[한국ICT인재개발원] 스프링 프레임워크

7. 게시판 검색기능 추가하기

강의자 소개 : 채종훈 강사

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

검색기능 추가

요즘 커뮤니티 게시판에서는 대체로 검색기능을 제공합니다.
검색기능은 주로 글쓴이, 제목, 본문내용에서 찾고싶은 내용을
검색창에 입력하면, 해당 내용이 존재하는 글들로만 다시 정렬하는것을
의미합니다.

검색기능 추가

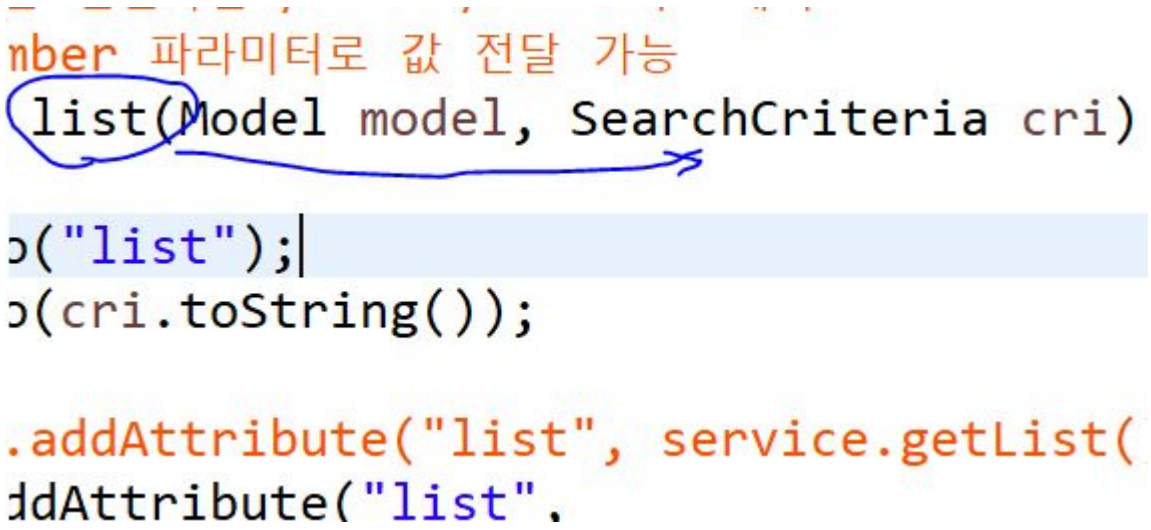
최대 검색조건을 6개로 해서 작성해보겠습니다.
1. 글쓴이만으로 조회, 2. 제목만으로 조회, 3. 본문만으로 조회
4. 제목+본문으로 조회, 5. 글쓴이 + 본문으로 조회, 6. 글쓴이+제목+본문조회

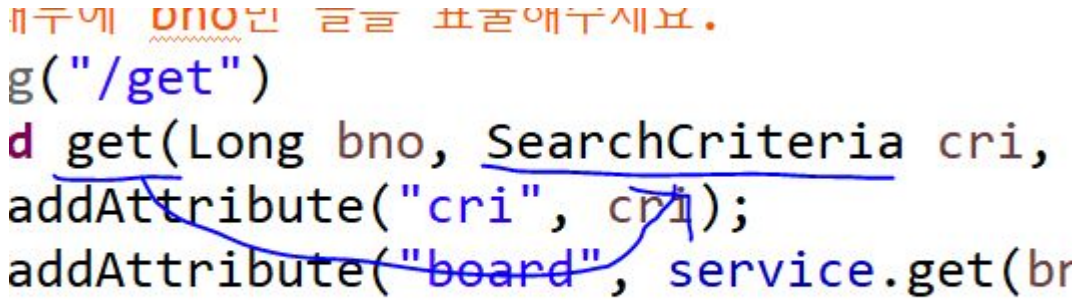
검색기능을 위한 Criteria


```java
package org.ict.domain;

import lombok.Data;

@Data
public class SearchCriteria extends Criteria {

    private String searchType;
    private String keyword;
}
```

먼저 검색조건도 같이 넘겨야 하기 때문에, 검색에 대한 정보를 포함시켜서 전송할 **SearchCriteria**를 새로 생성한 뒤 **Criteria**를 상속해서 사용합니다.
이 객체는 조회타입(**searchType**), 조회키워드(**keyword**)를 전달합니다.

