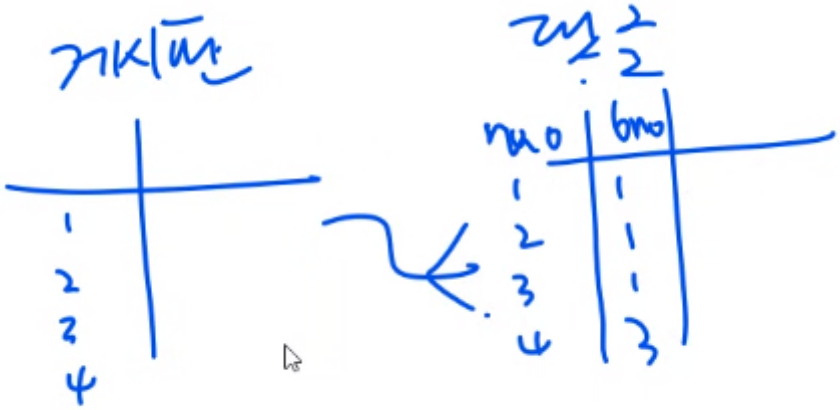
Spring Boot 6일차

엔티티간의 연관관계 JPA는 DB와 다르다

DB 는 1:1 , N:1, N:N

JPA 는 DB 를 고려 ERD를 잘 그려야 한다.

항상 테이블을 세로로 그려줘야 한다.



ERD를 제대로 그리지 못하면 나는 JPA를 못한다.

- 이걸 이력서에 써도 괜찮을 듯.
- 연관관계의 주인 FK

양뱡향 관계란?

- 양방향을 쓰면 매우 복잡해짐

mappedBy - 연관관계의 주인이 누구인가?

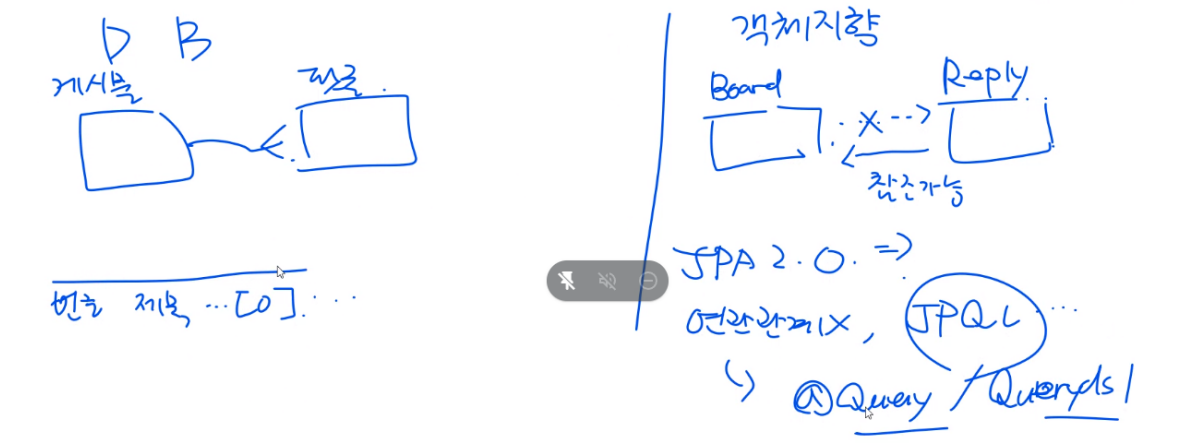
연관관계라는게 fk 가 있어야 형성이 되는것이기 때문에 fk 가 연관관계의 주인

- @ -> 주인 => 단방향
- @ManyToOne
    - fk -> pk

지연 로딩 **lazy** <---> 즉시 로딩 **eager**

- (서로 반댓말)

댓글 끝나면 해쉬태그 ㄱㄱ



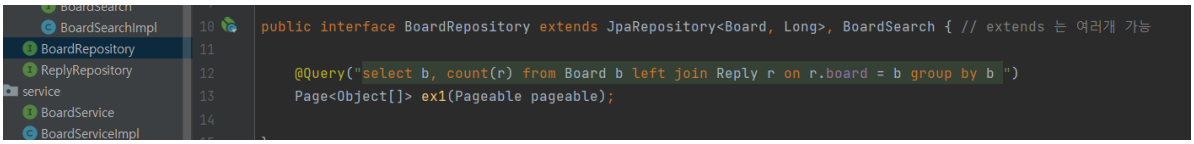
객체 지향에서

- Board 는 Reply를 참조하지 못한다.
- 하지만 Reply는 Board를 참조 가능하다.
- 이러면 생기는 문제!!
- 번호　제목 ...[0] 할때 이 댓글 개수를 측정하는데 문제가 생긴다.

하지만 JPA 2.0 버전이 올라가면서 연관관계가 없어도 JPQL 을 사용해 처리 가능해졌다.

그래서 @Query를 사용해 둘다 가져올수 있다

예를들어



이렇게

시작

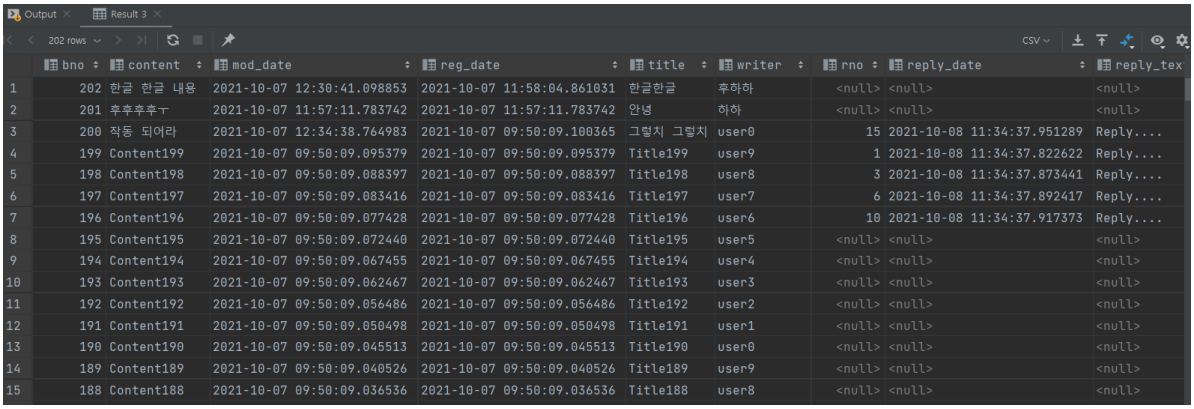
댓글 개수

sql 문 first

```
select * from board b left join reply r on b.bno = r.board_bno
group by b.bno
order by b.bno desc
;
```

mySql 과 마리아DB 에서는 오류가 없으나 Oracle 에서는 오류가 난다. 그래서 Oracle 에서는 나 min 처리 해줘야 한다고 한다.

