

Django + Crawling

Naver news Crawling Web Application

2021.01.25 ~ 2021.01.29

Contents

1. 프로젝트 내용

- 요구사항

2. 프로젝트 구성도

- 모듈 별 기능

3. 프로젝트 설명

- 코드&작동 설명

4. 개선점 및 소감

프로젝트 내용: 요구사항

경제 / IT / 스포츠뉴스 가져오기

- '최신 데이터 가져오기' 버튼으로 최신 데이터 100개 크롤링
- 검색 기능
- 각 카테고리 별로 하단에 페이지 기능



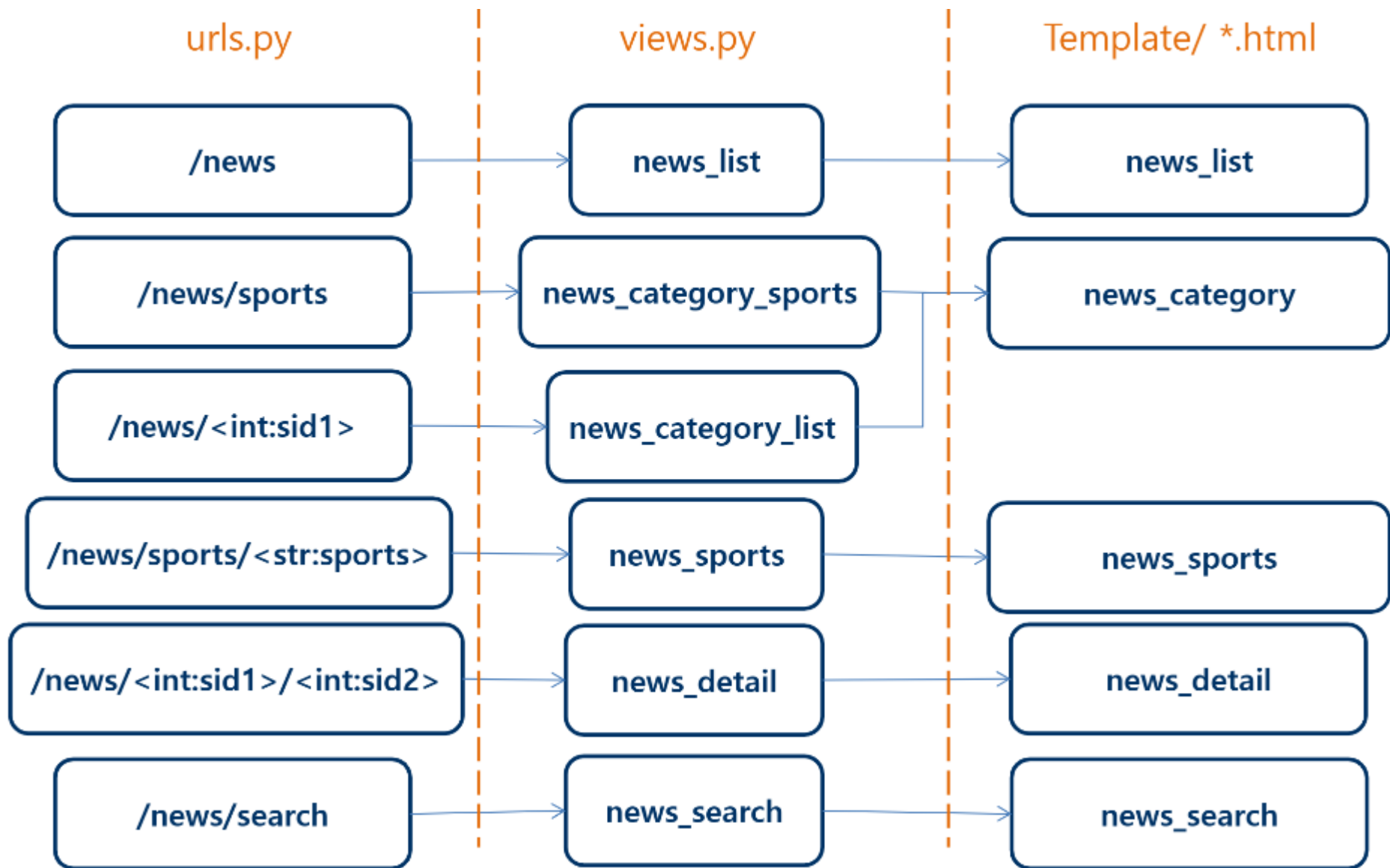
프로젝트 구성도: 모듈 별 기능

Data Base (models.py) 구성

Table 설계

필드명	타입	제약조건	설명
category	CharField(10)		뉴스 카테고리
topic	CharField(10)		뉴스 세부주제
title	CharField(100)	PK	뉴스 제목, 기본 키
letter_link	URLField		뉴스 URL
published_date	CharField(20)		뉴스 작성일자
created_date	DateTimeField	auto_now	사용자 조회일자
preview	TextField	Blank	뉴스 미리보기
writer	CharField(20)		신문사

프로젝트 구성도: 모듈 별 기능



프로젝트 구성도: 모듈 별 기능

URLConf

URL 설계		
URL 패턴 (urls.py)	View 이름 (views.py)	Template 파일명 (templates/ *.html)
/admin	Django	
/news	news_list	news_list.html
/news/sports	news_category_sports	news_cartegory.html
/news/sports/<str:sports>	news_sports	news_sports.html
/news/<int:sid1>	news_category_list	news_cartegory.html
/news/<int:sid1>/<int:sid2>	news_detail	news_detail.html
/news/search	news_search	news_search.html

프로젝트 설명

_crawling.py : Views.py에서 사용하는 크롤링 모듈

1. 경제 / IT뉴스

- BeautifulSoup4 사용 (정적 페이지)
- Method : new100_parsing(sid1, sid2), parsing(sid1, sid2)

2. 스포츠 뉴스

- Selenium 사용 (동적 페이지)
- Method : selenium_parsing_new100(sports), selenium_parsing

BeautifulSoup



Case 1: 최신 데이터 10개 ~ 20개를 가져오기

<div class= "type06_headline"> 으로 상위 10개의 뉴스를,

