

0120

동적 쿼리

→ SQL 문장이 조건에 따라 변경되서 적용되어야 하는 경우

정적 쿼리

- insert
- delete
- update
- 하나의 데이터 조회

: 위 4개의 작업들은 데이터가 변하는 것이지
수행되어야 하는 SQL 자체가 변경되지는 않는다

- 조회의 경우도 where 절의 컬럼이 변경되지는 않는다

: 이런 경우

- JPA의 기본 Repository method 생성
- 쿼리 method 를 생성
- @Query 를 이용해서 해결 가능

조건을 변경해 가면서 검색하는 경우

정적 쿼리를 사용할 수 없음

정적 쿼리를 이용한다면 여러개의 method를 만들고

조건문을 이용해서

적절한 method 를 호출하는 방식으로 처리

즉 method 를 많이 생성해야한다.

: 하나의 method 가 처리해야할 내용이 많아서

method 를 분할하는 것은 괜찮지만

그 이외의 경우 method 가 많아지면 가독성이 떨어짐

JPA 에서는 이 경우 사용할 수 있도록 Querydsl 이라는 라이브러리를 제공

많이 복잡한 쿼리를 사용해야하는 경우

nativeSQL 이나 MyBatis 를 이용하기도 한다

Build. gradle

```
def querydslDir = "$buildDir/generated/querydsl"

querydsl {
    jpa = true
    querydslSourcesDir = querydslDir
}

sourceSets {
    main.java.srcDir querydslDir
}

configurations {
    compileOnly {
        extendsFrom annotationProcessor
    }
    querydsl.extendsFrom compileClasspath
}

compileQuerydsl {
    options.annotationProcessorPath = configurations.querydsl
}
```

변경 내용을 적용

compileQuerydsl 옆에 생기는 아이콘을 눌러서 실행
각 Entity 앞에 Q가 붙는 클래스가 생성
이 클래스를 이용해서 동적인 쿼리를 생성

SearchBoardRepositoryImpl 클래스를 생성

```
package com.singsiuk.board.repository;

public interface SearchBoardRepository {
}
```

어제 까지의 내용

SearchboardRepository에 method 호출

```
package com.singsiuk.board.repository;

import com.singsiuk.board.entity.Board;

public interface SearchBoardRepository {
    public Board search();
}
```

SearchBoardRepositoryImpl 에 method 호출

```
package com.singsiuk.board.repository;

import com.querydsl.jpa.JPQLQuery;
import com.singsiuk.board.entity.Board;
import com.singsiuk.board.entity.QBoard;
import lombok.extern.log4j.Log4j2;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.jpa.repository.support.QuerydslRepositorySupport;

import java.util.List;

@Log4j2
public class SearchBoardRepositoryImpl extends QuerydslRepositorySupport implements SearchBoardRepository {
    public SearchBoardRepositoryImpl() {
        super(Board.class);
    }

    @Override
    public Board search() {
        log.info("search method call");
        //board Entity에 쿼리를 수행하기 위한 객체 생성
        QBoard board = QBoard.board;
        JPQLQuery<Board> jpqlQuery = from(board);

        jpqlQuery.select(board).where(board.bon.eq(1L));
        List<Board> result = jpqlQuery.fetch();
        return null;
    }
}
```

BoardRepository Interface 에 SearchBoardRepository 를 Extends

```
package com.singsiuk.board.repository;

import com.singsiuk.board.entity.Board;
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.springframework.data.repository.query.Param;

import java.util.List;

public interface BoardRepository extends JpaRepository<Board, Long>, SearchBoardRepository {
```

Test // RepositoryTest

```
@Test
public void testQuerydsl() {
    boardRepository.search();
}
```

Board 와 Reply Entity 의 Join

Board Entity 에는 Reply Entity 방향으로 관계가 없다
→ On 절이 있어야 한다

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

```
JPQLQuery<Board> jpqlQuery = from(board);
```

```
jpqlQuery.leftjoin(reply).on(reply.board.eq(board));
```

이런 형태로 여러개의 Entity 를 조인할 수 있다 .