결과



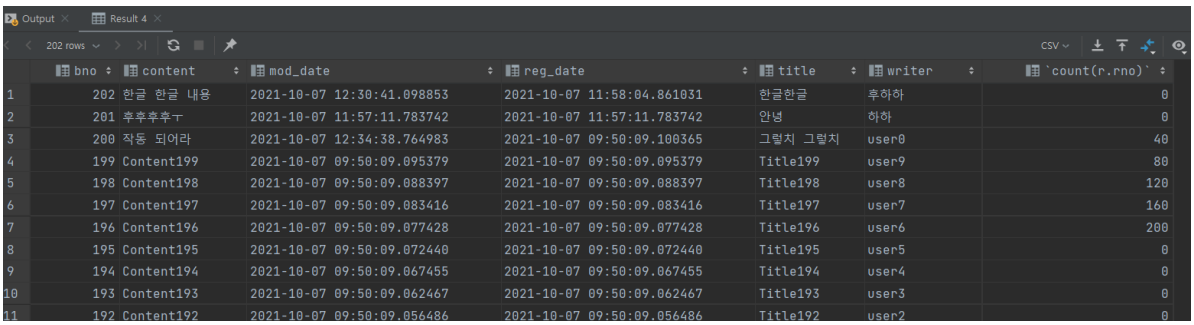
	bno	content	mod_date	reg_date	title	writer	rno	reply_date	reply_tex
1	202	한글 한글 내용	2021-10-07 12:30:41.098853	2021-10-07 11:58:04.861031	한글한글	후하하	<null>	<null>	<null>
2	201	후후후후ㅜ	2021-10-07 11:57:11.783742	2021-10-07 11:57:11.783742	안녕	하하	<null>	<null>	<null>
3	200	작동 되어라	2021-10-07 12:34:38.764983	2021-10-07 09:50:09.100365	그렇치 그렇치	user0	15	2021-10-08 11:34:37.951289	Reply....
4	199	Content199	2021-10-07 09:50:09.095379	2021-10-07 09:50:09.095379	Title199	user9	1	2021-10-08 11:34:37.822622	Reply....
5	198	Content198	2021-10-07 09:50:09.088397	2021-10-07 09:50:09.088397	Title198	user8	3	2021-10-08 11:34:37.873441	Reply....
6	197	Content197	2021-10-07 09:50:09.083416	2021-10-07 09:50:09.083416	Title197	user7	6	2021-10-08 11:34:37.892417	Reply....
7	196	Content196	2021-10-07 09:50:09.077428	2021-10-07 09:50:09.077428	Title196	user6	10	2021-10-08 11:34:37.917373	Reply....
8	195	Content195	2021-10-07 09:50:09.072440	2021-10-07 09:50:09.072440	Title195	user5	<null>	<null>	<null>
9	194	Content194	2021-10-07 09:50:09.067455	2021-10-07 09:50:09.067455	Title194	user4	<null>	<null>	<null>
10	193	Content193	2021-10-07 09:50:09.062467	2021-10-07 09:50:09.062467	Title193	user3	<null>	<null>	<null>
11	192	Content192	2021-10-07 09:50:09.056486	2021-10-07 09:50:09.056486	Title192	user2	<null>	<null>	<null>
12	191	Content191	2021-10-07 09:50:09.050498	2021-10-07 09:50:09.050498	Title191	user1	<null>	<null>	<null>
13	190	Content190	2021-10-07 09:50:09.045513	2021-10-07 09:50:09.045513	Title190	user0	<null>	<null>	<null>
14	189	Content189	2021-10-07 09:50:09.040526	2021-10-07 09:50:09.040526	Title189	user9	<null>	<null>	<null>
15	188	Content188	2021-10-07 09:50:09.036536	2021-10-07 09:50:09.036536	Title188	user8	<null>	<null>	<null>

저 결과에서 rno는 의미가 없다.

rno의 개수가 필요하다

그래서

```
select b.*, count(r.rno) from board b left join reply r on b.bno = r.board_bno
group by b.bno
order by b.bno desc
;
```



	bno	content	mod_date	reg_date	title	writer	`count(r.rno)`
1	202	한글 한글 내용	2021-10-07 12:30:41.098853	2021-10-07 11:58:04.861031	한글한글	후하하	0
2	201	후후후후ㅜ	2021-10-07 11:57:11.783742	2021-10-07 11:57:11.783742	안녕	하하	0
3	200	작동 되어라	2021-10-07 12:34:38.764983	2021-10-07 09:50:09.100365	그렇치 그렇치	user0	40
4	199	Content199	2021-10-07 09:50:09.095379	2021-10-07 09:50:09.095379	Title199	user9	80
5	198	Content198	2021-10-07 09:50:09.088397	2021-10-07 09:50:09.088397	Title198	user8	120
6	197	Content197	2021-10-07 09:50:09.083416	2021-10-07 09:50:09.083416	Title197	user7	160
7	196	Content196	2021-10-07 09:50:09.077428	2021-10-07 09:50:09.077428	Title196	user6	200
8	195	Content195	2021-10-07 09:50:09.072440	2021-10-07 09:50:09.072440	Title195	user5	0
9	194	Content194	2021-10-07 09:50:09.067455	2021-10-07 09:50:09.067455	Title194	user4	0
10	193	Content193	2021-10-07 09:50:09.062467	2021-10-07 09:50:09.062467	Title193	user3	0
11	192	Content192	2021-10-07 09:50:09.056486	2021-10-07 09:50:09.056486	Title192	user2	0

요렇게

JPQL 을 이용하면 이것들을 좀더 수월하게 할 수 있다고 한다.

```
public interface BoardRepository extends JpaRepository<Board, Long>, BoardSearch
{ // extends 는 여러개 가능

    @Query("select b.bno, b.title, b.writer, b.regDate, count(r) from Board b
left join Reply r on r.board = b group by b ") // 이렇게 수정
    Page<Object[]> ex1(Pageable pageable);
    // select 에 파라미터가 여러개이다! 그럼 무조건 Object[] 배열로 뽑을것.
}
```

이렇게 select에 파라미터가 2개 이상인 경우!! Object[] 로 뽑는다!!

