

10장. 페이징, 검색 및 정렬



목 차

1

게시판페이징

2

검색기능

3

정렬기능

페이징 기능

- index 함수에 페이징 기능 추가

```
from django.core.paginator import Paginator
```

```
def index(request):
```

```
    # 질문 목록
```

```
    # 페이지 - localhost:8000/pybo/?page=1
```

```
    page = request.GET.get('page', '1')
```

```
    # 조회
```

```
    question_list = Question.objects.order_by('-create_date')
```

```
    # 페이징 처리 : 페이지당 10개씩 보여주기
```

```
    paginator = Paginator(question_list, 10)
```

```
    page_obj = paginator.get_page(page)
```

페이징 기능

- **Paginator 클래스**

- `get('page[, '1']`에서 '1'은 page 파라미터가 없는 URL을 위해 기본값으로 1을 지정한 것이다. 다른 페이지는 `/pybo/?page=2` 형태로 표시됨
- Paginator 클래스는 `question_list`를 페이징 객체 `paginator`로 변환한다. 두번째 파라미터인 10은 페이지당 보여줄 게시물 개수를 의미한다.
`page_obj = paginator.get_page(page)`로 만들어진 `page_obj` 객체의 속성은 표를 참조,

페이징 기능

▪ Paginator 클래스

항목	설명
paginator.count	전체 게시물수
paginator.per_page	페이지당 보여줄 게시물 개수
paginator.page_range	페이지 범위
number	현재 페이지 번호
previous_page_number	이전 페이지 번호
next.page_number	다음 페이지 번호
has_previous	이전 페이지 유무
has_next	다음 페이지 유무
start_index	현재 페이지 시작 인덱스(1부터 시작)
end_index	현재 페이지의 끝 인덱스(1부터 시작)

부트스트랩 적용

▪ Componets > Pagination



```
<nav aria-label="...">
  <ul class="pagination">
    <li class="page-item disabled">
      <a class="page-link" href="#" tabindex="-1" aria-disabled="true">Previous</a>
    </li>
    <li class="page-item"><a class="page-link" href="#">1</a></li>
    <li class="page-item active" aria-current="page">
      <a class="page-link" href="#">2</a>
    </li>
    <li class="page-item"><a class="page-link" href="#">3</a></li>
    <li class="page-item">
      <a class="page-link" href="#">Next</a>
    </li>
  </ul>
</nav>
```

[Copy](#)

페이징 기능

▪ 페이지 번호 출력하기

(..생략..)

</table>

<!--페이징 처리 시작-->

<ul class="pagination justify-content-center">

<!-- 페이지 리스트-->

{% for page_number in question_list.paginator.page_range %}

<li class="page-item">

{{ page_number }}

{% endfor %}

페이지징 기능

■ 페이지 번호 출력하기 – 현재 페이지 활성화

```
<ul class="pagination justify-content-center">
  <!-- 페이지 리스트 -->
  {% for page_number in question_list.paginator.page_range %}
    {% if page_number == question_list.number %}
      <li class="page-item active" aria-current="page">
        <a class="page-link" href="?page={{ page_number }}">
          {{ page_number }}
        </a>
      </li>
    {% else %}
      <li class="page-item">
        <a class="page-link" href="?page={{ page_number }}">
          {{ page_number }}
        </a>
      </li>
    {% endif %}
  {% endfor %}
</ul>
```


페이지징 기능

← → ↻ ⓘ 127.0.0.1:8000/pybo/?page=1

Pybo giyong(로그아웃)

번호	제목	글쓴이
1	코로나 백신	giyong

← → ↻ ⓘ 127.0.0.1:8000/pybo/?page=2

Pybo giyong(로그아웃)

번호	제목	글쓴이
1	아이스커피	colli
2	장미	colli
9	아이스커피	colli
10	아이스커피	colli

질문 등록하기

1 2 3 4

페이지징 기능

■ 이전 페이지 구현

<!-- 페이지징 처리 시작-->

```
<ul class="pagination justify-content-center">
```

<!-- 이전 페이지-->

```
{% if question_list.has_previous %}
```

```
<li class="page-item">
```

```
<a class="page-link"
```

```
href="?page={{ question_list.previous_page_number }}">이전
```

```
</a>
```

```
</li>
```

```
{% else %} <!-- 이전 페이지가 없으면 비활성화 -->
```

```
<li class="page-item disabled">
```

```
<a class="page-link" tabindex="-1" aria-disabled="true" href="#">
```