SearchBoardRepositoryImp

```
@Log4j2
public class SearchBoardRepositoryImpl extends QuerydslRepositorySupport implements SearchBoardRepository {
    public SearchBoardRepositoryImpl(){
        super(Board.class);
    }

    @Override
    public Board search() {
        log.info("search method call");
        //board Entity에 쿼리를 수행하기 위한 객체 생성
        QBoard board = QBoard.board;
        QMember member = QMember.member;
        QReply reply = QReply.reply;

        JPQLQuery<Board> jpqlQuery = from(board);

        // Member Entity 와 Join
        // Board 의 writer 가 Member 를 참조
        jpqlQuery.leftJoin(member).on(board.writer.eq(member));
        // Reply Entity 와 Join

        // reply 가 board 거 가지고 있다
        jpqlQuery.leftJoin(reply).on(reply.board.eq(board));

        //필요한 항목 추출
        jpqlQuery.select(board, member.email, reply.count())
            .groupBy(board);
        //댓글은 board 에 groupby 해서

        List<Board> result =jpqlQuery.fetch();
        System.out.println(result);
        return null;
    }
}
```

TEST

```
@Test
    public void testQuerydsl(){
        boardRepository.search();
    }
}
```

```
reply reply2_
on (
    reply2_.board_bon=board0_.bon
)
group by
    board0_.bon
[[Board(bon=1, title=제목 : 인생1, content=내용 : 날개다람쥐1), user1@gmail.com, 2], [Board(bon=2, title=제목 : 인생2, content=내용 : 날개다람쥐2),
2022-01-20 10:44:48.832 INFO 8429 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for
2022-01-20 10:44:48.837 INFO 8429 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated.
2022-01-20 10:44:48.879 INFO 8429 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
Deprecated Gradle features were used in this build, making it incompatible with Gradle 8.0.
You can use '--warning-mode all' to show the individual deprecation warnings and determine if they come from your own scripts or plugins.
See https://docs.gradle.org/7.3.2/userguide/command\_line\_interface.html#sec:command\_line\_warnings
BUILD SUCCESSFUL in 14s
```

Tuple 객체

- 지금까지의 방법은 Entity 객체를 이용해서 데이터를 가져왔다.
 - > 이를 Tuple을 이용해서 가져올 수 있다.

: Entity 는 기존의 클래스 처럼 미리 만들어 두어야 한다

Tuple

: Object 객체처럼 어떤 종류의 결과도 모두 저장이 가능하다.

SearchBoardRepositoryImpl

```
@Log4j2
public class SearchBoardRepositoryImpl extends QuerydslRepositorySupport implements SearchBoardRepository {
    public SearchBoardRepositoryImpl(){
        super(Board.class);
    }

    @Override
    public Board search() {
        log.info("search method call");
        //board Entity에 쿼리를 수행하기 위한 객체 생성
        QBoard board = QBoard.board;
        QMember member = QMember.member;
        QReply reply = QReply.reply;

        JPQLQuery<Board> jpqlQuery = from(board);

        // Member Entity 와 Join
        // Board 의 writer 가 Member 를 참조
        jpqlQuery.leftJoin(member).on(board.writer.eq(member));
        // Reply Entity 와 Join

        // reply 가 board 거 가지고 있다
        jpqlQuery.leftJoin(reply).on(reply.board.eq(board));

        // 필요한 항목 추출
        // jpqlQuery.select(board, member.email, reply.count())
        // .groupBy(board);
        // 댓글은 board 에 groupby 해서
        // List<Board> result =jpqlQuery.fetch();

        JPQLQuery<Tuple> tuple =
            jpqlQuery.select(board, member.email,reply.count()).groupBy(board);
        List<Tuple> result = tuple.fetch();

        System.out.println(result);
        return null;
    }
}
```

Test

```
@Test
public void testQuerydsl(){
    boardRepository.search();
}
```

```

    reply reply2_
        on (
            reply2_.board_bon=board0_.bon
        )
    group by
        board0_.bon
[[Board(bon=1, title=제목 : 인생1, content=내용 : 날개다람쥐1), user1@gmail.com, 2], [Board(bon=2, title=제목 : 인생2, content=내용 : 날개다람쥐2)
2022-01-20 10:58:32.506 INFO 9638 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory f
2022-01-20 10:58:32.517 INFO 9638 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory f
```

기존의 Board Repository를 사용하지 않고 SearchBoardRepository를 추가해서 구현한 이유

Answer :
최대한 현재 구현되어 있는 내용을 수정하지 않고 동작하기 위해서

SearchBoardRepository Interface 에 검색기능을 위한 method 를 생성

```
package com.singsiuk.board.repository;

import com.singsiuk.board.entity.Board;
import org.springframework.data.domain.Page;

import java.awt.print.Pageable;

public interface SearchBoardRepository {
    //테스트용 method
    public Board search();

    // 목록보기를 위한 method
    Page<Object[]> searchPage(String type, String keyword, Pageable pageable);
}
```