정리

1 규칙 파라미터가 Pageable 이면 리턴 타입은 Page

2 규칙 내가 select 를 여러개 한다!! 그러면 무조건 Object[ ] 배열이다

그다음 테스트 ㄱㄱ

BoardRepositoryTests ㄱㄱ

```
@Test
public void testEx1() {

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

    Page<Object[]> result = boardRepository.ex1(pageable);

    log.info(result);

    result.get().forEach(element -> { // 이게 배열 안에 배열이 있는 것이다.

        Object[] arr = (Object[])element;

        log.info(Arrays.toString(arr));
        // 배열안에 배열은 이렇게 Arrays.toString() 으로 inner 배열을 뽑아 올수 있나보다.

    });
}
```

이거 테스트

```
.541  INFO 23528 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : Page 1 of 21 containing [Ljava.lang.Object; instances
.543  INFO 23528 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [202, 한글한글, 후하하, 2021-10-07T11:58:04.861031, 0]
.544  INFO 23528 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [201, 안녕, 하하, 2021-10-07T11:57:11.783742, 0]
.544  INFO 23528 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [200, 그렇치 그렇치, user0, 2021-10-07T09:50:09.100365, 40]
.544  INFO 23528 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [199, Title199, user9, 2021-10-07T09:50:09.095379, 80]
.544  INFO 23528 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [198, Title198, user8, 2021-10-07T09:50:09.088397, 120]
.545  INFO 23528 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [197, Title197, user7, 2021-10-07T09:50:09.083416, 160]
.545  INFO 23528 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [196, Title196, user6, 2021-10-07T09:50:09.077428, 200]
.545  INFO 23528 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [195, Title195, user5, 2021-10-07T09:50:09.072440, 0]
.545  INFO 23528 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [194, Title194, user4, 2021-10-07T09:50:09.067455, 0]
.545  INFO 23528 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [193, Title193, user3, 2021-10-07T09:50:09.062467, 0]
.581  INFO 23528 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
.584  INFO 23528 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
.592  INFO 23528 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
14s
```

하지만 우리는 동적 검색을 만들것이다. 검색 조건에 따라 검색이 달라지는것!!

그래서 우리는 위에것을 사용 못한다고 한다.

검색 기능

BoardSearch ㄱㄱ

```
public interface BoardSearch {

    Page<Board> search1(char[] typeArr, String keyword, Pageable pageable);

    Page<Object[]> searchWithReplyCount(); // 추가
}
```

BoardSearchImpl ㄱㄱ

implement method 해주기

```
.
.
.

@Override
public Page<Object[]> searchWithReplyCount() {
    log.info("==============  searchWithReplyCount  ============");


    return null;
}
```

테스트 ㄱㄱ

BoardRepositoryTests ㄱㄱ

```
@Test
public void testSearchWithReplyCount() {

    boardRepository.searchWithReplyCount();

}
```

결과

```
o.z.sb.repository.BoardRepositoryTests   : Started BoardRepositoryTests in 8.236 seconds (JVM running for 10.514)
o.z.s.repository.search.BoardSearchImpl  : ==============  searchWithReplyCount  ============
j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
```

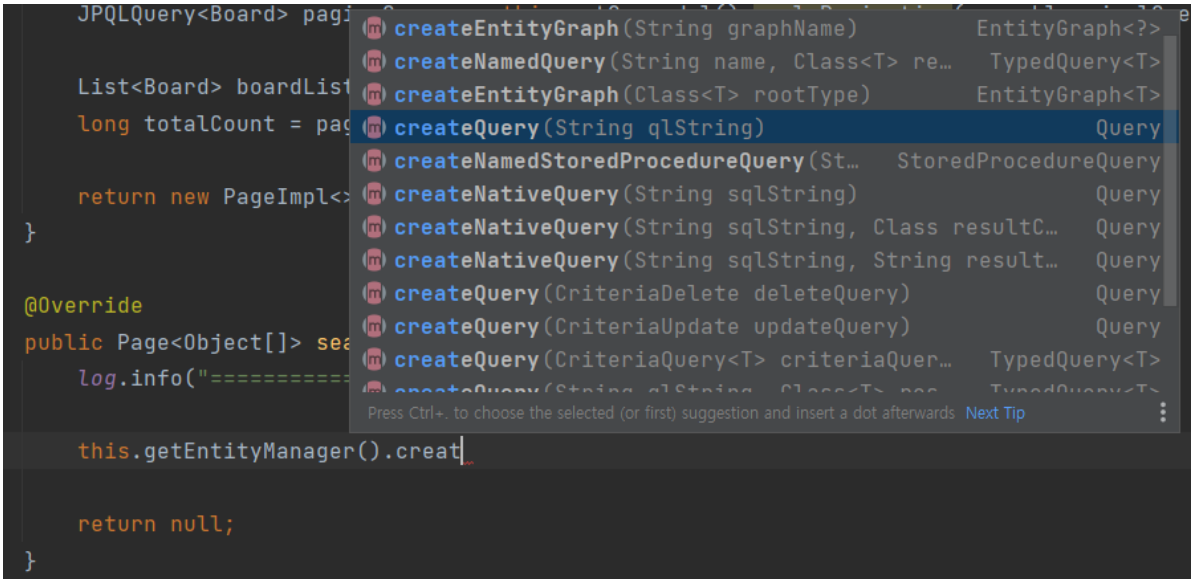
============== searchWithReplyCount ============ 가 뜨면 성공

쿼리문 준비!! - BoardSearchImpl

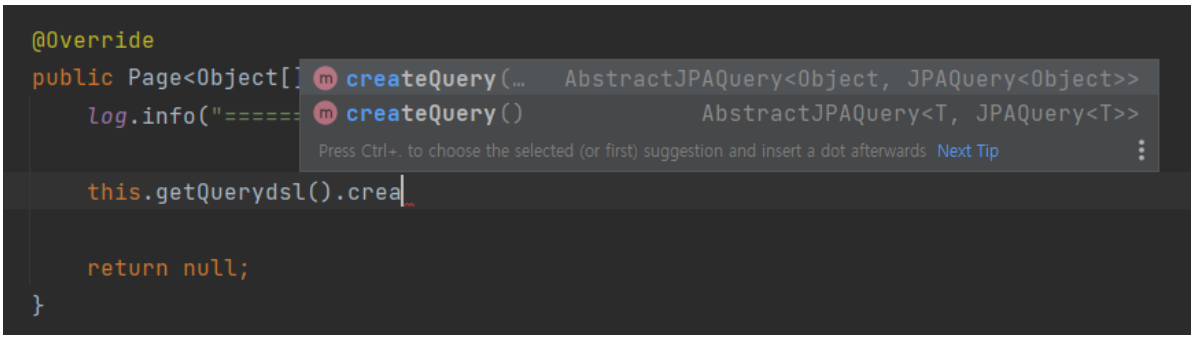
BoardSearchImpl ㄱㄱ

QuerydslRepositorySupport 를 extends 했기때문에 this.getEntityManager(), this.getQuerydsl() 이런 것들을 활용 가능