<div class= "type06">

스포츠 뉴스의 경우, 뉴스 리스트를 경제/IT 뉴스와 같이 상위와 하위 클래스(CSS)를 나누어 두지 않았기 때문에 전체 20개를 가져오는 selenium_parsing() 메소드, 버튼을 구현했다.

```

▶ <colgroup>...</colgroup>
▼ <tbody>
  ▼ <tr>
    ▶ <td class="snb">...</td>
    ▼ <td class="content">
      ▼ <div id="main content" class="content">
        ▶ <div class="list_header newsflash_header">...</div> == $0
        ▼ <div class="list_body newsflash_body">
          ▶ <ul class="type06_headline">...</ul>
          ▶ <ul class="type06">...</ul>

```

[illegible]

프로젝트 설명

Case 2: 최신 데이터 100개를 가져오기

뉴스 페이지는 날짜 페이지와 날짜에 종속된 페이지로 나누고 있다.

경제/IT/스포츠 동일하게 Page를 기준으로 100개를 크롤링 할 때,

1월 31일에서 2월 1일이 되고 기사가 쌓이지 않는 상황, 페이지의 CSS selector가 없거나 1페이지 밖에 없는 예외상황이 발생해서 크롤러가 멈춘다.

👉 페이지가 한 개일 때,

page 클래스로 페이지의 최대를 먼저 파악한 뒤, 데이터가 쌓일 때까지 페이지를 넘기며 크롤링한다.

👉 페이지가 maxPage까지 도달했지만 데이터가 100개가 안쌓였을 때,

day를 어제 날짜로 바꾸어서 어제 뉴스의 page 1 부터 데이터가 쌓일 때까지 페이지를 넘기며 크롤링한다.

프로젝트 설명

스포츠뉴스 크롤링 테스트 전, 필수!

```
180  def selenium_parsing(sports):  
181      URL = "https://sports.news.naver.com/"+sports+"/news/index.nhn?isphoto=N"  
182  
183      chrome_options=webdriver.ChromeOptions()  
184      chrome_options.add_argument('headless')  
185      chrome_options.add_argument('--disable-gpu')  
186      chrome_options.add_argument('lang=ko_KR')  
187      path = "C:\\cloud\\selenium\\webdriver\\chromedriver.exe"  
188      driver = webdriver.Chrome(path, chrome_options= chrome_options)  
189
```

Selenium의 경우, Web driver가 필요하다.

