



[한국ICT인재개발원] 스프링 프레임워크

6. MySQL의 페이징 처리

前) 광고데이터 분석 1년

前) IT강의 경력 2년 6개월

前) 머신러닝을 활용한 데이터 분석 프로젝트반 운영 1년

前) 리그오브 레전드 데이터 분석 등...

現) 국비반 강의 진행중

90972	[일반] 화가 많으면 [4]	toheaven	21/03/19	2049	2
90971	[일반] 남의 밥그릇을 깨기 전에 필요한 고민의 크기 [29]	눈팅만일년	21/03/19	5703	87
90970	[일반] [완전스포] 스나이더컷 2017 버전과 차이점에 중점을 둔 정리 [61]	나주풀	21/03/18	4472	8
90969	[일반] [슬램덩크] 강백호의 점프슛 이야기 [32]	라울리스타	21/03/18	3142	37
90968	[일반] 그 때 너를 붙잡았더라면... [4]	조공플레이	21/03/18	1620	3

목록

이전

다음

글쓰기

1

2

3

4

5

6

7

8







9

10

NEXT 10

페이징 처리는 게시판의 모든 글을 보여주지 않고 n개씩 끊어서 보여주는 처리를 의미합니다.

Oracle sql에서는 복잡한 개념을 몇 가지 더 알아야 하지만, MySQL에서는 비교적 쉽게 처리할 수 있습니다.

Result Grid						
Filter Rows: <input type="text"/>						
Edit:   						
Export/Import:  						
Wrap Cell Content: 						
bno	content	title	writer	regdate	updatedate	
1	테스트 본문	테스트 제목	테스트 글쓴이	2021-03-14 01:55:42	2021-03-14 01:55:42	
4	테스트 본문	테스트 제목	테스트 글쓴이	2021-03-14 01:55:43	2021-03-14 01:55:43	
9	Mock본문	Mock제목	Mock글쓴이	2021-03-16 20:27:56	2021-03-16 20:27:56	
12	42141	412	4124	2021-03-18 21:27:48	2021-03-18 21:27:48	
14	4124	훈련교강사 점수 판정에 대한 기준	15125	2021-03-21 16:05:07	2021-03-21 16:05:07	
15	dsafa	수정했다222	efafew	2021-03-21 16:05:12	2021-03-21 16:05:12	
27	ㅎㅋㄷㅎㅋ	ㅋㄴㄷㄷ	ㅎㅋㄴㄷㅎ	2021-03-21 16:21:50	2021-03-21 16:21:50	
29	테스트 본문	테스트 제목	테스트 글쓴이	2021-03-21 23:27:18	2021-03-21 23:27:18	
30	테스트 본문	테스트 제목	테스트 글쓴이	2021-03-21 23:27:18	2021-03-21 23:27:18	
31	Mock본문	Mock제목	Mock글쓴이	2021-03-21 23:27:18	2021-03-21 23:27:18	

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:	
no	content	title	writer	regdate	updatedate
126	Mock본문	Mock제목	Mock글쓴이	2021-03-21 23:27:20	2021-03-21 23:27:20
127	42141	412	4124	2021-03-21 23:27:20	2021-03-21 23:27:20
128	4124	훈련교강사 점수 판정에 대한 기준	15125	2021-03-21 23:27:20	2021-03-21 23:27:20
129	dsafa	수정했다222	efafew	2021-03-21 23:27:20	2021-03-21 23:27:20
130	ㄱ ㅋ ㄴ ㄷ ㄹ	ㅋ ㄴ ㄷ ㄹ	ㄱ ㅋ ㄴ ㄷ ㄹ	2021-03-21 23:27:20	2021-03-21 23:27:20
131	테스트 본문	테스트 제목	테스트 글쓴이	2021-03-21 23:27:20	2021-03-21 23:27:20
132	테스트 본문	테스트 제목	테스트 글쓴이	2021-03-21 23:27:20	2021-03-21 23:27:20
133	Mock본문	Mock제목	Mock글쓴이	2021-03-21 23:27:20	2021-03-21 23:27:20
134	42141	412	4124	2021-03-21 23:27:20	2021-03-21 23:27:20
135	4124	훈련교강사 점수 판정에 대한 기준	15125	2021-03-21 23:27:20	2021-03-21 23:27:20
NULL	NULL	NULL	NULL	NULL	NULL