쿼리 개발 방식 1



쿼리 개발 방식 2



쿼리 개발 방식 3 사용

```
    @Override
    public Page<Object[]> searchWithReplyCount() {
        log.info("==============  searchWithReplyCount  ============");


        // 1 방법.  this.getEntityManager() 를 사용해 쿼리를 만듬
        // 2 방법.  this.getQuerydsl() 를 사용해 쿼리를 만듬
        // 3 번째 방법 사용
        // Query를 만들때는 Q도메인을 사용 , 값을 뽑을때는 엔티티 타입 사용
        QBoard qBoard = QBoard.board;
        QReply qReply = QReply.reply;

        JPQLQuery<Board> query = from(qBoard);
        query.leftJoin(qReply).on(qReply.board.eq(qBoard));
        query.where(qBoard.bno.eq(200L));
        query.groupBy(qBoard);

        log.info(query); // 자 로그 찍어서 쿼리가 잘 만들어졌나 확인해보자.

        return null;
    }
```

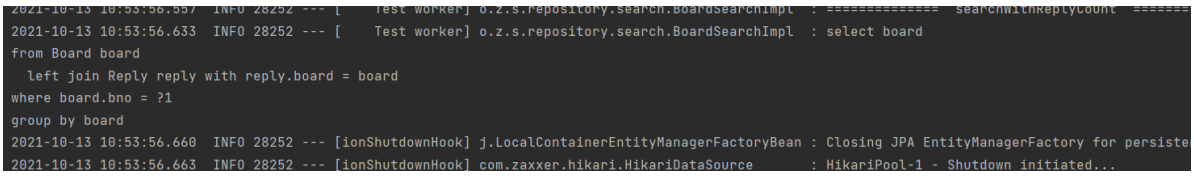
BoardRepositoryTests ㄱㄱ

그다음 테스트 ㄱㄱ

```
@Test
public void testSearchWithReplyCount() {

    boardRepository.searchWithReplyCount();

}
```

실행

결과



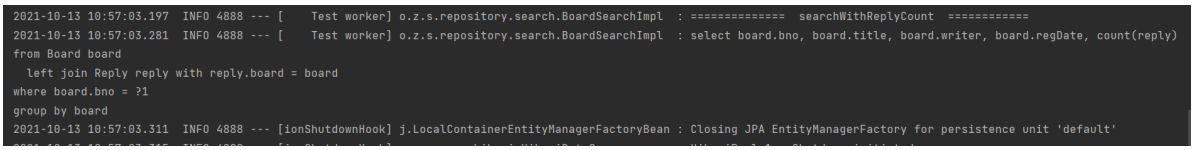
쿼리문이 만들어진것을 볼 수 있다.

그다음

```
@Override
public Page<Object[]> searchWithReplyCount() {
    log.info("===============  searchWithReplyCount  =============");


    // 1 방법.  this.getEntityManager() 를 사용해 쿼리를 만듬
    // 2 방법.  this.getQuerydsl() 를 사용해 쿼리를 만듬
    // 3 번째 방법 사용
    // Query를 만들때는 Q도메인을 사용 , 값을 뽑을때는 엔티티 타입 사용
    QBoard qBoard = QBoard.board;
    QReply qReply = QReply.reply;

    // Board b left join Reply r on r.board.bno = b.bno 란 뜻 (밑에)
    JPQLQuery<Board> query = from(qBoard);
    query.leftJoin(qReply).on(qReply.board.eq(qBoard));
    query.where(qBoard.bno.eq(200L));
    query.groupBy(qBoard);

    // 추가
    // 자 이제 우리가 원하는 애들만 뽑게 가공시켜보자
    JPQLQuery<Tuple> selectQuery = query.select(qBoard.bno, qBoard.title,
qBoard.writer, qBoard.regDate, qReply.count()); // 추가

    log.info(selectQuery); // 추가

    return null;
}
```

BoardRepositoryTests ㄱㄱ

테스트 ㄱㄱ



이렇게 뜬다.

위의 방식이 정석적인 방법

```
@Override
public Page<Object[]> searchWithReplyCount() {
    log.info("==============  searchWithReplyCount  ============");


    // 1 방법.  this.getEntityManager() 를 사용해 쿼리를 만듬
    // 2 방법.  this.getQuerydsl() 를 사용해 쿼리를 만듬
    // 3 번째 방법 사용
    // Query를 만들때는 Q도메인을 사용 , 값을 뽑을때는 엔티티 타입 사용
    QBoard qBoard = QBoard.board;
    QReply qReply = QReply.reply;

    // Board b left join Reply r on r.board.bno = b.bno 란 뜻 (밑에)
    JPQLQuery<Board> query = from(qBoard);
    query.leftJoin(qReply).on(qReply.board.eq(qBoard));
    query.where(qBoard.bno.eq(200L));
    query.groupBy(qBoard);

    // 자 이제 우리가 원하는 애들만 뽑게 가공시켜보자
    JPQLQuery<Tuple> selectQuery = query.select(qBoard.bno, qBoard.title,
qBoard.writer, qBoard.regDate, qReply.count());

    // 추가
    // 자 이제 진짜 쿼리문을 만들어보자
    List<Tuple> tupleList = selectQuery.fetch(); // 추가

    log.info(tupleList);

    return null;
}
```

```
2021-10-13 11:00:51.490  INFO 25320 --- [    Test worker] o.
Hibernate:
    select
        board0_.bno as col_0_0_,
        board0_.title as col_1_0_,
        board0_.writer as col_2_0_,
        board0_.reg_date 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_.bno=?
    group by
        board0_.bno
2021-10-13 11:00:51.637  INFO 25320 --- [    Test worker] o.
```

하지만!!! 이 쿼리문은 못쓴다!!!

왜 못쓰나.

데이터는 량이 100만건 1000만건이다. 하지만 이 모든것들을 left join 걸어서 가져온다 하면 엄청 부담이 커진다.

자 그럼 이제 저 위에 쿼리를 우리한테 맞게 바꿔보자!!

