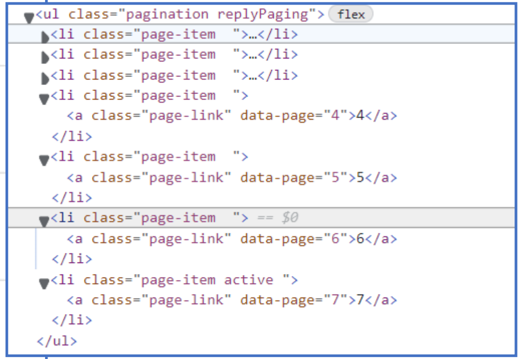
```
509    새로운 댓글    replyer1     2022-03-13
                                   21:58:06
510    댓글 추가      replyer11    2022-03-13
                                   22:00:10
```

댓글 페이지 번호 클릭



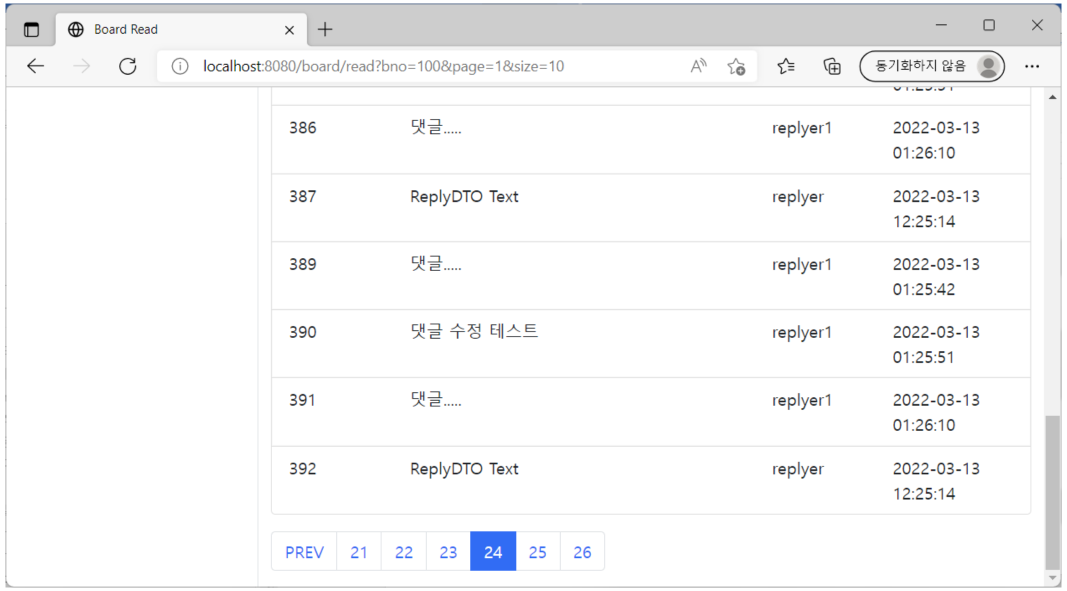
398	새로운 댓글 추가
403	새로운 댓글 추가
408	새로운 댓글 추가
413	새로운 댓글 추가

1 2 3 4 5 6 7



```
<ul class="pagination replyPaging"> flex
  <li class="page-item  ">…</li>
  <li class="page-item  ">…</li>
  <li class="page-item  ">…</li>
  <li class="page-item  ">
    <a class="page-link" data-page="4">4</a>
  </li>
  <li class="page-item  ">
    <a class="page-link" data-page="5">5</a>
  </li>
  <li class="page-item  "> == $0
    <a class="page-link" data-page="6">6</a>
  </li>
  <li class="page-item active ">
    <a class="page-link" data-page="7">7</a>
  </li>
</ul>
```

```
//댓글 페이지 클릭
let page = 1
let size = 10

replyPaging.addEventListener("click", function (e){

    e.preventDefault()
    e.stopPropagation()

    const target = e.target

    if(!target || target.tagName != 'A'){
        return
    }

    const pageNum = target.getAttribute("data-page")
    page = pageNum
    printReplies(page, size)

},false)
```