1

```
48 • select * from ictboard limit 0, 5;
```

	bno	content	title	writer	regdate	updatedate
▶	1	테스트 본문	테스트 제목	테스트 글쓴이	2021-03-14 01:55:42	2021-03-14 01:55:42
	4	테스트 본문	테스트 제목	테스트 글쓴이	2021-03-14 01:55:43	2021-03-14 01:55:43
	9	Mock본문	Mock제목	Mock글쓴이	2021-03-16 20:27:56	2021-03-16 20:27:56
	12	42141	412	4124	2021-03-18 21:27:48	2021-03-18 21:27:48
	14	4124	훈련강사 점수 판정에 대한 기준	15125	2021-03-21 16:05:07	2021-03-21 16:05:07

```
48 • select * from ictboard limit 5, 5;
```

	bno	content	title	writer	regdate	updatedate
▶	15	dsafa	수정했다222	efafew	2021-03-21 16:05:12	2021-03-21 16:05:12
	27	ㅎㅋㄷㅎㅋ	ㅋㄴㄷㄷ	ㅎㅋㄴㄷㅎ	2021-03-21 16:21:50	2021-03-21 16:21:50
	29	테스트 본문	테스트 제목	테스트 글쓴이	2021-03-21 23:27:18	2021-03-21 23:27:18
	30	테스트 본문	테스트 제목	테스트 글쓴이	2021-03-21 23:27:18	2021-03-21 23:27:18
	31	Mock본문	Mock제목	Mock글쓴이	2021-03-21 23:27:18	2021-03-21 23:27:18

즉 페이징 처리를 할 때 기본적으로
SELECT * FROM 테이블명 limit 시작번호, 보여줄갯수;
구문을 선택한 페이지에 맞게 호출하면 쉽게 구현 가능합니다.

```
@Data
public class Criteria {

    private int page;
    private int number;

    // 페이지 * 페이지당 숫자가 실제 limit구문에 들어갈 시작점이 됩니다.
    // mybatis는 getter를 가져다 쓸 수 있습니다.
    public int getPageStart() {
        return (this.page - 1) * number;
    }

}
```

Limit라는 구문이 새로 생기고, 2개의 정보(시작번호, 출력갯수)를 넘겨야 하기 때문에 Criteria객체가 추가됩니다.

```
<select id="listPage" resultType="org.ict.domain.BoardVO">
  <![CDATA[
    SELECT
      bno, title, content, writer, regdate, updatedate
    FROM
      ictboard
    WHERE bno > 0
    ORDER BY bno DESC, regdate DESC
    limit #{page} -1 * #{number}, #{number}
  ]]>
</select>
```

시작 페이지는 (페이지번호 -1) * 출력갯수
로 계산할 수 있지만 위와 같이 mysql구문에서는 내부적으로 계산구문을
처리할 수 없습니다.


```
// 페이지 * 페이지당 숫자가 실제 limit구문에 들어갈 시작점이 됩니다.  
// mybatis는 getter를 가져다 쓸 수 있습니다.
```

```
public int getPageStart() {  
    return (this.page - 1) * number;  
}
```

```
ORDER BY bno DESC, regdate DESC  
limit #{pageStart}, #{number}  
]]>  
/select>
```

대신 getter를 #{ }로 호출할 수 있다는 특성을 이용해서
위와 같이 getPageStart라는 getter를 호출시 자동으로 시작페이지가
구해지도록 합니다.