이미지의 183~186 Line은 Chrome이 활성화되면서 창이 열리기 때문에, 크롤링 할 때 Chrome 브라우저를 백그라운드로 실행시킨다.

프로젝트 설명

news_list.html : 127.0.0.1:8000/

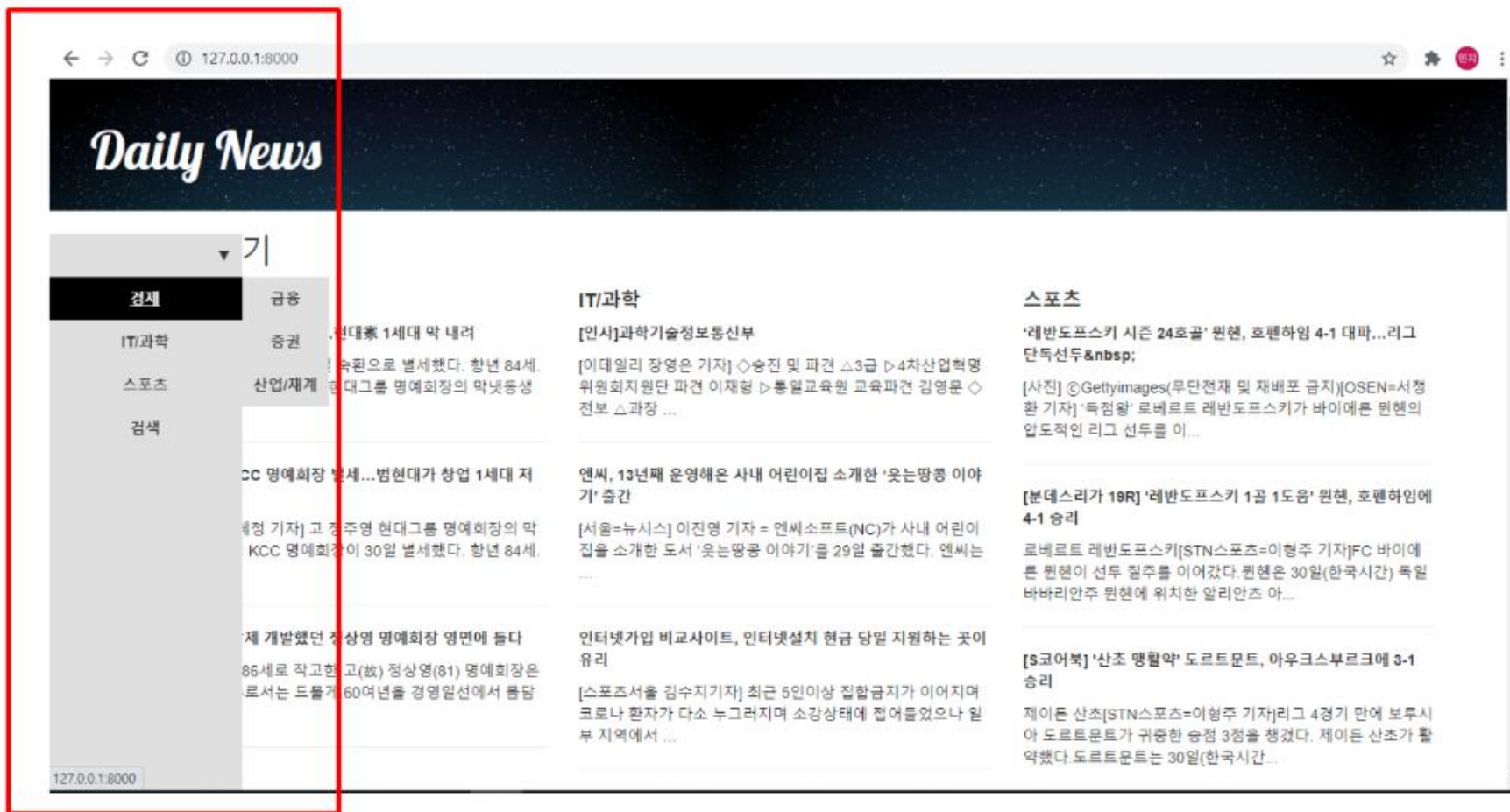
- 각 카테고리 별로 가장 최신 데이터 3개를 미리 보여준다.



