

Spring Framework

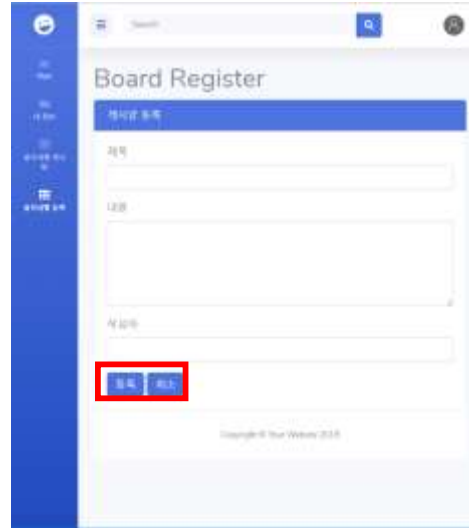
- 게시판
- 1. 구현 순서
- 2. 페이징 처리

1. 게시판 구현 순서



Board List Page showing a table of board entries. The '게시판 목록' button is highlighted in the top right corner.

번호	제목	작성자	작성일	수정일
30	게시판	게시판	2019-04-22 06:01:00	2019-04-22
29	게시판	게시판	2019-04-22 06:01:00	2019-04-22
28	게시판	게시판	2019-04-22 06:01:00	2019-04-22
27	게시판	게시판	2019-04-22 06:01:00	2019-04-22
26	게시판	게시판	2019-04-22 06:01:00	2019-04-22
25	게시판	게시판	2019-04-22 06:01:00	2019-04-22
24	게시판	게시판	2019-04-22 06:01:00	2019-04-22
23	게시판	게시판	2019-04-22 06:01:00	2019-04-22
22	게시판	게시판	2019-04-22 06:01:00	2019-04-22
21	게시판	게시판	2019-04-22 06:01:00	2019-04-22



Board Register Page showing the form for creating a new board entry. The '등록' (Register) button is highlighted.

Board Register

게시판 등록

제목

내용

작성자

등록 | 취소



Board Read Page showing the details of a selected board entry. The '변경' (Modify) button is highlighted.

Board Read Page

상세보기

번호

제목

내용

작성자

변경 | 삭제



Board Modify Page showing the form for editing an existing board entry. The '변경' (Modify) button is highlighted.

Board Modify Page

수정 게시판

번호

제목

내용

작성자

변경 | 등록 | 삭제

1. 게시판 구현 순서

1. 컨트롤러 생성(화면 확인)
 2. 등록 처리
 3. 테이블 생성
 4. DB관련 설정(root-xml작업)
 5. BoardVO 생성(DB컬럼명과 반드시 동일하게 생성)
 6. Service구현
 7. DAO 구현
 8. 마이바티스 DB작업
-
9. 상세보기 처리
 10. 변경 처리
 11. 삭제 처리
 12. 페이징 처리



테이블 SQL

```
create table tbl_board(  
  
    num int auto_increment primary key,  
    title varchar(200) not null,  
    content varchar(2000) not null,  
    writer varchar(50) not null,  
    regdate datetime default current_timestamp,  
    updatedate datetime default current_timestamp  
  
);
```

2. 페이징

1. 반드시 GET 방식으로만 처리한다
2. 이동할 때 페이지 번호, 보여줄 페이지 개수 를 가지고 다녀야 한다
 - > 목록 으로 나올 때 현재 페이지를 유지하기 위해
3. 페이징 처리하는 로직을 클래스로 분류한다
 - > Criteria클래스, PageVO클래스

2. 페이징 (Criteria 클래스)

-페이징을 처리하는 기준

```
public class Criteria {

    private int pageNum; //페이지 번호
    private int count; //몇개의 데이터를 보여줄건가

    public Criteria() {
        this.pageNum = 1;
        this.count = 10;
    }

    public Criteria(int pageNum, int count) {
        super();
        this.pageNum = pageNum;
        this.count = count;
    }

    public int getPageStart() {
        return (pageNum - 1) * count;
    }
}
```

이하 게터 세터 생성

mysql의 limit함수 쿼리문

```
select *
  from tbl_board
 order by num desc
 limit 0, 10; ← 1-10번 데이터가 조회된다
```

```
select *
  from tbl_board
 order by num desc
 limit 10, 20; ← 21-30번 데이터가 조회된다
```

마이바티스의 sql문

```
select *
  from tbl_board
 order by num desc
 limit #{pageStart} , #{count}
```

2. 페이징 (PageVO 클래스)

-페이징 계산 처리 클래스

- 1 total 게시판 글 전체 개수.
- 2 endPage: 게시판을 화면에 보여질 마지막 페이지 번호.
- 3 startPage: 게시판을 화면에 보여질 첫번째 페이지 번호.
- 4 realEnd: 게시판의 실제 마지막 페이지 번호.
- 5 prev: 이전 페이지 버튼 활성화 여부.
- 6 next: 다음 페이지 버튼 활성화 여부.



공식

2 endPage =

1~10 페이지 클릭시 10

11~20 페이지 클릭시 20

21~30 페이지 클릭시 30

3 startPage =

endPage가 10 일때 1

endPage가 20 일때 11

endPage가 30 일때 21

4 realEnd =

게시물이 50개 라면 5

게시물이 51개 라면 6

게시물이 101개 라면 11

공식

4 prev =

startPage가 1이면 비활성화

startPage가 11이면 활성화

5 next =

realEnd가 endPage보다 크면 활성화

Math.ceil() 함수를 이용한다

2. 페이징 (PageVO 클래스)

```
private int startPage; // 화면에 보여지는 시작 페이지 번호
private int endPage; // 화면에 보여지는 끝 페이지 번호
private boolean prev, next; //다음, 이전 버튼
private int total; //총 게시물 수
private Criteria cri; //기준

//생성자
public PageVO(Criteria cri, int total) {
    this.cri = cri; //기준
    this.total = total; //총 게시물 수
    this.endPage = (int)Math.ceil(cri.getPageNum() / (double)10 ) * 10;
    this.startPage = endPage - 10 + 1;
    int realEnd = (int)Math.ceil(total / (double)cri.getCount() );

    if(this.endPage > realEnd ) {
        this.endPage = realEnd;
    }

    this.prev = this.startPage > 1;
    this.next = realEnd > this.endPage;
}
//이후 아래 게터 세터
```