SearchBoardRepositoryImpl

```
@Override
public Page<Object[]> searchPage(String type, String keyword, Pageable pageable) {
    //3개의 Entity Join
    QBoard board = QBoard.board;
    QMember member = QMember.member;
    QReply reply = QReply.reply;

    JPQLQuery<Board> jpqlQuery = from(board);
    //Member Entity와 join
    //Board의 writer가 Member를 참조
    jpqlQuery.leftJoin(member).on(board.writer.eq(member));
    //Reply Entity와 join
    jpqlQuery.leftJoin(reply).on(reply.board.eq(board));

    //Board 와 Member.email 그리고 Reply의 count를 가져오는
    //쿼리를 작성
    JPQLQuery<Tuple> tuple = jpqlQuery.select(board, member, reply.count());
    //동적 쿼리 생성
    BooleanBuilder booleanBuilder = new BooleanBuilder();
    //조건 생성
    BooleanExpression expression = board.bon.gt(0L);
    //조건 추가
    booleanBuilder.and(expression);

    //t: title로 검색
    //c: content로 검색
    //w: writer로 검색
    //tc: or검색
    //tcw; ``
    if(type != null){
        String [] typear = type.split("");
        BooleanBuilder conditionBuilder = new BooleanBuilder();
        for(String t : typear){
            switch (t){
                case "t" :
                    conditionBuilder.or(board.title.contains(keyword));
                    break;
                case "c" :
                    conditionBuilder.or(board.content.contains(keyword));
                    break;
                case "w" :
                    conditionBuilder.or(member.email.contains(keyword));
                    break;
            }
        }
        booleanBuilder.and(conditionBuilder);
    }
    //조건 검색 추가
    tuple.where(booleanBuilder);
    //정렬 추가
    Sort sort = pageable.getSort();

    sort.stream().forEach(order -> {
        //정렬 방향 찾아오기
        Order direction = order.isAscending()? Order.ASC: Order.DESC;
        String prop = order.getProperty();

        PathBuilder orderByExpression = new PathBuilder(Board.class, "board");
        tuple.orderBy(new OrderSpecifier(direction, orderByExpression.get(prop)));
    });
    //위에꺼랑 똑같은거 for(임시변수 : 컬렉션){}
    //위에 처럼 하는 이유는 병렬처리 같은 것들을 할 수 있기에 속도가 빠르다.

    //그룹화
    tuple.groupBy(board);

    //페이지 처리
    tuple.offset(pageable.getOffset());
    tuple.limit(pageable.getPageSize());

    //데이터 가져오기
    List<Tuple> result = tuple.fetch();
    //데이터 리턴
    return new PageImpl<Object[]> (
        result.stream().map(t -> t.toArray()).collect(Collectors.toList()),pageable,tuple.fetchCount()
    );
}
```

Test

```
@Test
public void testQuerydsl(){
//    boardRepository.search();
    Pageable pageable = PageRequest.of(0,10,Sort.by("bon").descending()
        .and(Sort.by("title").ascending()));

    Page<Object []> result =
        boardRepository.searchPage("t","1",pageable);
    System.out.println(result);
}
```

BoardServiceImpl

```
@Override
public PageResultDTO<BoardDTO, Object[]> getList(PageRequestDTO dto) {
    // Repository 메서드를 호출해서 결과 가져오기
    // log.info(pageRequestDTO);
    // Page<Object []> result = boardRepository.getBoardWithReplyCount(
    //     pageRequestDTO.getPageable(Sort.by("bon").descending())
    // );

    Page<Object []> result = boardRepository.searchPage(
        dto.getType(), dto.getKeyword(),
        dto.getPageable(Sort.by("bon").descending()));

    Function<Object[] ,BoardDTO> fn =
        (en->entityToDTO((Board)en[0], (Member)en[1], (Long)en[2]));

    return new PageResultDTO<>(result, fn);
}
```

Start Bootstrap

Toggle Menu

Home Link Dropdown ▾

Dashboard

Shortcuts

Overview

Events

Profile

Status

Board List Page REGISTER

제목 ▾ 지나 Search Clear

TEST	Title	Writer	Regdate
109	지나간것으 ----- [0]	QWER@ASDF.cim	2022/01/19

1

대화 상자 : 삽입이나 삭제 후 목록보기로 온 경우 작업내용을 출력

댓글 처리 방식

일반적으로 댓글은 작성 시

전체 화면을 재출력하지 않고 바로 출력 되는 경우가 많다 .

ajax 로 처리하기 위해서는

Server 에서 데이터를 리턴해주는
형태로 처리 해주어야 한다.