프로젝트 설명

news_list.html

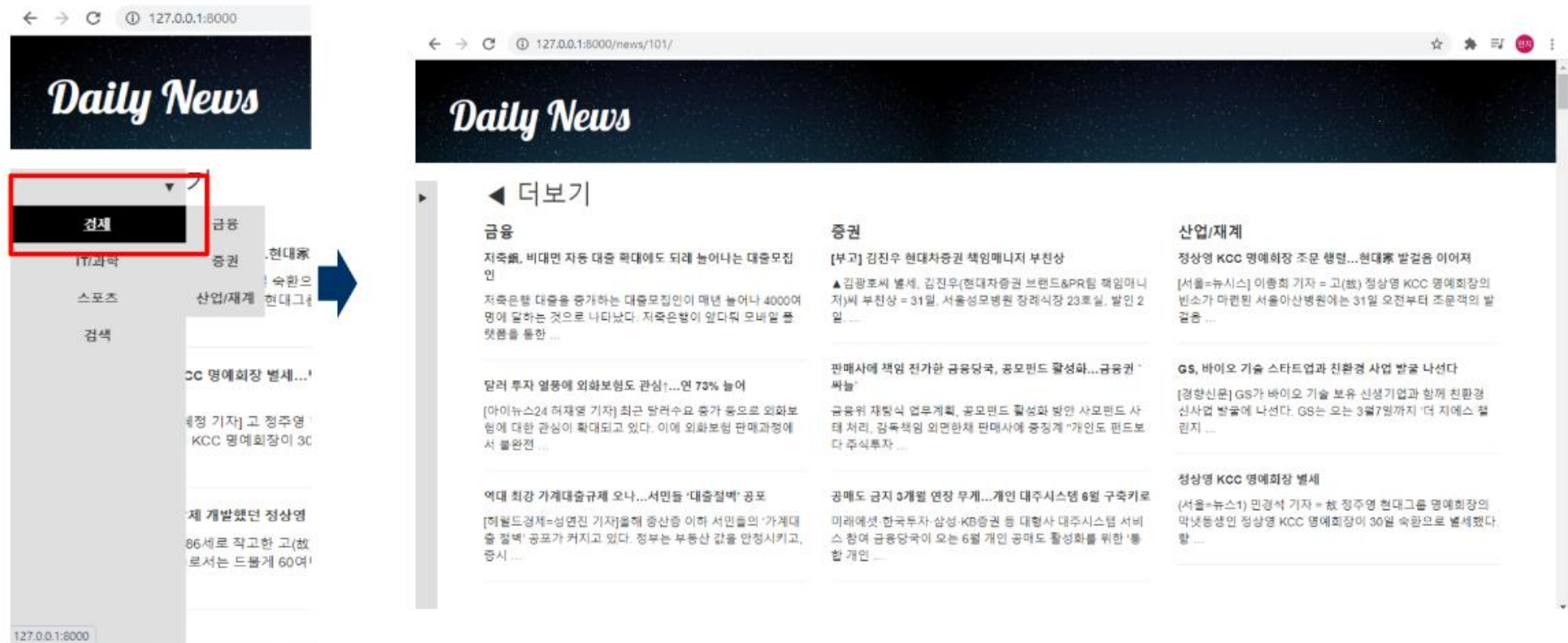
- Sidebar(nav)로 카테고리 하위 페이지를 표시



프로젝트 설명

news_category_list.html

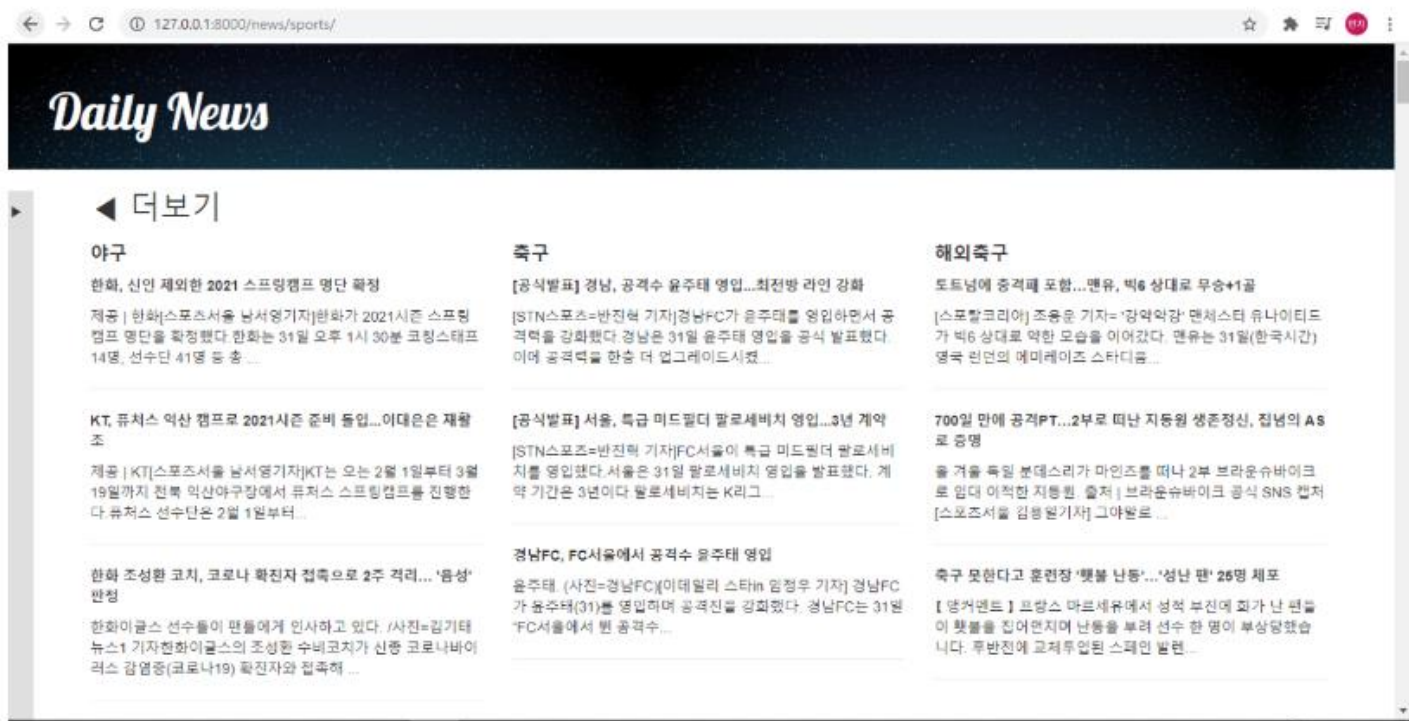
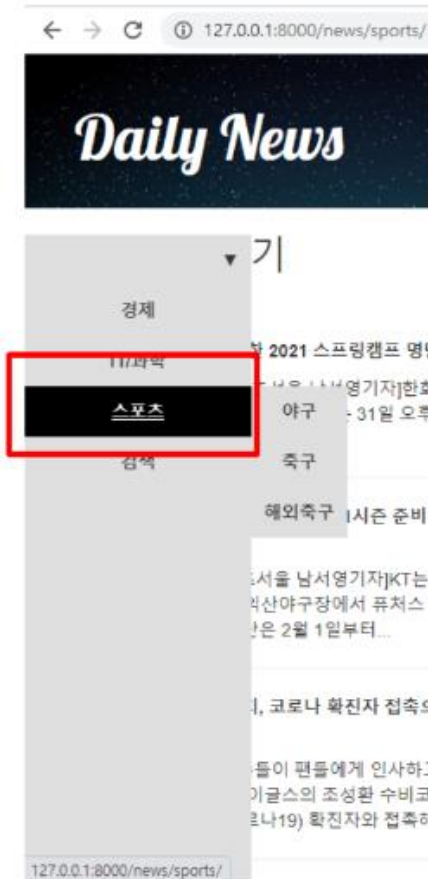
- /news/<ini: sid1>
- /news/sports
- Sidebar에서 Category 클릭시, 3개의 뉴스 토픽의 가장 최신 뉴스 3개를 미리 보여준다.



프로젝트 설명

news_category_list.html

- 경제/IT, 스포츠 크롤링은 views.py 메소드는 다르지만(URL, 크롤링) 같은 news_category.html로 렌더링 된다.
- 그러므로 news_category_sports()와 news_category_list() 사용하는 Queryset, 데이터 구조 또한 같다.