Board Read

localhost:8080/board/read?bno=100&page=1&size=10

386	댓글.....	replyer1	2022-03-13 01:26:10
387	ReplyDTO Text	replyer	2022-03-13 12:25:14
389	댓글.....	replyer1	2022-03-13 01:25:42
390	댓글 수정 테스트	replyer1	2022-03-13 01:25:51
391	댓글.....	replyer1	2022-03-13 01:26:10
392	ReplyDTO Text	replyer	2022-03-13 12:25:14

PREV 21 22 23 24 25 26

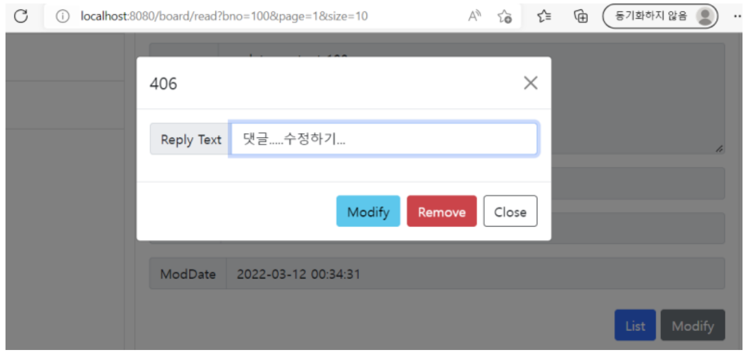
댓글 조회와 수정

```
async function getReply(rno) {
   const response = await axios.get(`/replies/${rno}`)
   return response.data
}

async function modifyReply(replyObj) {

   const response = await axios.put(`/replies/${replyObj.rno}`, replyObj)
   return response.data
}
```

```
<div class="modal modifyModal" tabindex="-1">
   <div class="modal-dialog">
      <div class="modal-content">
         <div class="modal-header">
            <h5 class="modal-title replyHeader"></h5>
            <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
         </div>
         <div class="modal-body">
            <div class="input-group mb-3">
               <span class="input-group-text">Reply Text</span>
               <input type="text" class="form-control modifyText" >
            </div>
         </div>
         <div class="modal-footer">
            <button type="button" class="btn btn-info modifyBtn">Modify</button>
            <button type="button" class="btn btn-danger removeBtn">Remove</button>
            <button type="button" class="btn btn-outline-dark closeModifyBtn">Close</button>
         </div>
      </div>
   </div>
</div> <!--modifyModal -->
```

```
//modifyModal
const modifyModal = new bootstrap.Modal(document.querySelector(".modifyModal"))

const replyHeader = document.querySelector(".replyHeader")
const modifyText = document.querySelector(".modifyText")
const modifyBtn = document.querySelector(".modifyBtn")
const removeBtn = document.querySelector(".removeBtn")
const closeModifyBtn = document.querySelector(".closeModifyBtn")


replyList.addEventListener("click", function (e){

   e.preventDefault()
   e.stopPropagation()

   const target = e.target

   if(!target || target.tagName != 'SPAN'){
      return
   }

   const rno = target.getAttribute("data-rno")

   if(!rno){
      return
   }

   getReply(rno).then(reply => { //댓글의 내용을 모달창에 채워서 보여주는

      console.log(reply)
      replyHeader.innerHTML = reply.rno
      modifyText.value = reply.replyText
      modifyModal.show()

   }).catch(e => alert('error'))

},false)
```

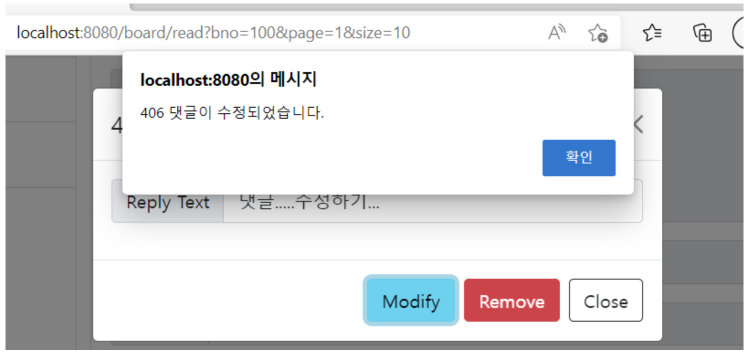
```
modifyBtn.addEventListener("click", function(e) {

    const replyObj = {
        bno:bno,
        rno:replyHeader.innerHTML,
        replyText:modifyText.value}

    modifyReply(replyObj).then(result => {
        alert(result.rno+' 댓글이 수정되었습니다.')
        replyText.value = ''
        modifyModal.hide()
        printReplies(page, size)

    }).catch(e => {
        console.log(e)
    })
},false)

closeModifyBtn.addEventListener("click", function(e){

    modifyModal.hide()

}, false)
```


localhost:8080/board/read?bno=100&page=1&size=10

406

Reply Text 댓글.....수정하기...