BoardSearch ㄱㄱ

```
public interface BoardSearch {

    Page<Board> search1(char[] typeArr, String keyword, Pageable pageable);

    Page<Object[]> searchWithReplyCount(Pageable pageable); // 수정
}
```

BoardSearchImpl ㄱㄱ

```
@Override
```

```
    public Page<Object[]> searchWithReplyCount(Pageable pageable) { // 수정
        log.info("===============  searchWithReplyCount  ============");


        // 1 방법.  this.getEntityManager() 를 사용해 쿼리를 만듬
        // 2 방법.  this.getQuerydsl() 를 사용해 쿼리를 만듬
        // 3 번째 방법 사용
        // Query를 만들때는 Q도메인을 사용 , 값을 뽑을때는 엔티티 타입 사용
        QBoard qBoard = QBoard.board;
        QReply qReply = QReply.reply;

        // Board b left join Reply r on r.board.bno = b.bno 란 뜻 (밑에)
        JPQLQuery<Board> query = from(qBoard);
        query.leftJoin(qReply).on(qReply.board.eq(qBoard));
        query.groupBy(qBoard);


        // 자 이제 우리가 원하는 애들만 뽑게 가공시켜보자
        JPQLQuery<Tuple> selectQuery = query.select(qBoard.bno, qBoard.title,
qBoard.writer, qBoard.regDate, qReply.count());

        this.getQuerydsl().applyPagination(pageable, selectQuery); // 추가

        // 자 이제 진짜 쿼리문을 만들어보자
        List<Tuple> tupleList = selectQuery.fetch();

        log.info(selectQuery);

        return null;
    }
```

테스트 ㄱㄱ

```
@Test
public void testSearchWithReplyCount() {

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

    boardRepository.searchWithReplyCount(pageable); // 수정

}
```

결과

```
2021-10-13 11:11:44.361  INFO 20208 --- [    Test worke
Hibernate:
    select
        board0_.bno as col_0_0_,
        board0_.title as col_1_0_,
        board0_.writer as col_2_0_,
        board0_.reg_date 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
            )
    group by
        board0_.bno
    order by
        board0_.bno desc limit ?,
        ?
2021-10-13 11:11:44.479  INFO 20208 --- [    Test worke
```

밑에 limit 가 떠야 한다.

그다음

```java
@Override
public Page<Object[]> searchWithReplyCount(Pageable pageable) {
    log.info("===============  searchWithReplyCount  ============");


    // 1 방법.  this.getEntityManager() 를 사용해 쿼리를 만듬
    // 2 방법.  this.getQuerydsl() 를 사용해 쿼리를 만듬
    // 3 번째 방법 사용
    // Query를 만들때는 Q도메인을 사용 , 값을 뽑을때는 엔티티 타입 사용
    QBoard qBoard = QBoard.board;
    QReply qReply = QReply.reply;

    // Board b left join Reply r on r.board.bno = b.bno 란 뜻 (밑에)
    JPQLQuery<Board> query = from(qBoard);
    query.leftJoin(qReply).on(qReply.board.eq(qBoard));
    query.groupBy(qBoard);


    // 자 이제 우리가 원하는 애들만 뽑게 가공시켜보자
```

```
    JPQLQuery<Tuple> selectQuery = query.select(qBoard.bno, qBoard.title,
qBoard.writer, qBoard.regDate, qReply.count());

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

    // 자 이제 진짜 쿼리문을 만들어보자
    List<Tuple> tupleList = selectQuery.fetch();
    long totalCount = selectQuery.fetchCount();

    // tutple 에서 Array 로 변환시켜서 List로 바꿔줘야한다.
    List<Object[]> arr = tupleList.stream().map(tuple ->
tuple.toArray()).collect(Collectors.toList());

    return new PageImpl<>(arr, pageable, totalCount);
}
```

테스트 ㄱㄱ

```
@Test
public void testSearchWithReplyCount() {

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

    Page<Object[]> result = boardRepository.searchWithReplyCount(pageable); // 수
정

    log.info("total:" + result.getTotalPages()); // 추가

    result.get().forEach(arr -> { // 추가
        log.info(Arrays.toString(arr));
    });

}
```

결과

```
Hibernate:
    select
        board0_.bno as col_0_0_,
        board0_.title as col_1_0_,
        board0_.writer as col_2_0_,
        board0_.reg_date 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
            )
    group by
        board0_.bno
    order by
        board0_.bno desc limit ?,
        ?
Hibernate:
    select
        count(distinct board0_.bno) as col_0_0_
    from
        board board0_
    left outer join
        reply reply1_
            on (
                reply1_.board_bno=board0_.bno
            )
```

```
INFO 20208 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : total:21
INFO 20208 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [202, 한글한글, 후하하, 2021-10-07T11:58:04.861031, 0]
INFO 20208 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [201, 안녕, 하하, 2021-10-07T11:57:11.783742, 0]
INFO 20208 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [200, 그렇치 그렇치, user0, 2021-10-07T09:50:09.100365, 40]
INFO 20208 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [199, Title199, user9, 2021-10-07T09:50:09.095379, 80]
INFO 20208 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [198, Title198, user8, 2021-10-07T09:50:09.088397, 120]
INFO 20208 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [197, Title197, user7, 2021-10-07T09:50:09.083416, 160]
INFO 20208 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [196, Title196, user6, 2021-10-07T09:50:09.077428, 200]
INFO 20208 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [195, Title195, user5, 2021-10-07T09:50:09.072440, 0]
INFO 20208 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [194, Title194, user4, 2021-10-07T09:50:09.067455, 0]
INFO 20208 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [193, Title193, user3, 2021-10-07T09:50:09.062467, 0]
INFO 20208 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
INFO 20208 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
INFO 20208 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
```

이렇게 나오면 성공!!!!

리스트에 댓글수 뜨게 하기

BoardSearch ㄱㄱ

```
public interface BoardSearch {

    Page<Board> search1(char[] typeArr, String keyword, Pageable pageable);

    Page<Object[]> searchWithReplyCount(char[] typeArr, String keyword, Pageable
pageable); // 수정
}
```

BoardSearchImpl

```
@Override
public Page<Object[]> searchWithReplyCount(char[] typeArr, String keyword,
Pageable pageable) { // 수정
    log.info("==============  searchWithReplyCount  ============");


    // 1 방법.  this.getEntityManager() 를 사용해 쿼리를 만듬
    // 2 방법.  this.getQuerydsl() 를 사용해 쿼리를 만듬
    // 3 번째 방법 사용
    // Query를 만들때는 Q도메인을 사용 , 값을 뽑을때는 엔티티 타입 사용
    QBoard qBoard = QBoard.board;
    QReply qReply = QReply.reply;

    // Board b left join Reply r on r.board.bno = b.bno 란 뜻 (밑에)
    JPQLQuery<Board> query = from(qBoard);
    query.leftJoin(qReply).on(qReply.board.eq(qBoard));
    query.groupBy(qBoard);

    // 검색 조건 // 추가
    if (typeArr != null && typeArr.length > 0) { // 추가

        BooleanBuilder condition = new BooleanBuilder(); // 추가

        for (char type : typeArr) { // 추가
            if (type == 'T') {
                condition.or(qBoard.title.contains(keyword)); // 추가
            } else if (type == 'C') {
                condition.or(qBoard.content.contains(keyword)); // 추가
            } else if (type == 'W') {
                condition.or(qBoard.writer.contains(keyword)); // 추가
            }
        }
        query.where(condition); // 추가
    } // 추가


    // 자 이제 우리가 원하는 애들만 뽑게 가공시켜보자
```

```
    JPQLQuery<Tuple> selectQuery = query.select(qBoard.bno, qBoard.title,
qBoard.writer, qBoard.regDate, qReply.count());

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

    // 자 이제 진짜 쿼리문을 만들어보자
    List<Tuple> tupleList = selectQuery.fetch();
    long totalCount = selectQuery.fetchCount();

    // tutple 에서 Array 로 변환시켜서 List로 바꿔줘야한다.
    List<Object[]> arr = tupleList.stream().map(tuple ->
tuple.toArray()).collect(Collectors.toList());

    return new PageImpl<>(arr, pageable, totalCount);
}
```