이전

```
</a>
```

```
</li>
```

```
{% endif %}
```

<!-- 페이지 리스트-->

이전

1

2

3

4

다음

페이지징 기능

▪ 다음 페이지 구현

이전 1 2 3 4 다음

```
<!-- 다음 페이지-->
{% if question_list.has_next %}
<li class="page-item">
  <a class="page-link" href="?page={{ question_list.next_page_number }}" >
    다음
  </a>
</li>
{% else %}
<li class="page-item disabled">
  <a class="page-link" tabindex="-1" aria-disabled="true" href="#">
    다음
  </a>
</li>
{% endif %}
</ul>
<!-- 페이지징 처리 끝-->
```

게시물 번호

■ 항상 1로 시작하는 게시물 번호 해결

게시물 번호 공식 만들기

일련번호 = 전체 게시물수 - 시작 인덱스 - 현재 인덱스 + 1

전체 게시물수가 12개이고 페이지당 10건씩 게시물을 보여 준다면

- 1페이지의 일련번호는 $12 - 1 - (0\sim9\text{반복}) + 1 \rightarrow 12\sim3$ 까지 표시
- 2 페이지의 일련번호는 $12 - 11 - (0\sim9\text{반복}) + 1 \rightarrow 2\sim1$ 이 표시

템플릿에서 이 공식을 적용하려면 빼기 기능이 필요하다.

더하기 필터 (`|add5`)는 있지만 (`|sub:3`)과 같은 빼기 필터가 없으므로
빼기 필터를 직접 만들어야 한다.

템플릿 필터 직접 만들기

- 템플릿 필터 : 템플릿 태그에서 | 문자 뒤에 사용하는 코드를 말한다.

예) none 대신 공백 문자열로 보여주기 – |default_if_none: ' '

1. 템플릿 필터 디렉터리 만들기

```
(mysite) C:\projects\misite\pybo> mkdir templatetags
```

2. 템플릿 필터 작성하기

```
from django import template
```

```
register = template.Library()
```

```
@register.filter
```

```
def sub(value, arg):
```

```
    return value - arg
```

#기존값 value에서 입력으
로 받은값 arg,를 빼서 반환

템플릿 필터 직접 만들기

3. 템플릿 필터 사용하기 – pybo/question_list.html

파일 위쪽에 {% load pybo_filter %}을 로드해야 한다.

{{ forloop.counter }} 대신에 필터 입력

```
{% extends 'base.html' %}
```

```
{% load pybo_filter %}
```

```
<td>{{ question_list.paginator.count|sub:question_list.start_index|sub:forloop.counter0|add:1 }}</td>
```

공식 요소	설명
question_list.paginator.count	전체 게시물수
question_list.start_index	시작 인덱스(1부터 시작)
forloop.counter0	루프 내의 현재 인덱스(forloop.counter0은 0부터 forloop.counter는 1부터 시작)

템플릿 필터 직접 만들기

▪ 항상 1로 시작하는 게시물 번호 해결

번호	제목
31	냉방 온도
30	냉방 온도
29	냉방 온도
28	냉방 온도
27	냉방 온도
26	냉방 온도
25	냉방 온도
24	냉방 온도
23	냉방 온도
22	냉방 온도

이전 1 2 3 4

질문 등록하기

번호	제목	글쓴이
21	코로나 백신	giyong
20	코로나 백신	giyong
19	코로나 백신	giyong
18	코로나 백신	giyong
17	코로나 백신	giyong
16	코로나 백신	giyong
15	코로나 백신	giyong
14	아이스커피	coli
13	아이스커피	coli
12	아이스커피	coli

이전 1 2 3 4 다음

질문 등록하기

답변 개수 표시하기

■ 질문에 달린 답변 개수 표시하기

번호	추천	제목	글쓴이	작성일시
2	2	장고로 만든 유명한 사이트가 있나요? [2]	admin	2021년 10월 11일 7:09 오전
1	1	pybo란 무엇인가요 [2]	admin	2021년 10월 11일 6:44 오전

[이전](#) [1](#) [2](#) [3](#) [다음](#)

```
<td class="text-left">
  <a href = "{% url 'pybo:detail' question.id %}">
    {{ question.subject }}
  </a>
  {% if question.answer_set.count > 0 %}
  <span class="text-danger small ml-2">
    [{{ question.answer_set.count }}]
  </span>
  {% endif %}
</td>
<td>{{ question.author.username }}</td>
<td>{{ question.create_date }}</td>
```


검색 기능 추가하기

■ 검색 기능

검색 대상은 '제목', '질문내용', '질문 글쓴이', '답변 글쓴이'