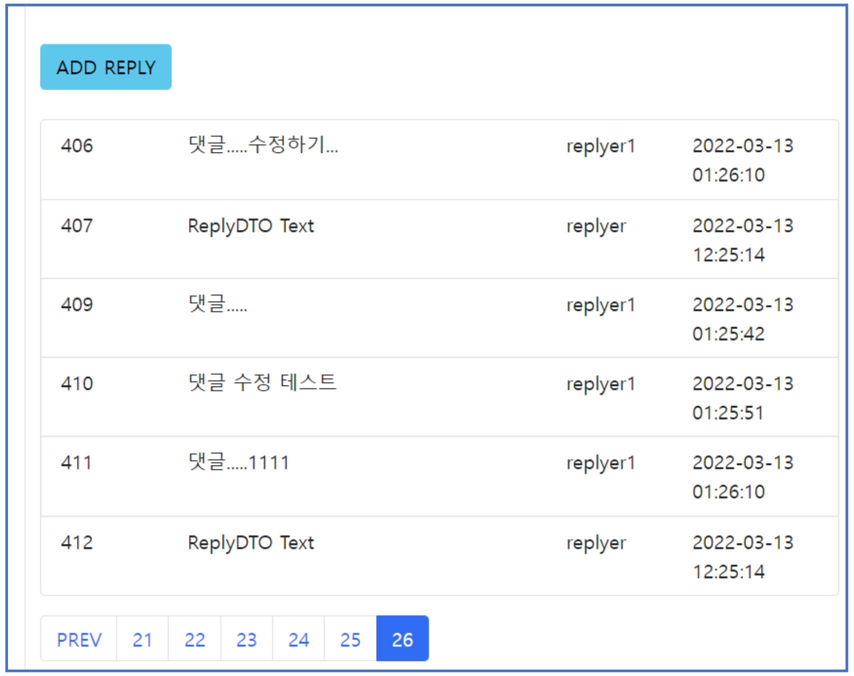
ModDate 2022-03-12 00:34:31


localhost:8080/board/read?bno=100&page=1&size=10

localhost:8080의 메시지

406 댓글이 수정되었습니다.

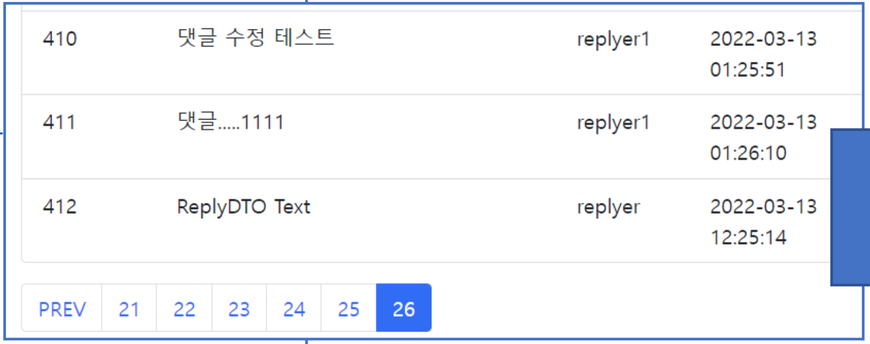
Reply Text 댓글.....수정하기...