bno	title	content	writer	regdate	updatedate
-----	-----	-----	-----	-----	-----
30729	ㅋㅋㅋㅋ	ㅎㅎㅎㅎ	ㅎㅎㅎㅎ	2021-03-21	2021-03-21
30728	수정했다222	dsafa	efafew	2021-03-21	2021-03-21
30727	훈련교강사 점수 판정에 대한 기준	4124	15125	2021-03-21	2021-03-21
30726	412	42141	4124	2021-03-21	2021-03-21
30725	Mock제목	Mock본문	Mock글쓴이	2021-03-21	2021-03-21
30724	테스트 제목	테스트 본문	테스트 글쓴이	2021-03-21	2021-03-21
30723	테스트 제목	테스트 본문	테스트 글쓴이	2021-03-21	2021-03-21
30722	ㅋㅋㅋㅋ	ㅎㅎㅎㅎ	ㅎㅎㅎㅎ	2021-03-21	2021-03-21
30721	수정했다222	dsafa	efafew	2021-03-21	2021-03-21
30720	훈련교강사 점수 판정에 대한 기준	4124	15125	2021-03-21	2021-03-21

역순으로 출력하도록 하는 **Order by DESC** 구문까지 추가한 다음 테스트코드를 작성해 위와 같이 원하는 개수로 끊어지는지 체크해보세요.

글번호	글제목	글쓴이	작성일	수정일
30729	크르드드	ㅎㅋㄴㄷㅎ	2021-03-21	2021-03-21
30728	수정했다222	efafew	2021-03-21	2021-03-21
30727	훈련교강사 점수 판정에 대한 기준	15125	2021-03-21	2021-03-21
30726	412	4124	2021-03-21	2021-03-21
30725	Mock제목	Mock글쓴이	2021-03-21	2021-03-21
30724	테스트 제목	테스트 글쓴이	2021-03-21	2021-03-21
30723	테스트 제목	테스트 글쓴이	2021-03-21	2021-03-21
30722	크르드드	ㅎㅋㄴㄷㅎ	2021-03-21	2021-03-21
30721	수정했다222	efafew	2021-03-21	2021-03-21
30720	훈련교강사 점수 판정에 대한 기준	15125	2021-03-21	2021-03-21

«
1
2
3
4
5
»

글쓰기

이제 게시판 하단의 페이지 번호를 클릭했을때 자동으로 클릭한 페이지로 넘어가도록 설정을 바꿔봅니다.

글번호	글제목	글쓴이	작성일	수정일
30729	크르르르	ㅎㅋㄴㄷㅎ	2021-03-21	2021-03-21
30728	수정했다222	efafew	2021-03-21	2021-03-21
30727	훈련교감사 점수 판정에 대한 기준	15125	2021-03-21	2021-03-21
30726	412	4124	2021-03-21	2021-03-21
30725	Mock제목	Mock글쓴이	2021-03-21	2021-03-21
30724	테스트 제목	테스트 글쓴이	2021-03-21	2021-03-21
30723	테스트 제목	테스트 글쓴이	2021-03-21	2021-03-21
30722	크르르르	ㅎㅋㄴㄷㅎ	2021-03-21	2021-03-21
30721	수정했다222	efafew	2021-03-21	2021-03-21
30720	훈련교감사 점수 판정에 대한 기준	15125	2021-03-21	2021-03-21

«
1
2
3
4
5
»

글쓰기

먼저, `insert into ictboard(content, title, writer) (select * from ictboard);` 구문을 여러 번 실행해 2배씩 들어간 게시물을 늘려놓은 다음 `getList` 대신 `getCriteriaList`라는 새로운 메서드를 만들어주세요. 이 메서드는 위와 같이 글을 전부 호출하지 않고 설정한 개수대로만 출력합니다.