			휴머노이드	찾기
번호	제목	글쓴이	작성일시	
2	휴머노이드 1	today	2021년 7월 10일 6:07 오후	
1	천 개의 파랑 1	admin	2021년 7월 10일 6:41 오전	
		이전 1 다음		

			today	찾기
번호	제목	글쓴이	작성일시	
12	테스트 댓글 작업	today	2021년 7월 11일 5:54 오전	
11	테스트	today	2021년 7월 11일 5:54 오전	

검색 기능 추가하기

1. 질문 목록 화면에 검색창 추가하기

```
{% block content %}  
<div class="container my-3">  
  <div class="row justify-content-end my-3">  
    <div class="col-4 input-group">  
      <input type="text" class="form-control kw"  
        value="{{ kw|default_if_none:'' }}">  
      <div class="input-group-append">  
        <button class="btn btn-outline-secondary"  
          type="button" id="btn_search">찾기</button>  
      </div>  
    </div>  
  </div>  
</div>  
<table class="table">
```

input 엘리먼트의 class 속성에 kw은 자바스크립트 검색 창에 입력된 값을 읽을 수 있도록 설정한 것이다.

검색 기능 추가하기

2. 질문 목록에 검색 폼 추가 - 맨 아래에 작성

```
<!-- 페이징 처리 끝-->
<a href="{% url 'pybo:question_create' %}" class="btn btn-primary">질문 등록하기</a>
</div>
<!-- 검색 폼(hidden으로 전달) -->
<form id="searchForm" method="get" action="{% url 'pybo:board' %}">
  <input type="hidden" name="kw" id="kw" value="{{ kw|default_if_none:'' }}">
  <input type="hidden" name="page" id="page" value="{{ page }}">
</form>
{% endblock %}
```

page와 kw를 동시에 GET 방식으로 요청할 수 있도록 form을 작성함.

kw와 page는 이전에 요청했던 값을 기억해야 하므로 value 속성에 대입하고,
질문 목록 함수(board())에서 전달받는다.

hidden 속성은 페이지에 보이지 않고 정보만 전달하는 기능을 함

검색 기능 추가하기

3. 질문 목록의 페이징 코드 수정

(..생략..)

<!--이전 페이지-->

{% if question_list.has_previous %}

<li class="page-item">

<a class="page-link" href="#"

data-page="{{ question_list.previous_page_number }}">이전

{% else %}

기존 코드를 data-page 속성으로 바꾼어서 자바스크립트제이쿼리로 처리할 수 있도록 한다.

검색 기능 추가하기

(..생략..)

<!--페이지 리스트-->

{% if page_number == question_list.number %}

<li class="page-item active" aria-current="page">

{{ page_number }}

{% else %}

<li class="page-item">

{{ page_number }}

검색 기능 추가하기

(..생략..)

<!--다음 페이지-->

{% if question_list.has_next %}

<li class="page-item">

<a class="page-link"

data-page="{{ question_list.next_page_number }}" href="#">다음

{% else %}

검색 기능 추가하기

4. 질문 목록의 페이징 – 제이쿼리로 구현

(..생략..)

```
{% endblock %}
```

```
{% block script %}
```

```
<script>
```

```
$(function(){
```

```
    $(".page-link").on('click', function(){ //페이지 번호를 클릭하면
```

```
        $("#page").val($(this).data("page")); //data-page의 page값 id가 page인 필드에 설정
```

```
        $("#searchForm").submit(); //폼을 전송
```

```
    });
```

```
});
```

검색 기능 추가하기

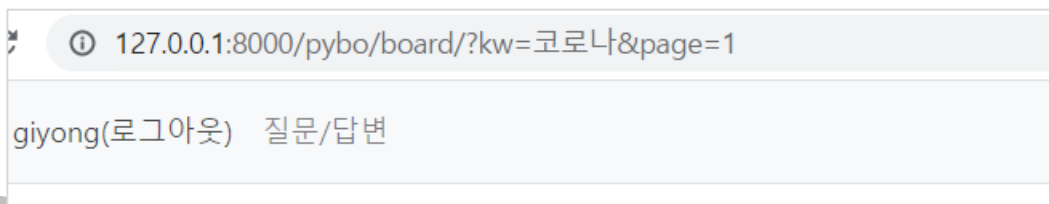
4. 질문 목록의 페이징 – 제이쿼리로 구현

```
$("#btn_search").on('click', function(){  
    $("#kw").val($(".kw").val()); //입력된 kw값을 id가 kw인 필드에 설정  
    $("#page").val(1);           //1페이지부터 검색  
    $("#searchForm").submit();   //폼을 전송  
});
```