테스트 ㄱㄱ

```
@Test
public void testSearchWithReplyCount() {

    char[] typeArr = {'T'}; // 추가
    String keyword = "10"; // 추가
    Pageable pageable = PageRequest.of(0, 10, Sort.by("bno").descending());

    Page<Object[]> result = boardRepository.searchWithReplyCount(typeArr,
keyword, pageable); // 수정

    log.info("total:" + result.getTotalPages());

    result.get().forEach(arr -> {
        log.info(Arrays.toString(arr));
    });

}
```

결과

제목에 10이 들어가는 애들만 검색이 되어 나열된다!!! 페이징까지 되어서!!!!

```
cape '!'
8404 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : total:2
8404 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [110, Title110, user0, 2021-10-07T09:50:08.686471, 0]
8404 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [109, Title109, user9, 2021-10-07T09:50:08.682484, 0]
8404 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [108, Title108, user8, 2021-10-07T09:50:08.678494, 0]
8404 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [107, Title107, user7, 2021-10-07T09:50:08.673507, 0]
8404 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [106, Title106, user6, 2021-10-07T09:50:08.668520, 0]
8404 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [105, Title105, user5, 2021-10-07T09:50:08.664530, 0]
8404 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [104, Title104, user4, 2021-10-07T09:50:08.660540, 0]
8404 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [103, Title103, user3, 2021-10-07T09:50:08.657549, 0]
8404 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [102, Title102, user2, 2021-10-07T09:50:08.653561, 0]
8404 --- [    Test worker] o.z.sb.repository.BoardRepositoryTests   : [101, Title101, user1, 2021-10-07T09:50:08.649569, 0]
8404 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
8404 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
```

공부 순서

쿼리 어노테이션으로 먼저 만들어보고 대신 동적인 검색은 안된다.

그다음 오늘 했던 식으로 동적인 검색 조건 만들어보기.

```
@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class BoardListDTO {

    private Long bno;
    private String title;
    private String writer;
    private LocalDateTime regDate;
    private long replyCount; // 값이 없을때는 null 이 아닌 0이였음 좋겠어서 long 이라고 함  Long 이 아니라
}
```

BoardService ㄱㄱ

```
@Transactional
public interface BoardService {

    Long register(BoardDTO boardDTO);

    PageResponseDTO<BoardDTO> getList(PageRequestDTO pageRequestDTO);

    PageResponseDTO<BoardListDTO> getListWithReplyCount(PageRequestDTO pageRequestDTO); // 추가

    BoardDTO read(Long bno);

    void modify(BoardDTO boardDTO);

    void remove(Long bno);
}
```

BoardServiceImpl ㄱㄱ

implement method

```
@Override
    public PageResponseDTO<BoardListDTO> getListWithReplyCount(PageRequestDTO pageRequestDTO) {
        // 이부분 테스트 부분에서 복붙하옴
        char[] typeArr = pageRequestDTO.getTypes();
        String keyword = pageRequestDTO.getKeyword();
        Pageable pageable = PageRequest.of(
```

```
                pageRequestDTO.getPage() - 1,
                pageRequestDTO.getSize(),
                Sort.by("bno").descending());
        Page<Object[]> result = boardRepository.searchWithReplyCount(typeArr,
keyword, pageable);


        List<BoardListDTO> dtoList = result.get().map(objects -> {
           BoardListDTO listDTO = BoardListDTO.builder()
                   .bno((Long)objects[0])
                   .title((String)objects[1])
                   .writer((String)objects[2])
                   .regDate((LocalDateTime)objects[3])
                   .replyCount((Long)objects[4])
                   .build();
           return listDTO;
        }).collect(Collectors.toList());

        return new PageResponseDTO<>(pageRequestDTO,
(int)result.getTotalElements(), dtoList);
    }
```

BoardController ㄱㄱ

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

//        model.addAttribute("responseDTO",
boardService.getList(pageRequestDTO));
        model.addAttribute("responseDTO",
boardService.getListWithReplyCount(pageRequestDTO));
    }
```

---

n+1 문제

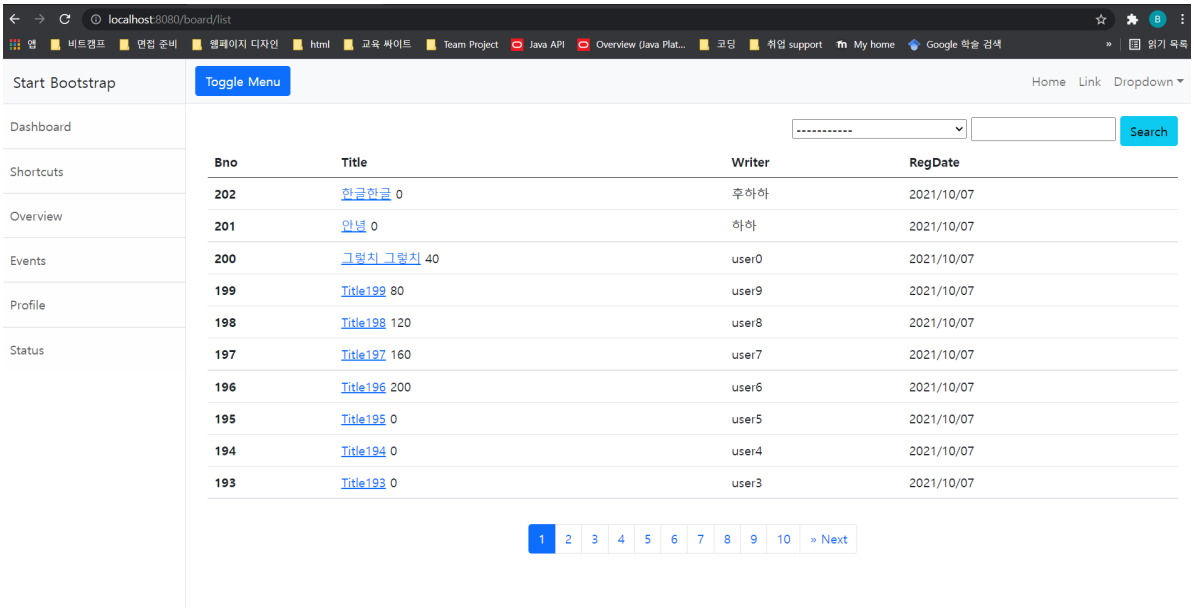
게시물 하나당 replyCount 를 가져오는 쿼리문을 계속 실행 시켜주는 문제

---

list.html ㄱㄱ