→ Spring 에서는

String 과 JSON 을 Return 하는

RestController를 제공

→ 이 경우 GET, POST , PUT , DELETE 모든 방식 지원

→ ajax 는

1. 자바스크립트 코드로 직접 작성

2. 외부 라이브러리를 이용해서 작성

Reply entity 의 연관관계 확인

```
package com.singsiuk.board.entity;

import lombok.*;
import javax.persistence.*;

@Entity
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Getter
@ToString(exclude = "board")
public class Reply extends BaseEntity{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long rno;
    private String text;
    private String replyer;

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

댓글 출력 작업

ReplyRepository Interface 에서 댓글에 해당하는 모든 데이터를 들고오는 method 생성

```
package com.singsiuk.board.repository;

import com.singsiuk.board.entity.Board;
import com.singsiuk.board.entity.Reply;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;

import java.util.List;

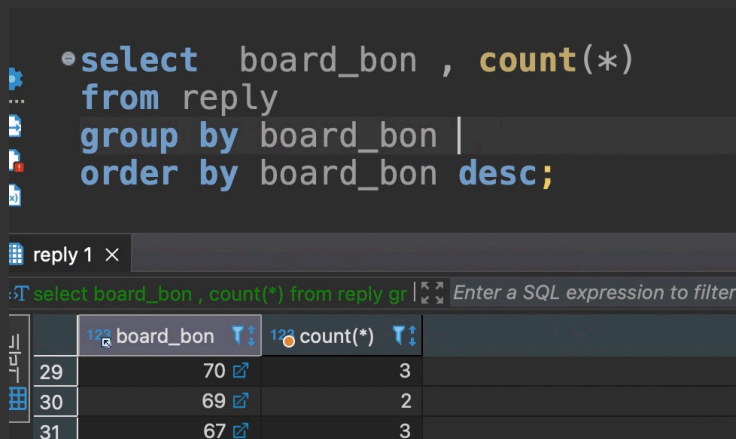
public interface ReplyRepository extends JpaRepository<Reply, Long> {

    // 삽입 삭제 갱신에는 Modifying을 써줘야한다.
    @Modifying
    // 글번호를 가지고 댓글을 삭제하는 method
    @Query("delete from Reply r where r.board.bon = :bon")
    public void deleteByBon(Long bon);

    // 게시글에 해당하는 모든 댓글을 가져오는 method
    // Board에서 들고오고 Rno 를 사용해서 정렬할 것이다 .
    public List<Reply> getRepliesByBoardOrderByRno(Board board);
}
```

TEST

댓글이 있는 글 .없는 글 찾기



The screenshot shows a SQL query being executed in a database client. The query is: `select board_bon , count(*) from reply group by board_bon order by board_bon desc;`. The results are displayed in a table with two columns: `board_bon` and `count(*)`. The data shows three board numbers: 29 with 3 replies, 30 with 2 replies, and 31 with 3 replies.

board_bon	count(*)
29	3
30	2
31	3

댓글 읽는 게시글 : 70

댓글 없는 게시글 : 68

RepositoryTest

— 댓글이 없는 경우

```
@Test
public void testReplyList(){
    List<Reply> list =
        replyRepository.getRepliesByBoardOrderByRno(
            Board.builder().bon(68L).build());
    System.out.println(list);
}
```

```
        reply reply0_
    where
        reply0_.board_bon=?
    order by
        reply0_.rno asc
```

```
[]
```

```
2022-01-20 13:02:17.823 INFO 20078 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean: 
```

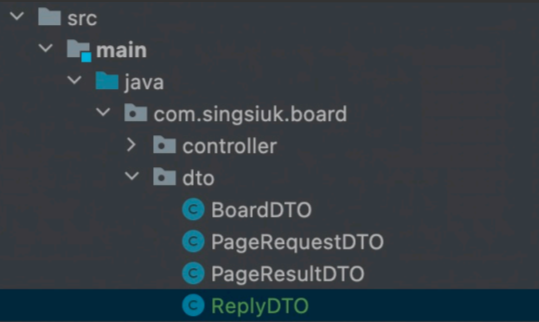
— 댓글이 있는 경우

```
@Test
public void testReplyList2(){
    List<Reply> list =
        replyRepository.getRepliesByBoardOrderByRno(
            Board.builder().bon(70L).build());
    System.out.println(list);
}
```