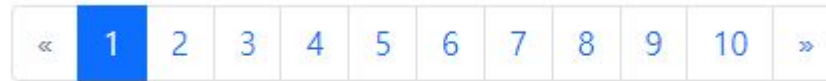
- org.ict.service
 - BoardService.java
 - BoardServiceImpl.java
 - package-info.java
- src/main/resources
- src/test/java
 - org.ict.controller
 - org.ict.mapper
 - BoardMapperTests.java

```
21 |
22 |     public List<BoardVO> getListCriteria(Criteria cri);
23 | }
24 |
```

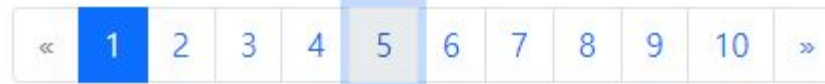
- src/main/java
 - org.ict.controller
 - BoardController.java
 - HomeController.java
 - org.ict.domain
 - BoardVO.java
 - Criteria.java
 - package-info.java
 - org.ict.mapper
 - BoardMapper.java
 - package-info.java
 - BoardMapper.xml
 - org.ict.service
 - BoardService.java
 - BoardServiceImpl.java

```
34 | // void 리턴을 하녀, .addAttribute()를 이용해
35 | // list라는 이름으로 전체 글 목록을 뷰에 전달합니다.
36 | @RequestMapping("/list")
37 | public void list(Model model, Criteria cri) {
38 |
39 |     Log.info("list");
40 |     model.addAttribute("list", service.getListCriteria(cri));
41 | }
```

이를 위해 서비스쪽에 Criteria 정보를 받는 getListCriteria를 정의하시고, 컨트롤러에서도 신설한 getListCriteria를 사용하도록 처리해주세요.



다음으로 화면 하단 표시를 처리합니다.
먼저 화면 하단의 버튼 개수는 10개씩 끊어서 출력할것입니다.



이 페이지네이션은, 10개단위로 실행되기 때문에 시작번호, 끝번호를 구하는것이 먼저입니다.

가령 내가 5페이지를 본다고 해도 시작페이지는 1번, 끝페이지는 10번이어야 합니다.


```
5 @Data
6 public class PageMaker {
7
8     private int totalBoard;
9     private int startPage;
10    private int endPage;
11    private boolean prev;
12    private boolean next;
13
14
15 }
```

이번엔 아래쪽 페이지 이동 버튼 정보를 제공하기 위해

PageMaker 객체를 선언해보겠습니다.

이 객체는 총 페이지 개수를 구하기 위한 전체 글 수 **totalBoard**

화면상 시작페이지를 가능하기 위한 **startPage**

화면상 끝페이지를 가능하기 위한 **endPage**

그리고 다음페이지, 이전페이지 표시여부를 나타낸 **prev, next**를 가집니다.

```
public void calcData() {  
    this.displayPageNum = 10;  
}
```

제일 먼저 내부의 `this.displayPageNum`의 숫자(페이지 버튼을 몇 개 단위로 출력할지)에 대해서 10개로 세팅합니다.

추후 이 숫자만 바꾸면 버튼갯수가 유동적으로 바뀝니다.

```
this.endPage = (int)(Math.ceil(cri.getPage() /  
    (double) displayPageNum) * displayPageNum);
```

먼저 끝나는 지점을 구하고, 그걸 토대로 다시 시작지점을 구하기 위해 끝나는 지점을 구하는 메서드부터 보겠습니다.

끝페이지 = 올림((현재페이지 / 출력페이지, 실수)) * 출력페이지

253페이지, 10개출력 예시(시작 250, 끝 260)

$253 / 10 = 25.3$

25.3을 올리면 => 26

$26 * 10 = 260$

```
this.startPage = (endPage - displayPageNum) + 1;
```

끝나는 지점을 구하면 시작페이지를 구합니다.