```
<tbody>
<tr th:each="board:${responseDTO.dtoList}">
    <th scope="row">[[${board.bno}]]</th>
    <td><a th:href="|javascript:movePage(${responseDTO.page}, ${board.bno} )|">
[[${board.title}]]</a>
        [[${board.replyCount}]] // 이거 추가
    </td>
    <td>[[${ board.writer}]]</td>
    <td>[[${#temporals.format(board.regDate, 'yyyy/MM/dd')}]]</td>
</tr>
</tbody>
```

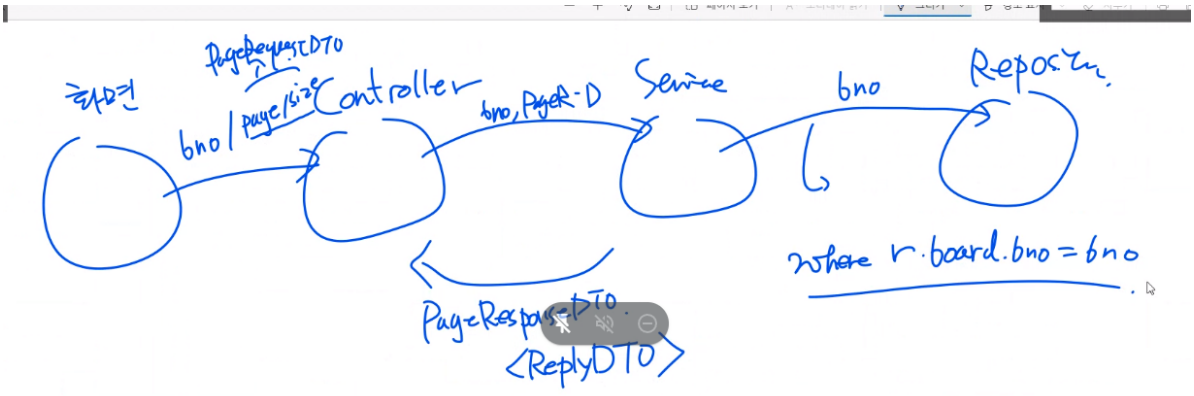
그리고나서 sbApplication 실행 ㄱㄱ



이렇게 replyCount가 뜨는것을 볼 수 있다.

성공!!!!!

조회 페이지 - 댓글 리스트 (paging) 출력



이번것은 동적인것이 아닌 즉 계속 바뀌는 값이 아닌 정적인것 즉 바뀌지 않는 값이기 때문에 @Query 어노테이션을 사용해도 된다.

그리고 post 방식으로 rest 방식을 테스트 할것이다.

ReplyRepository ㄱㄱ

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

    List<Reply> findReplyByBoard_BnoOrderByRno(Long bno);

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

}
```

ReplyRepositoryTests ㄱㄱ

```
@Test
public void testListOfBoard() {

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

    Page<Reply> result = replyRepository.getListByBno(198L, pageable);

    log.info(result.getTotalElements());

    result.get().forEach(reply -> log.info(reply));

}
```

결과

```
y.ReplyRepositoryTests   : 120
y.ReplyRepositoryTests   : Reply(rno=590, replyText=Reply...., replyer=replyer..., replyDate=2021-10-08T11:34:39.980114)
y.ReplyRepositoryTests   : Reply(rno=589, replyText=Reply...., replyer=replyer..., replyDate=2021-10-08T11:34:39.978121)
y.ReplyRepositoryTests   : Reply(rno=588, replyText=Reply...., replyer=replyer..., replyDate=2021-10-08T11:34:39.976127)
y.ReplyRepositoryTests   : Reply(rno=575, replyText=Reply...., replyer=replyer..., replyDate=2021-10-08T11:34:39.946176)
y.ReplyRepositoryTests   : Reply(rno=574, replyText=Reply...., replyer=replyer..., replyDate=2021-10-08T11:34:39.944208)
y.ReplyRepositoryTests   : Reply(rno=573, replyText=Reply...., replyer=replyer..., replyDate=2021-10-08T11:34:39.942218)
y.ReplyRepositoryTests   : Reply(rno=560, replyText=Reply...., replyer=replyer..., replyDate=2021-10-08T11:34:39.916257)
y.ReplyRepositoryTests   : Reply(rno=559, replyText=Reply...., replyer=replyer..., replyDate=2021-10-08T11:34:39.913295)
y.ReplyRepositoryTests   : Reply(rno=558, replyText=Reply...., replyer=replyer..., replyDate=2021-10-08T11:34:39.911300)
y.ReplyRepositoryTests   : Reply(rno=545, replyText=Reply...., replyer=replyer..., replyDate=2021-10-08T11:34:39.886397)
EntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'
i.HikariDataSource       : HikariPool-1 - Shutdown initiated...
```

그다음

ReplyDTO 클래스 생성

그 안에

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

    private Long rno;
```

```
    private String replyText;

    private String replyer;

    private Long bno;

    private LocalDateTime replyDate;

}
```

ReplyService 인테페이스 생성

Transactional 어노테이션 걸어주기

```
@Transactional
public interface ReplyService {

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

ReplyServiceImpl 클래스 생성

implements ReplyService 해주고 implement method 해주기

그다음 싹다 추가

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

    private final ModelMapper modelMapper;

    private final ReplyRepository replyRepository;

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

        Pageable pageable = PageRequest.of(pageRequestDTO.getPage() -1,
pageRequestDTO.getSize(), Sort.by("rno").descending());

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

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

        dtoList.forEach(replyDTO -> log.info(replyDTO));

        return null;
    }
}
```

그다음 테스트 ㄱㄱ

ReplyServiceTests 클래스 생성

그다음 아래처럼 추가

```
@SpringBootTest
@Log4j2
public class ReplyServiceTests {

    @Autowired
    private ReplyService replyService;

    @Test
    public void testList() {
        Long bno = 196L;