```
        reply0_.text as text5_1_
    from
        reply reply0_
    where
        reply0_.board_bon=?
    order by
        reply0_.rno asc
```

```
[Reply(rno=9, text=*9, replier=Customer), Reply(rno=13, text=*13, replier=Customer), Reply(rno=103, text=*103, replier=Customer)]
```

Repository 이외에 ReplyDTO 파일 생성



```
package com.singsiuk.board.dto;

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

import java.time.LocalDateTime;

@Builder
@NoArgsConstructor
@AllArgsConstructor
@Data
public class ReplyDTO {
    private Long rno;
    private String text;
    private String replyer;
    private Long bon;
    private LocalDateTime regDate;
    private LocalDateTime modDate;
}
```

—— ReplyService Interface 생성 하고 필요한 method 선언

필요한 method

- 댓글 삽입
- 댓글 삭제
- 댓글 수정
- 댓글 목록
- DTO 와 Entity 사이의 변환을 위한 method

```
package com.singsiuk.board.service;

import com.singsiuk.board.dto.ReplyDTO;
import com.singsiuk.board.entity.Board;
import com.singsiuk.board.entity.Reply;

import java.util.List;

public interface ReplyService {
    // 데이터 삽입을 위한 method
    // 삽입은 항상 기본키를 return 하도록 생성
    public Long register(ReplyDTO replyDTO);

    // 데이터 수정을 위한 method
    public void modifu(ReplyDTO replyDTO);

    // 데이터 삭제를 위한 method
    public void remove(Long rno);

    // 댓글 목록을 가져오기
    // 글 번호가 필요
    public List<ReplyDTO> getList(Long bon);

    // ReplyDTO를 reply Entity로 변환해주는 method
    default Reply dtoEntity(ReplyDTO replyDTO){
        Board board = Board.builder().bon(replyDTO.getBon()).build();

        Reply reply = Reply.builder()
            .rno(replyDTO.getRno())
            .text(replyDTO.getText())
            .replyer(replyDTO.getReplyer())
            .board(board)
            .build();
        return reply;
    }
    // Reply Entity 를 ReplyDto로 변환해주는 method
    default ReplyDTO entityToDTO(Reply reply){
        ReplyDTO replyDTO = ReplyDTO.builder()
            .rno(reply.getRno())
            .text(reply.getText())
            .replyer(reply.getReplyer())
            .regDate(reply.getRegDate())
            .modDate(reply.getModDate())
            .build();

        return replyDTO;
    }
}
```


ReplyServiceImpl 클래스 생성

method 구현

```
package com.singsiuk.board.service;

import com.singsiuk.board.dto.ReplyDTO;
import com.singsiuk.board.entity.Board;
import com.singsiuk.board.entity.Reply;
import com.singsiuk.board.repository.ReplyRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.stream.Collectors;

@Service
@RequiredArgsConstructor
public class ReplyServiceImpl implements ReplyService{

    private final ReplyRepository replyRepository;

    @Override
    public Long register(ReplyDTO replyDTO) {
        Reply reply = dtoEntity(replyDTO);
        replyRepository.save(reply);
        return reply.getRno();
    }

    @Override
    public void modify(ReplyDTO replyDTO) {
        Reply reply = dtoEntity(replyDTO);
        replyRepository.save(reply);
    }

    @Override
    public void remove(Long rno) {
        replyRepository.deleteById(rno);
    }

    @Override
    public List<ReplyDTO> getList(Long bon) {
        List<Reply> result = replyRepository.getRepliesByBoardOrderByRno(
            Board.builder().bon(bon).build());

        return result.stream().map(reply -> entityToDTO(reply)).collect(Collectors.toList());
    }
}
```

TEST

ServiceTest

```
@Autowired
private ReplyService replyService;

@Test
public void testGetReplies(){
    List<ReplyDTO> list =
        replyService.getList(70L);
    System.out.println(list);
}

}
```

```
from
  reply reply0_
where
  reply0_.board_bon=?
order by
  reply0_.rno asc
[ReplyDTO(rno=9, text=*9, replier=Customer, bon=null, regDate=2022-01-18T16:00:02.005004, modDate=2022-01-18T16:00:02.005004), Re
2022-01-20 14:52:55.964 INFO 4806 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFact
```

댓글을 출력

댓글에 관련된 요청을 JSON 으로 출력할 ReplyController 생성하고 댓글 목록 요청을 처리하는 method 생성