검색기능을 위한 Criteria





컨트롤러쪽에서도 **Criteria**를 사용하는 모든 파라미터를 **SearchCriteria**로 교체합니다.

list.jsp의 변경


```
<div class="container">
  <div class="row">
      <h1 class="text-primary text-center">전체 글 목록</h1>
  </div>
  <div class="row">
      <div class="box-body">

          <select name="searchType">
              <option value="n"
              <c:out value="${cri.searchType == null ? 'selected' :'' }"/>>
              -
              </option>
              <option value="t"
              <c:out value="${cri.searchType eq 't' ? 'selected' :'' }"/>>
              제목
              </option>
```

다음으로 검색창을 만들기 위해 검색조건을 **list.jsp**에 추가해줍니다.
<select> 태그 내부의 **<option>**을 이용해 검색조건을 명문화합니다.

list.jsp의 변경

값	의미
n	검색조건 없이 리스팅
t	제목에 포함시 리스팅
c	본문내용에 검색단어 포함시 리스팅
w	글쓴이 일치시 리스팅
tc	제목 + 본문내용에 포함시 리스팅
cw	본문내용에 포함과 글쓴이 일치시 리스팅
tcw	상기 모든 내용이 일치시 리스팅

위와 같이 7개의 option태그가 필요합니다. 주의하실점은 ${}구문 내에서 문자열의 비교는 문자열1 eq 문자열2 로 합니다.

list.jsp의 변경


```
  </select>

  <input type="text"
       name="keyword"
       id="keywordInput"
       value="${cri.keyword }">
  <button id="searchBtn">Search</button>

'div>
```

select태그 바로 뒤에 검색어 입력창과 검색실행버튼을 배치합니다.

검색 키워드 전달하기


```
eMaker.cri.page == idx ? 'active' : '' }" />">

page=${idx }&searchType=${cri.searchType}&keyword=${cri.keyword}">

t && pageMaker.endPage > 0 }">

"
```

검색후 다음 페이지로 넘어가도 검색이 유지되어야 하기 때문에 이를 위해
검색단어를 페이지 이동간에도 전달할 수 있도록 수정합니다.
이 작업은 뒤로가기, 앞으로 가기에도 적용해야 합니다(중요)

검색 키워드 전달하기

f12를 눌러 나오는 개발자도구창을 보시면 위와 같이 파라미터가 추가로 붙어야 합니다.

검색 버튼 동작시키기


```
// 검색버튼 작동
$('#searchBtn').on("click", function(event){

    self.location = "list"
        + "?page=1"
        + "&searchType="
        + $("select option:selected").val()
        +"&keyword=" + $("#keywordInput").val();
})
```

검색버튼 동작시키기는 자바스크립트로 작동시키겠습니다.
위와같이 **self.location**을 사용하며, 주소는 문자열로 위와같이 파라미터 단위로 연결해 사용합니다.

검색 버튼 동작시키기

수정 후 검색버튼을 클릭했을때 위와같이 검색타입과 검색어가 잘 전달되는지 체크해주세요.

조건별 SQL구문(동적SQL구문)

기능	사용예시	설명
if	<if test=”title != null”> AND title like #{title} </if>	조건적으로 쿼리문을 추가하고싶을때
choose, when, otherwise	<choose> <when test=”title != null”> </when> <when test=”author != null”> </when>.....	switch~case문처럼 사용
trim, where, set	<trim prefix=”WHERE” prefixOverrides=”AND|OR”> </trim>	구문 변경시 사용
foreach	<foreach item=”item” index=”intex” collection=”list”> </foreach>	반복문 실행시 사용

검색기능 추가시 7개의 모드가 있으므로 원래대로라면 7개의 쿼리문을 따로따로 처리해야합니다.
그러나 쿼리문의 내용이 거의 동일할것이기 때문에 조건별로 일부 쿼리문만 수정되어 돌아가는 동적 SQL구문을 짜서 쓰겠습니다.