시작페이지 = (끝페이지 - 출력페이지개수) + 1;

260페이지 예시 -> $260 - 10 + 1 = 251$ 페이지가 시작점
251 ~ 260페이지가 모두 구해졌음.

```
int tempEndPage = (int)(Math.ceil(totalBoard / (double)cri.getNumber()));  
  
if(endPage > tempEndPage) {  
    endPage = tempEndPage;  
}
```

전체 페이지의 개수는 10으로 떨어지지 않을수도 있기 때문에 다시 계산합니다.

먼저 tempEndPage에 실제 최종페이지를 구한 다음
만약 자동으로 집계되는 endPage보다 값이 크다면
실제 최종페이지로 하락시킵니다.

실제 최종페이지가 753페이지인 경우, 목록상 endPage는 760으로 먼저
집계되므로 만약 실 페이지가 목록상 마지막 페이지보다 작다면
실 페이지로 재지정하는 로직입니다.

```
prev = startPage == 1 ? false : true;
```

이전 페이지는 출력페이지가 1~10인 경우만 아니면 갈 수 있어야 하므로
위와 같이 `startPage` 변수가 1인지 여부로 검사합니다.


```
next = endPage * cri.getNumber() >= totalBoard ? false : true;
```

다음 페이지 여부는 현재 조회중인 페이지 뒤에 더 데이터가 있는지 없는지 여부입니다.

따라서 $\text{endPage} * \text{글 개수}$ 가 현재 전체 글 수보다 적다면 제공합니다.

예를 들어 전체 글 수가 101개인데 현재 10페이지까지 출력중이라면 $10 * 10 = 100$ 까지만 현재 범위 이하에서 표현하는 글 개수이므로 Next버튼이 필요한 식입니다.

```
public void setTotalBoard(int totalBoard) {  
    this.totalBoard = totalBoard;  
  
    calcData();  
}
```

이상의 정보는 페이지를 처리할 때 한꺼번에 받아놔야 화면에 표시할 수 있기 때문에 그냥 전체 글 개수를 조회하는 시점에 다 계산하도록 합니다.

```
public void list(Model model, Criteria cri) {  
  
    Log.info("list");  
    Log.info(cri.toString());  
  
    //model.addAttribute("list", service.getList());  
    model.addAttribute("list",  
        service.getListPage(cri));  
  
    // 페이지네이터 그리기 위해 처리 정보도 같이 전달하기  
    PageMaker pageMaker = new PageMaker();  
    pageMaker.setCri(cri);  
    pageMaker.setTotalCount(131);  
    model.addAttribute("pageMaker", pageMaker);  
}
```

이제 list화면이 그려지기 전에, 하단 버튼정보를 담은 pageMaker 객체까지 같이 전달해야 합니다.

먼저 테스트를 위해 setTotalCount는 임의의 값을 넣어봅니다.

추후 **SELECT count(*) FROM ictboard;** 쿼리를 이용해 실제 데이터를 가져오겠습니다.

```
<c:if test="${pageMaker.prev }">
  <li class="page-item">
    <a class="page-link"
      href="/board/list?page=${pageMaker.startPage -1}
      &laquo;
    </a>
  </li>
</c:if>
```

먼저 뒤로가기 버튼은. `pageMaker`의 `prev` 변수가 `boolean` 타입이기 때문에 그 자체로 `true/false`의 의미를 가지므로 `c:if` 태그의 조건문에 변수째로 넣습니다.