```
package com.singsiuk.board.controller;

import com.singsiuk.board.dto.ReplyDTO;
import com.singsiuk.board.service.ReplyService;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

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

    private final ReplyService replyService;

    // 댓글 목록 요청을 처리하는 Method
    @GetMapping(value="/board/{bon}", produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<ReplyDTO>> getListByBon(
        @PathVariable("bon") Long bon) {
        return new ResponseEntity<>(
            replyService.getList(bon), HttpStatus.OK);
    }
}
```

```
reply reply0_
where
    reply0_.board_bon=?
order by
    reply0_.rno asc
```

```
[ReplyDTO(rno=9, text=9, replier=Customer, bon=null, regDate=2022-01-18T16:00:02.005004, modDate=2022-01-18T16:00:02.005004),
2022-01-20 15:09:04.379 INFO 6065 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerF
2022-01-20 15:09:04.384 INFO 6065 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown ini
2022-01-20 15:09:04.406 INFO 6065 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown con
Deprecated Gradle features were used in this build, making it incompatible with Gradle 8.0.
```

Start Bootstrap

Toggle Menu

Dashboard

Shortcuts

Overview

Events

Profile

Status

Board Read Page

Bon

98

Title

제목 : 인생98

Content

내용 : 날개다람쥐98

Writer

오소리98

RegDate

2022/01/18 15:49:05

ModDate

2022/01/18 15:49:05

수정

목록

댓글 4

댓글 개수를 출력하면 댓글을 출력하는 Script Code 를 read.html 파일에 작성

```

<div >
  <div class="mt-4">
    <h5> <span class="badge badge-secondary replyCount">
댓글 [[${dto.replyCount}]]</span> </h5>
    </div>
    <div class="list-group replyList">
    </div>
  </div>

<script>
  // 클릭하면
  // JQuery 에서

  // 문서를 전부 읽고 나면
$(document).ready(function(){
  // 댓글의 개수 부분을 클릭하면
  $(".replyCount").click(function (){
    alert("Pikabu")
  });
})
</script>

```

Toggle Menu

localhost:8100/board/read?bon=98&page=1&type=&keyword=

Board Read Page

Bon

98

Title

제목 : 인생98

Content

내용 : 날개다람쥐98

Writer

오소리98

RegDate

2022/01/18 15:49:05

ModDate

2022/01/18 15:49:05

수정

목록

댓글 4

localhost:8100 내용:

Pikabu

확인

read.html 파일

댓글의 개수를 출력하고 댓글 출력 영역을 생성

```
<div class="mt-4">
    <h5> <span class="badge badge-secondary replyCount"> 댓글 [[${dto.replyCount}]]</span> </h5>
</div>
<div class="list-group replyList">
</div>
<div class="list-group replyList"
</div>

<!--      java 의 데이터를 JavaScript 변수에 바로 대입이 가능-->
<script th:inline="javascript">
    // 자바의 dto.bon 의 값이 bon에 대입
    var bon = [[${dto.bon}]]

    // 댓글이 추가 될 영역
    var listGroup = $(".replyList");
    // replyList 영역에 댓글이 추가가 될 것이다 .

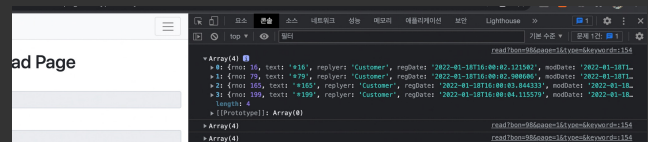
    //날짜 데이터를 문자열로 변환해서 리턴해주는 함수
    function formatTime(str){
        //날짜 객체 생성
        var date = new Date(str);
        // 문자열을 생성

        return date.getFullYear()+"/"+(date.getMonth()+1)+"/"
            +date.getDate()+" "+date.getHours()+":"+date.getMinutes()
        // JavaScript 의 Date 는 자바와 거의 똑같다
    }

    //댓글을 가져와서 출력하는 함수
    function loadJSONData(){
        //get 방식으로 ajax 요청
        $.getJSON('/replies/board/'+bon,function (arr){
            console.log(arr);
        });
    }

    // 클릭하면
    // JQuery 에서

    // 문서를 전부 읽고 나면
    $(document).ready(function(){
        // 댓글의 개수 부분을 클릭하면
        $(".replyCount").click(function (){
            loadJSONData();
        });
    })
</script>
</th:block>
</th:block>
```