`</script>`

`{% endblock %}`

>>코로나로 검색



127.0.0.1:8000/pybo/board/?kw=코로나&page=1

giyong(로그아웃) 질문/답변

검색 기능 추가하기

5. board 함수 수정하기 – pybo/views.py

```
from django.db.models import Q

page = request.GET.get('page', '1')
kw = request.GET.get('kw', '') # 검색어
# 조회
question_list = Question.objects.order_by('-create_date')
if kw:
    question_list = question_list.filter(
        Q(subject__icontains=kw) |    # 제목 검색(대소문자 허용)
        Q(content__icontains=kw) |    # 내용 검색
        Q(author__username__icontains=kw) |    # 질문 글쓴이
        Q(answer__author__username__icontains=kw) # 답변 글쓴이
    ).distinct() # 중복 제거
```

검색 기능 추가하기

5. board 함수 수정하기 – pybo/views.py

```
# 페이징 처리 : 페이지당 10개씩 보여주기
paginator = Paginator(question_list, 10)
page_obj = paginator.get_page(page)

context = {'question_list' : page_obj, 'page':page, 'kw':kw}
# page, kw도 추가됨
return render(request, 'pybo/question_list.html', context)
```

정렬 기능

● 정렬 기능 만들기

- 최신순: 최근 등록한 질문을 먼저 보여 주는 방식
- 추천순: 추천을 많이 받은 질문을 먼저 보여 주는 방식
- 인기순: 질문에 등록된 답변이 많은 질문을 먼저 보여주는 방식

최신순

▼

찾기

번호	제목	글쓴이	작성일자
27	거리두기 4단계 1	coli	2021년 7월 23일 9:27 오전
26	코로나 1	coli	2021년 7월 23일 9:27 오전
25	코로나 1	coli	2021년 7월 23일 9:27 오전

추천순

▼

찾기

번호	제목	글쓴이	작성일자
27	휴머노이드 1	admin	2021년 7월 16일 12:18 오후
26	거리두기 4단계 1	coli	2021년 7월 23일 9:27 오전
25	코로나 1	coli	2021년 7월 23일 9:27 오전

정렬 기능

● 질문 목록 화면에 정렬 조건 추가하기

```
{% block content %}  
<div class="container my-3">  
  <div class="row justify-content-between my-3">  
    <!-- 정렬 -->  
    <div class="col-2">  
      <select class="form-control so">  
        <option value="recent" {% if so == 'recent' %} selected {% endif %}>  
          최신순  
        </option>  
        <option value="recommend" {% if so == 'recommend' %} selected {% endif %}>  
          추천순  
        </option>  
        <option value="popular" {% if so == 'popular' %} selected {% endif %}>  
          인기순  
        </option>  
      </select>  
    </div>  
    <!-- 검색 창 -->  
    <div class="col-4 input-group">
```

정렬 기능

- 질문 목록 템플릿의 searchForm 수정하기

```
<form id="searchForm" method="get" action="{% url 'pybo:board' %}">  
  <input type="hidden" name="kw" id="kw" value="{{ kw|default_if_none:'' }}">  
  <input type="hidden" name="page" id="page" value="{{ page }}">  
  <input type="hidden" name="so" id="so" value="{{ so }}">  
</form>
```

정렬 기능

- 질문 목록 템플릿의 제이쿼리 코드 추가

```
$(document).ready(function(){  
    (... 생략 ...)  
    //정렬 처리  
    $(".so").on('change', function(){  
        $("#so").val($(this).val());  
        $("#page").val(1);  
        $("#searchForm").submit();  
    });  
});
```

정렬 기능

● index 함수 수정하기

```
def board(request):  
    # 질문 목록  
    # 127.0.0.1:8000/pybo/board/?page=1  
    page = request.GET.get('page', '1') # 페이지  
    kw = request.GET.get('kw', '') # 검색어  
    so = request.GET.get('so', 'recent') # 정렬 기준  
  
    # 정렬  
    if so == 'recommend': # 추천순  
        question_list = Question.objects.annotate(  
            num_voter=Count('voter')).order_by('-num_voter', '-create_date')  
    elif so == 'popular': # 인기순  
        question_list = Question.objects.annotate(  
            num_answer=Count('answer')).order_by('-num_answer', '-create_date')  
    else: # 최신 질문  
        question_list = Question.objects.order_by('-create_date')  
  
    # 검색  
    if kw:  
        context = {'question_list': page_obj, 'page': page, 'kw': kw, 'so': so}  
        return render(request, 'pybo/question_list.html', context)
```