조건별 SQL구문(동적SQL구문)


```
public List<BoardVO> getListPage(SearchCriteria cri)
    return mapper.listPage(cri);
}

public int getCountPage(SearchCriteria cri) {
    return mapper.countPageNum(cri);
}
```

SQL작성에 앞서 먼저 Mapper, Service쪽 모든 Criteria 사용처를 전부 SearchCriteria로 변경합니다.

조건별 SQL구문(동적SQL구문)


```
<select id="listPage" resultType="org.ict.domain.Bo
    <!-- 아래 <![CDATA[ 는 닫는부분인 ]]> 사이의 모든 문자를
    쿼리문으로만 인식하고 태그요소로 인식하지 않게 함. -->
    <![CDATA[
        SELECT bno, title, content, writer, regdate
            FROM ictboard
            WHERE bno > 0
    ]]>
        <!-- 이 부분에 동적쿼리가 들어갑니다. -->

    <![CDATA[
        ORDER BY bno DESC
            limit #{pageStart}, #{number}
    ]]>
</select>
```

검색어 없이 접속했을때도 전체 글을 가져오는 구문은 작동해야하고,
검색어가 추가되었을때만 추가된 검색어에 맞게 구문 추가가 이루어져야 합니다.

조건별 SQL구문(동적SQL구문)


```
<if test="searchType != null">
    <if test="searchType == 't'.toString()">
        and title like CONCAT('%', #{keyword}, '%')
    </if>
</if>
```

위와 같이 **mapper xml**에서도 **c:if** 와 유사하게 사용할 수 있습니다.
그러나 문자열 비교는 **eq**로 할 수 없으며 위와같이 **toString()** 메서드를 호출해서 대신하는 점에 주의해주세요.
like CONCAT('%', #{keyword}, '%') 구문은
앞 뒤 문자열 상관없이 **#{keyword}**에 해당하는 구간이 있다면 모두 조건으로 간주합니다.

조건별 SQL구문(동적SQL구문)


```
<if test="searchType == 'tc'.toString()">
    and (title like CONCAT('%', #{keyword}, '%')
        OR content like CONCAT('%', #{keyword}, '%'))
</if>
<if test="searchType == 'cw'.toString()">
    and (content like CONCAT('%', #{keyword}, '%')
        OR writer like CONCAT('%', #{keyword}, '%'))
</if>
<if test="searchType == 'tcw'.toString()">
    and ( title like CONCAT('%', #{keyword}, '%')
        OR
        content like CONCAT('%', #{keyword}, '%')
        OR
        writer like CONCAT('%', #{keyword}, '%')
        )
</if>
```

그리고 2개 이상의 like 조건절은 ()로 묶어서 처리합니다.
tc, cw, tcw를 모두 처리해주세요.

조건별 SQL구문(동적SQL구문)


```
<if test="searchType != null">
    <if test="searchType == 't'.toString()">
        and title like CONCAT('%', #{keyword}, '%')
    </if>
    <if test="searchType == 'c'.toString()">
        and content like CONCAT('%', #{keyword}, '%')
    </if>
    <if test="searchType == 'w'.toString()">
        and writer like CONCAT('%', #{keyword}, '%')
    </if>
    <if test="searchType == 'tc'.toString()">
        and (title like CONCAT('%', #{keyword}, '%')
            OR content like CONCAT('%', #{keyword}, '%'))
    </if>
    <if test="searchType == 'cw'.toString()">
        and (content like CONCAT('%', #{keyword}, '%')
            OR writer like CONCAT('%', #{keyword}, '%'))
    </if>
    <if test="searchType == 'tcw'.toString()">
        and ( title like CONCAT('%', #{keyword}, '%')
            OR
            content like CONCAT('%', #{keyword}, '%')
            OR
            writer like CONCAT('%', #{keyword}, '%')
            )
    </if>
</if>
```

여기서 단순히 **listPage**만 바꾸는것은 의미가 없습니다.
totalBoard 변수에 글 갯수도 변동이 생기기 때문에 같이 바뀌어야 합니다.
그러나 전체 구문을 다 집어넣기엔 두 번 이상 작성하는 부담이 생깁니다.