```

function loadJSONData() {
    $.getJSON('/replies/board/'+bon, function(arr){
        console.log(arr);
        var str = "";
        $('#replyCount').html("댓글 수 " + arr.length);
        $.each(arr, function(idx, reply) {
            console.log(reply);
            str += '<div class="card-body" data-rno="' + reply.rno + '"><b>' + reply.rno + '</b>';
            str += '<h5 class="card-title">' + reply.text + '</h5>';
            str += '<h6 class="card-subtitle mb-2 text-muted">' + reply.replier + '</h6>';
            str += '<p class="card-text">' + formatTime(reply.regDate) + '</p>';
            str += '</div>';
        })
        listGroup.html(str);
    });
}

// 클릭하면
// JQuery 에서

// 문서를 전부 읽고 나면
$(document).ready(function(){
    // 댓글의 개수 부분을 클릭하면
    $('#replyCount').click(function () {
        loadJSONData();
    });
})
</script>
</th:block>
</th:block>

```

2022/01/18 15:49:05	
수정	목록
댓글 수 4	
16	16
Customer	
2022/1/18 16:0	
79	79
Customer	
2022/1/18 16:0	
165	165
Customer	
2022/1/18 16:0	
199	199
Customer	
2022/1/18 16:0	

댓글 추가 삭제 수정에 이용할 대화 상자 설정

```
<div class="modal" tabindex="-1" role="dialog">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Modal title</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span></button>
      </div>
      <div class="modal-body">
        <div class="form-group">
          <input class="form-control" type="text" name="replyText" placeholder="댓글 작성...">
        </div>
        <div class="form-group">
          <input class="form-control" type="text" name="replyer" placeholder="작성자..." >
          <input type="hidden" name="rno" >
        </div>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-danger replyRemove">삭제</button>
        <button type="button" class="btn btn-warning replyModify">수정</button>
        <button type="button" class="btn btn-primary replySave">추가</button>
        <button type="button" class="btn btn-outline-secondary replyClose" data-dismiss="modal">닫기</button>
      </div>
    </div>
  </div>
</div>
```

댓글 추가 요청을 위한 버튼을 생성

```
<h5><span class="'badge badge-info addReply'">댓글 달기</span></h5>
```

댓글 추가 영역을 눌렀을 때 처리할 Script Code 작성

```
var modal =$(".modal")
// 댓글 작성 버튼 누를 때
$(".addReply").click(function (e){
  modal.modal('show');

  // 댓글 입력할 부분을 초기화
  $('input[name="replyText"]').val(' ');
  $('input[name="replyer"]').val(' ');

  // 모든 버튼을 숨기기
  $('.modal-footer .btn').hide();

  // 필요한 버튼만 출력
  $('.replySave, .replyClose').show();
});
```


ReplyController에 삽입 요청 처리

```
package com.singsiuk.board.controller;

import com.singsiuk.board.dto.ReplyDTO;
import com.singsiuk.board.service.ReplyService;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