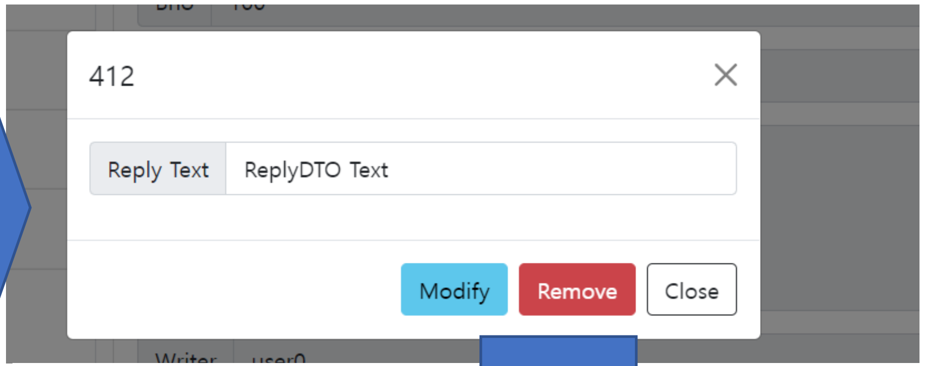
406	댓글.....수정하기...	replyer1	2022-03-13 01:26:10
407	ReplyDTO Text	replyer	2022-03-13 12:25:14
409	댓글.....	replyer1	2022-03-13 01:25:42
410	댓글 수정 테스트	replyer1	2022-03-13 01:25:51
411	댓글.....1111	replyer1	2022-03-13 01:26:10
412	ReplyDTO Text	replyer	2022-03-13 12:25:14

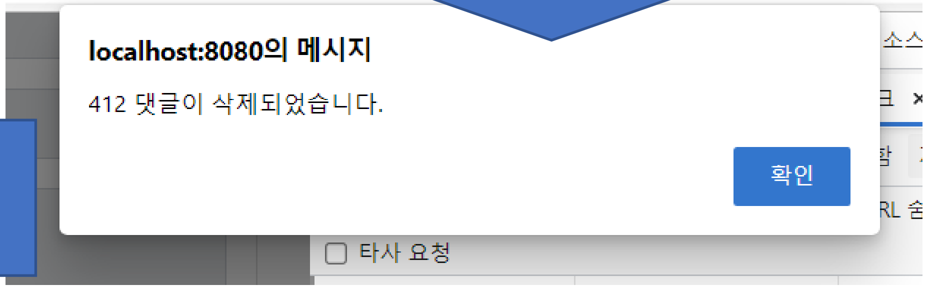
댓글의 삭제

```
removeBtn.addEventListener("click", function(e) {

  removeReply(replyHeader.innerHTML).then(result => {

    alert(result.rno +' 댓글이 삭제되었습니다.')
    replyText.value = ''
    modifyModal.hide()

    page = 1 // 이 부분이 없다면 원래 페이지로

    printReplies(page, size)

  }).catch(e => {
    console.log(e)
  })
},false)
```

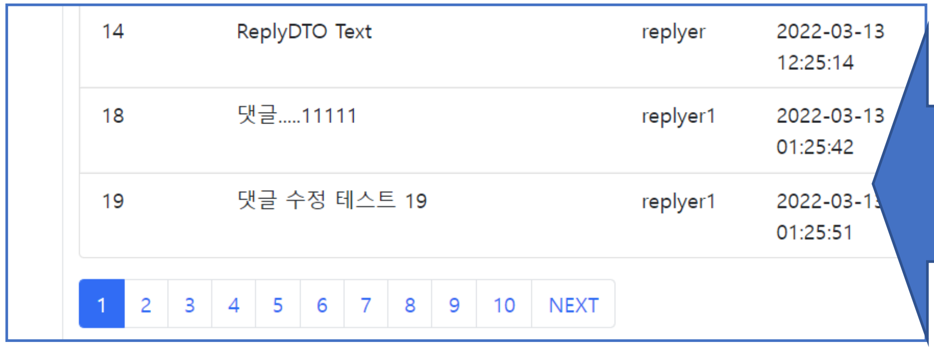


410	댓글 수정 테스트	replyer1	2022-03-13 01:25:51
411	댓글.....1111	replyer1	2022-03-13 01:26:10
412	ReplyDTO Text	replyer	2022-03-13 12:25:14




localhost:8080의 메시지

412 댓글이 삭제되었습니다.



14	ReplyDTO Text	replyer	2022-03-13 12:25:14
18	댓글.....11111	replyer1	2022-03-13 01:25:42
19	댓글 수정 테스트 19	replyer1	2022-03-1 01:25:51