        PageRequestDTO pageRequestDTO = PageRequestDTO.builder().build();

        replyService.getListOfBoard(bno, pageRequestDTO);
    }

}
```

결과

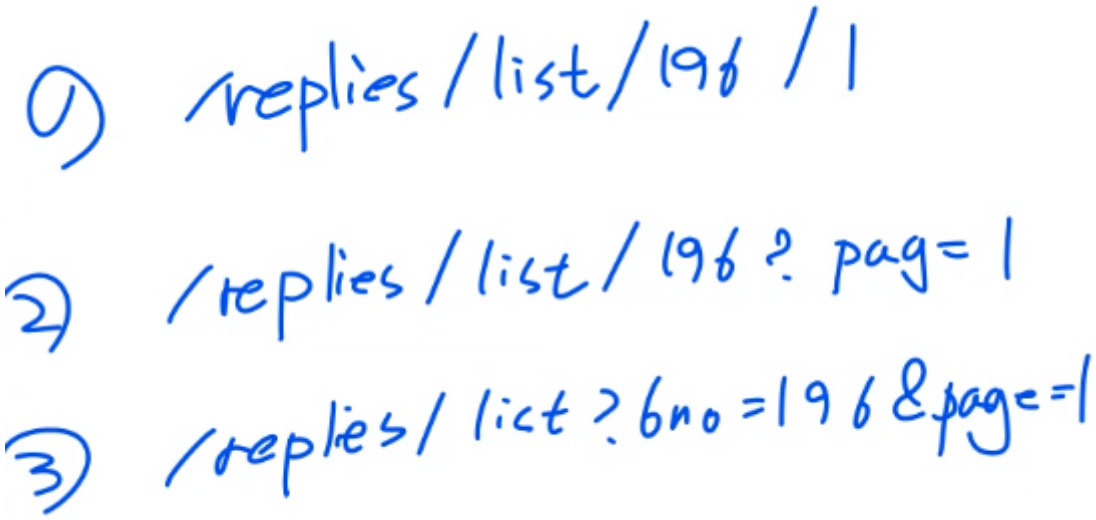
ReplyServiceImpl ㄱㄱ

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

    private final ModelMapper modelMapper;

    private final ReplyRepository replyRepository;

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

        Pageable pageable = PageRequest.of(pageRequestDTO.getPage() -1,
pageRequestDTO.getSize(), Sort.by("rno").descending());

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

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

//        dtoList.forEach(replyDTO -> log.info(replyDTO));

        return new PageResponseDTO<>(pageRequestDTO,
(int)result.getTotalElements(), dtoList); // 추가
    }
}
```

그다음

ReplyController 생성 - Rest Controller 이다.



/replies/200 에서 저 200 은 pk 이기때문에 고정된 값이다. 값이 바뀌지 않는다. 이런 애들은 강사님은 보통 파라미터로 처리한다고 한다.

그래서 2번 선택(?)

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

    private final ReplyService replyService;

    @GetMapping("/list/{bno}")
    public PageResponseDTO<ReplyDTO> getListOfBoard(@PathVariable("bno") Long
bno, PageRequestDTO pageRequestDTO) {
        return replyService.getListOfBoard(bno, pageRequestDTO);
    }
}
```

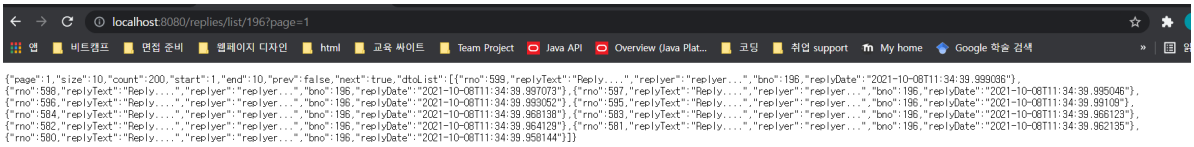
이거는 get 방식이라 바로 테스트가 가능

post man 안써도 됨.

그다음 실행하면 웹이 떠야함

페이지 1

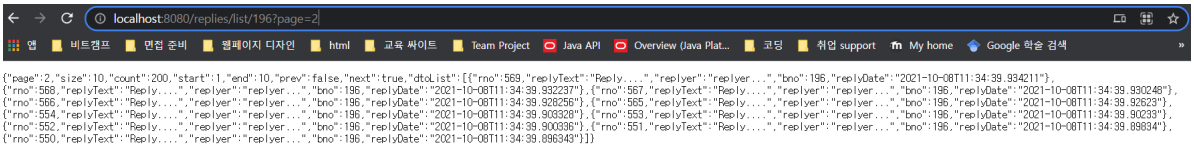
http://localhost:8080/replies/list/196?page=1



{"page":1,"size":10,"count":200,"start":1,"end":10,"prev":false,"next":true,"dtoList":[{"rno":599,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.999036"},
{"rno":598,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.997073"},{"rno":597,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.995046"},
{"rno":596,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.993052"},{"rno":595,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.99109"},
{"rno":594,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.968138"},{"rno":593,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.966123"},
{"rno":592,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.964129"},{"rno":591,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.962135"},
{"rno":590,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.958144"}]}

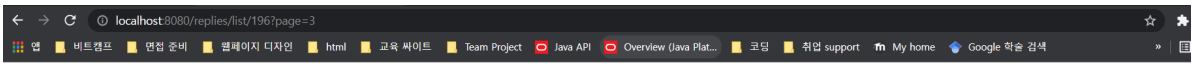
페이지 2

http://localhost:8080/replies/list/196?page=2



{"page":2,"size":10,"count":200,"start":1,"end":10,"prev":false,"next":true,"dtoList":[{"rno":569,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.934211"},
{"rno":568,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.932237"},{"rno":567,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.930248"},
{"rno":566,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.928256"},{"rno":565,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.92623"},
{"rno":554,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.903328"},{"rno":553,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.90233"},
{"rno":552,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.900336"},{"rno":551,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.89834"},
{"rno":550,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.896343"}]}

페이지 3



{"page":3,"size":10,"count":200,"start":1,"end":10,"prev":false,"next":true,"dtoList":[{"rno":539,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.873432"},
{"rno":538,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.871436"},{"rno":537,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.868447"},
{"rno":536,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.865452"},{"rno":535,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.863457"},
{"rno":524,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.841242"},{"rno":523,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.839247"},
{"rno":522,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.837251"},{"rno":521,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.835296"},
{"rno":520,"replyText":"Reply....","replyer":"replyer...","bno":196,"replyDate":"2021-10-08T11:34:39.832266"}]}

요런식으로 page = ? 을 먹일때마다 값이 바뀌어야 한다.

바뀌면 성공!!!!

끝