만약 `true`라면 이전 페이지가 존재하므로, 이동할 링크를 걸어주시면 됩니다.

```
c:forEach begin="${pageMaker.startPage }"  
            end  ="${pageMaker.endPage }"  
            var="idx">  
    <li class="page-item"  
        <c:out value="${pageMaker.cri.page == idx ? 'active' : '' }" />" />  
        <a class="page-link"  
            href="/board/list?page=${idx }">${idx }</a></li>  
::forEach>
```

다음은 숫자 버튼으로, **c:forEach**구문의 **begin**, **end** 속성을 이용해 반복횟수를 정하고, **var**의 **idx**숫자를 이용해 표기합니다.
li 태그의 **class** 내에 **active** 여부를 출력하기 위해 삼항연산자를 이용하고 링크주소도 페이지를 선택한 숫자 번호로 이동하게 세팅합니다.

```
<c:if test="${pageMaker.next && pageMaker.endPage > 0}">
  <li class="page-item">
    <a class="page-link"
      href="/board/list?page=${pageMaker.endPage +1}"
      &raquo;
    </a>
  </li>
</c:if>
```

마지막으로 다음 페이지네이션 목록 버튼입니다.
end페이지가 0이라면 당연히 출력할 필요가 없으니 0보다 큰 상태에서,
next가 참이어야 출력하는것입니다.
href를 이용해 역시 endPage로 지목된 마지막 번호 +1을 목표지로 잡습니다.

글번호	글제목	글쓴이	작성일	수정일
53126	Mock테스트제목수정	Mock글쓴이	2021-03-22	2021-03-22
53125	java is fun22	spring	2021-03-22	2021-03-22
53124	Modify반영제목	insert글쓴이	2021-03-22	2021-03-22
53123	수정된제목	insert글쓴이	2021-03-22	2021-03-22
53122	자바스크립트로 수정했다	íïïë,ïíïë,ï	2021-03-22	2021-03-22
53121	Mock테스트제목수정	Mock글쓴이	2021-03-22	2021-03-22
53120	java is fun22	spring	2021-03-22	2021-03-22
53119	Modify반영제목	insert글쓴이	2021-03-22	2021-03-22
53118	수정된제목	insert글쓴이	2021-03-22	2021-03-22
53117	자바스크립트로 수정했다	íïïë,ïíïë,ï	2021-03-22	2021-03-22

처리 완료시 위와 같이 전체 버튼목록도 맞게 나오고
선택한 페이지만 **active** 가 추가로 붙어서 색이 다르게 강조됩니다.
클릭시 이동까지 잘 되는지 확인해주세요.

```
public int countPageNum(Criteria cri);
```

```
<select id="countPageNum" resultType="int">
    <![CDATA[
        SELECT COUNT(bno) FROM ictboard WHERE bno > 0
    ]]>
</select>
```

```
public int getCountPage(Criteria cri);
```

```
@Override
public int getCountPage(Criteria cri) {
    return mapper.countPageNum(cri);
}
```

이제 그러면 **totalBoard** 변수에 정확한 값을 전달하기 위해서 쿼리문을 **DB**에 날려줄 필요가 있습니다.
전체 글 개수를 구하는 **SELECT COUNT(bno) * from ictboard;** 쿼리를 사용할 수 있도록 **mapper, service**쪽에 세팅합니다.

```
// 페이지네이터 그리기 위해 처리 정보도 같이 전달하기
PageMaker pageMaker = new PageMaker();
pageMaker.setCri(cri);
//pageMaker.setTotalCount(131);
pageMaker.setTotalCount(service.getCountPage(cri));
model.addAttribute("pageMaker", pageMaker);
Log.info(pageMaker.toString());
```

컨트롤러쪽에서 이제 이 서비스기능을 이용해 전체 개수를 받아오도록하면
완성됩니다.

53062번 글 내용

19페이지 글

글쓴이

í00ê,0í00ê,0

글 제목

자바스크립트로 수정했다

본문

자바스크립트 자바스크립트

등록날짜

2021-03-22

수정날짜

2021-03-22

수정

삭제

목록

고영원세크

자바스크립트로 수정했다

1

2

현재는 몇 페이지에서 목록 버튼을 눌러도 1페이지로 돌아갑니다.