조건별 SQL구문(동적SQL구문)


```
<sql id="search">
    <if test="searchType != null">
        <if test="searchType == 't'.toString()">
            and title like CONCAT('%', #{keyword}, '%')
        </if>
        <if test="searchType == 'c'.toString()">
            and content like CONCAT('%', #{keyword}, '%')
        </if>
        <if test="searchType == 'w'.toString()">
            and writer like CONCAT('%', #{keyword}, '%')
        </if>
        <if test="searchType == 'tc'.toString()">
            and (title like CONCAT('%', #{keyword}, '%')
                OR content like CONCAT('%', #{keyword}, '%'))
        </if>
        <if test="searchType == 'cw'.toString()">
            and (content like CONCAT('%', #{keyword}, '%')
                OR writer like CONCAT('%', #{keyword}, '%'))
        </if>
        <if test="searchType == 'tcw'.toString()">
            and ( title like CONCAT('%', #{keyword}, '%')
                OR
                content like CONCAT('%', #{keyword}, '%')
                OR
                writer like CONCAT('%', #{keyword}, '%')
                )
        </if>
    </if>
</sql>
```

특정 구문 부분을 변수처럼 저장할 수 있는 **<sql>** 태그와
그 **sql**태그를 변수처럼 호출할 수 있는 **<include>** 태그를 써보겠습니다.

먼저 조건절을 바깥으로 빼서 **<sql>** 태그에 넣고 **id**속성을 **“search”**로 줍니다.

조건별 SQL구문(동적SQL구문)


```
<select id="listPage" resultType="org.ict.domain.BoardVO">
    <!-- 아래 <![CDATA[ 는 닫는부분인 ]]> 사이의 모든 문자를
    쿼리문으로만 인식하고 태그요소로 인식하지 않게 함. -->
    <![CDATA[
        SELECT bno, title, content, writer, regdate, updatedate
            FROM ictboard
            WHERE bno > 0
    ]]>
        <include refid="search"></include>
    <![CDATA[
        ORDER BY bno DESC
            limit #{pageStart}, #{number}
    ]]>
</select>
```

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

다음, 필요한 부분에 **<include refid=”search”></include>**
와 같이 **refid**속성에 **<sql>**태그에서 저장한 명칭을 집어넣으면
변수 내 자료가 호출되듯 쿼리문이 들어갑니다.

동적 SQL문이 적용된 검색 게시판


```
<td>${board.bno }</td>
<td><a href="/board/get?bno=${board.bno}&page=${cri.page}&searchType=${cri.searchType}&keyword=${cri.keyword}">
      ${board.title }</a></td>
<td>${board.writer }</td>
<td>${board.regDate }</td>
<td>${board.updateDate }</td>
```

먼저 **list.jsp**부터 수정하겠습니다.
글 제목부분에도 이제 **searchType**과 **keyword**가 전달되게 해 주세요.

동적 SQL문이 적용된 검색 게시판


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

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

get.jsp가 역시 searchType, keyword정보를 전달하게 해 주세요.

이미 SearchCriteria로 변경했기때문에 전달은 문제없습니다.

리스트 페이지로 돌아가는 링크 역시 같은 정보를 반영하도록 해 주세요.

동적 SQL문이 적용된 검색 게시판


```
수정날짜<input name="updateDate" type="text" class="form-control"
    readonly=true value=${board.updateDate }><br>
<input type="hidden" name="bno" value=${board.bno } />
<input type="hidden" name="page" value=${cri.page } />
<input type="hidden" name="searchType" value=${cri.searchType } />
<input type="hidden" name="keyword" value=${cri.keyword } />
```

modify.jsp 역시 그대로 진행해줍니다.

동적 SQL문이 적용된 검색 게시판


```
rttr.addAttribute("searchType", cri.getSearchType());
rttr.addAttribute("keyword", cri.getKeyword());
```

그리고 컨트롤러쪽에서도, **Post**방식을 처리하는 메서드마다
Criteria로 **page**정보를 넣어줬듯 똑같이 처리합니다.