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

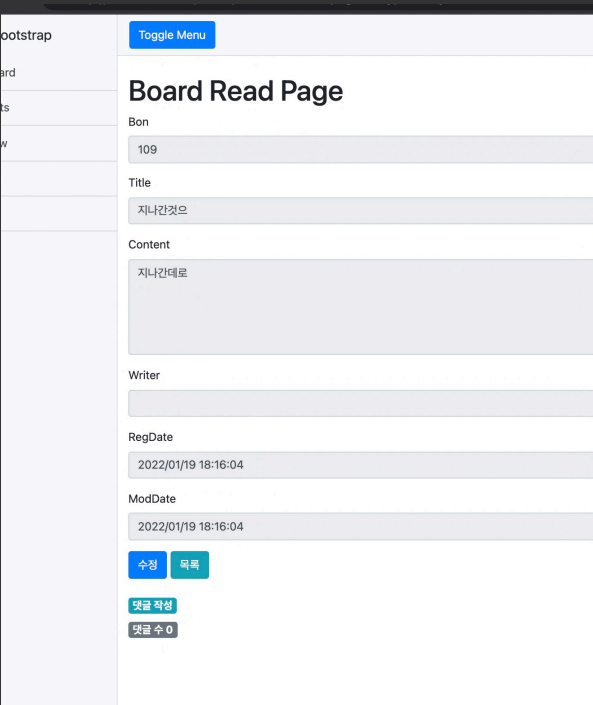
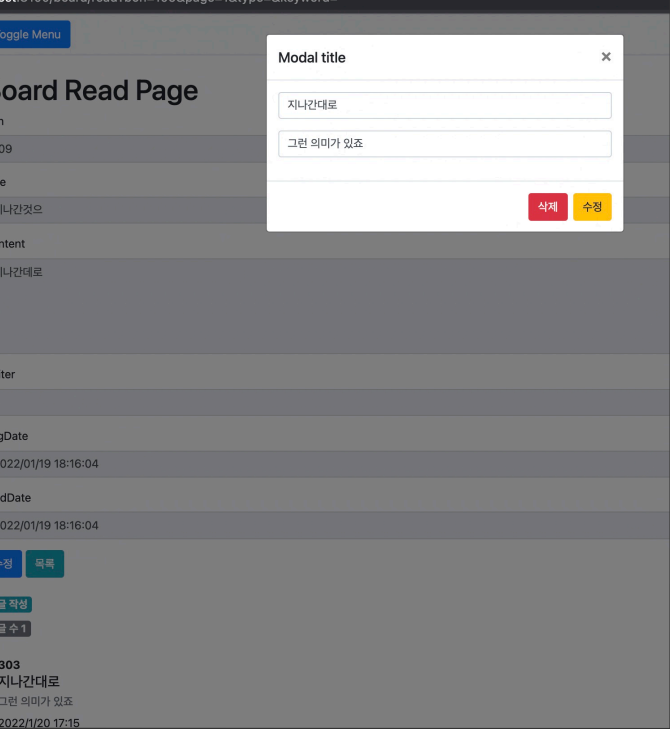
    private final ReplyService replyService;

    // 댓글 목록 요청을 처리하는 Method
    @GetMapping(value="/board/{bon}", produces = MediaType.APPLICATION_JSON_VALUE)
    public ResponseEntity<List<ReplyDTO>> getListByBon(
        @PathVariable("bon") Long bon){
        return new ResponseEntity<>(
            replyService.getList(bon), HttpStatus.OK);
    }

    @PostMapping("")
    public ResponseEntity<Long> register(@RequestBody ReplyDTO replyDTO){
        Long rno = replyService.register(replyDTO);
        return new ResponseEntity<>(rno, HttpStatus.OK);
    }
}
```


ReplyController에 댓글 삭제 요청을 처리하는 Method 구현

```
// 댓글 삭제 요청을 처리하는 method
@DeleteMapping("/{rno}")
public ResponseEntity<String> remove(@PathVariable("rno") Long rno){
    replyService.remove(rno);
    return new ResponseEntity<>("success",HttpStatus.OK);
}
```



댓글 수정

: 수정 요청 버튼을 눌렀을 때 수행될 Script 작성

```
// 수정 버튼을 눌렀을 때 처리
$(".replyModify").click(function() {
    // 수정에 사용할 데이터를 작성
    var rno = $("input[name='rno']").val();
    var reply = {
        rno: rno,
        bon: bon,
        text: $('input[name="replyText"]').val(),
        replyer: $('input[name="replyer"]').val()
    }
    console.log(reply);
    $.ajax({
        url: '/replies/' + rno,
        method: 'put',
        data: JSON.stringify(reply),
        contentType: 'application/json; charset=utf-8',
        success: function(result){
            console.log("RESULT: " + result);
            if(result === 'success'){
                alert("댓글이 수정되었습니다");
                modal.modal('hide');
                loadJSONData();
            }
        }
    });
});
</script>
</th:block>
</th:block>
```

Controller

```
@PutMapping("/{rno}")
public ResponseEntity<String> modify(@RequestBody ReplyDTO replyDTO){
    replyService.modify(replyDTO);
    return new ResponseEntity<>("success",HttpStatus.OK);
}
```

localhost:8100 내용:

댓글이 수정되었습니다

확인

Customerxcv

삭제

수정

정렬

ReplyServiceImpl 에서

```
@Override
public List<ReplyDTO> getList(Long bon) {
    // 글번호에 해당하는 댓글 가져오기
    List<Reply> result = replyRepository.getRepliesByBoardOrderByRno(
        Board.builder().bon(bon).build());

    // 댓글 정렬하기
    result.sort(new Comparator<Reply>() {
        @Override
        public int compare(Reply o1, Reply o2) {
            // return (int)(o2.getRno() - o1.getRno());
            return o2.getModDate().compareTo(o1.getModDate());
        }
    });

    return result.stream().map(reply -> entityToDTO(reply)).collect(Collectors.toList());
}
```