그러나 이런 경우는 매우 불편하기 때문에 추가적으로 해당되는 페이지로 돌아가도록 파라미터를 추가 제공할 것입니다.

```
//model.addAttribute("list", service.getList())
model.addAttribute("list",
    service.getListPage(cri));
model.addAttribute("page", cri.getPage());
// 페이지네이터 그리기 위해 처리 정보도 같이 전달하기
PageMaker pageMaker = new PageMaker();
pageMaker.setCri(cri);
//pageMaker.setTotalCount(131);

forEach var="board" items="${list }">
<tr>
    <td>${board.bno }</td>
    <td><a href="/board/get?bno=${board.bno}&page=${page}">
        ${board.title }</a></td>
    <td>${board.writer }</td>
    <td>${board.regDate }</td>
    <td>${board.updateDate }</td>
</tr>
```

제일 먼저, 컨트롤러에서 현재 페이지의 정보까지 함께 뷰(.jsp)로 보내준 다음 그 정보를 이용해 링크주소가 페이지 번호를 포함하도록 합니다.

```
class="btn btn-danger" href="/board/list?cri.page=${cri.page }">목록</a>
```

다음 디테일 페이지에서도 목록버튼 링크에 추가로 page정보가 붙도록 한 다음, 목록버튼을 눌러 작동여부를 확인해주세요.


```
// 그 처리 메서드에 bno를 리턴 프로젝트예요.  
@GetMapping("/get")  
public void get(Long bno, Criteria cri, Model model){  
    model.addAttribute("cri", cri);  
    model.addAttribute("board", service.get(bno));  
}
```

```
<input type="hidden" name="page" value="${cri.page }">
```

먼저, 수정하는 디테일페이지 `get.jsp`가 위와 같이 페이지 정보를 함께 보내도록 `get`메서드가 `Criteria`를 받도록 처리한 다음 `form`태그 내에 `hidden`태그로 숨겨둡니다

```
@PostMapping("/remove")
public String remove(Long bno,
                     Criteria cri,
                     RedirectAttributes rttr, Model model) {

    service.remove(bno);
    // 추후 삭제 완료시 XX번 글이 삭제되었습니다라는
    // 팝업을 띄우기 위해 미리 세팅
    rttr.addFlashAttribute("bno", bno);
    //model.addAttribute("bno", bno);

    rttr.addAttribute("page", cri.getPage());
    // 삭제된 글의 디테일페이지는 존재하지 않으므로 리스트로 이동
    return "redirect:/board/list";
}
```

다음, 삭제로직이 page정보를 받도록 Criteria를 선언해주시고 컨트롤러단에서는 위와 같이 리다이렉트 전에 page정보를 먼저 url에 전달할 수 있도록 처리해주면 됩니다.

```
public String modify(Model model, Criteria cri,

    // 계획
    // 1. 상세 글 정보를 저장합니다.
    BoardVO board = service.get(bno);
    // 2. 이 정보를 view 파일로 전송합니다.
    model.addAttribute("board", board);
    model.addAttribute("cri", cri);
    return "/board/modify";
```

```
<input type="hidden" name="page" value="${cri.page }">
```

역시, 수정페이지인 `modify.jsp`가 위와 같이 페이지 정보를 함께 보내도록 `get`메서드가 `Criteria`를 받도록 처리한 다음 `form`태그 내에 `hidden`태그로 숨겨둡니다

```
@PostMapping("/modifyrun")
public String modify(BoardVO board,
                    Criteria cri,
                    RedirectAttributes rttr) {
    // 넘겨받은 글 정보를 갱신 등록
    service.modify(board);

    // 수정된 글 번호 정보를 저장
    rttr.addFlashAttribute("bno", board.getBno());

    rttr.addAttribute("page", cri.getPage());

    // 디테일 페이지로 넘어가기 위해 redirect 주소 설정
    return "redirect:/board/get?bno=" + board.getBno();
}
```

다음 수정실행 창에서도 page를 전달받았을텐데, 이를 그대로 rttr.addAttribute()를 이용해 전달해주시면 됩니다.