프로젝트 설명

views.py : news_category_list()

```
def news_category_list(request, sid1):
    key = [str(sid1),]
    economy = ['259', '258', '261']
    it_sc=['731', '226', '227']
    economys=[]
    it_scs=[]
    contents={}
    func = lambda __sid1__, __sid2__: Letter.objects.filter(category__icontains=__sid1__,
    topic__icontains=__sid2__).order_by('-created_date')
    if key[0] == '101':
        print('101')
        for sid2 in economy:
            economy_topic= func(sid1,sid2)
            economys.append(economy_topic)
            contents={'m': economys}
            key.append(economy)

    elif key[0] == '105':
        print('105')
        for sid2 in it_sc:
            it_topic = func(sid1,sid2)
            it_scs.append(it_topic)
            contents={'m': it_scs}
            key.append(it_sc)

    return render(request, 'news/news_category.html', {'message': contents , 'key': key})
```

프로젝트 설명

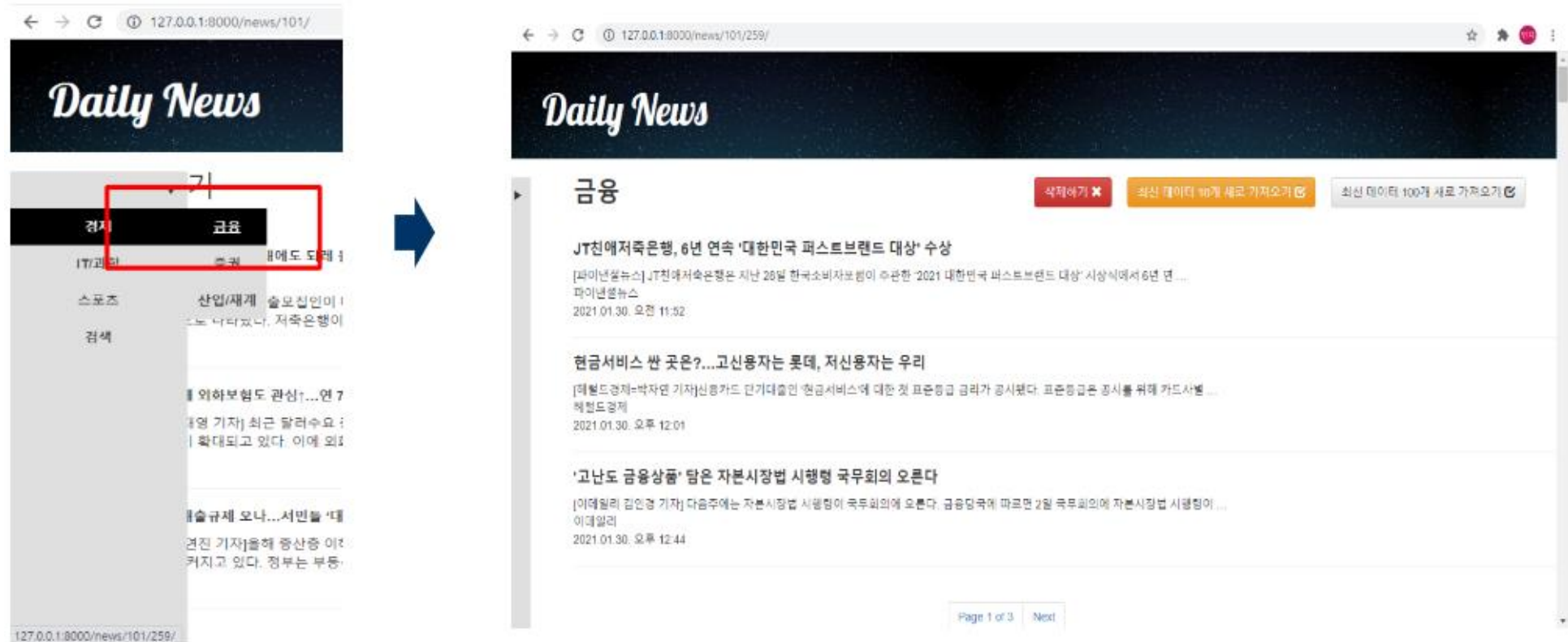
views.py : news_category_list() 데이터 구조

```
contents={
    economys[
        economy_topic[
            (금융)Query Resultset()..
            (증권)Query Resultset()..
            (산업/재계)Query Resultset()..
        ]
    ]
}
```


프로젝트 설명





news_detail.html

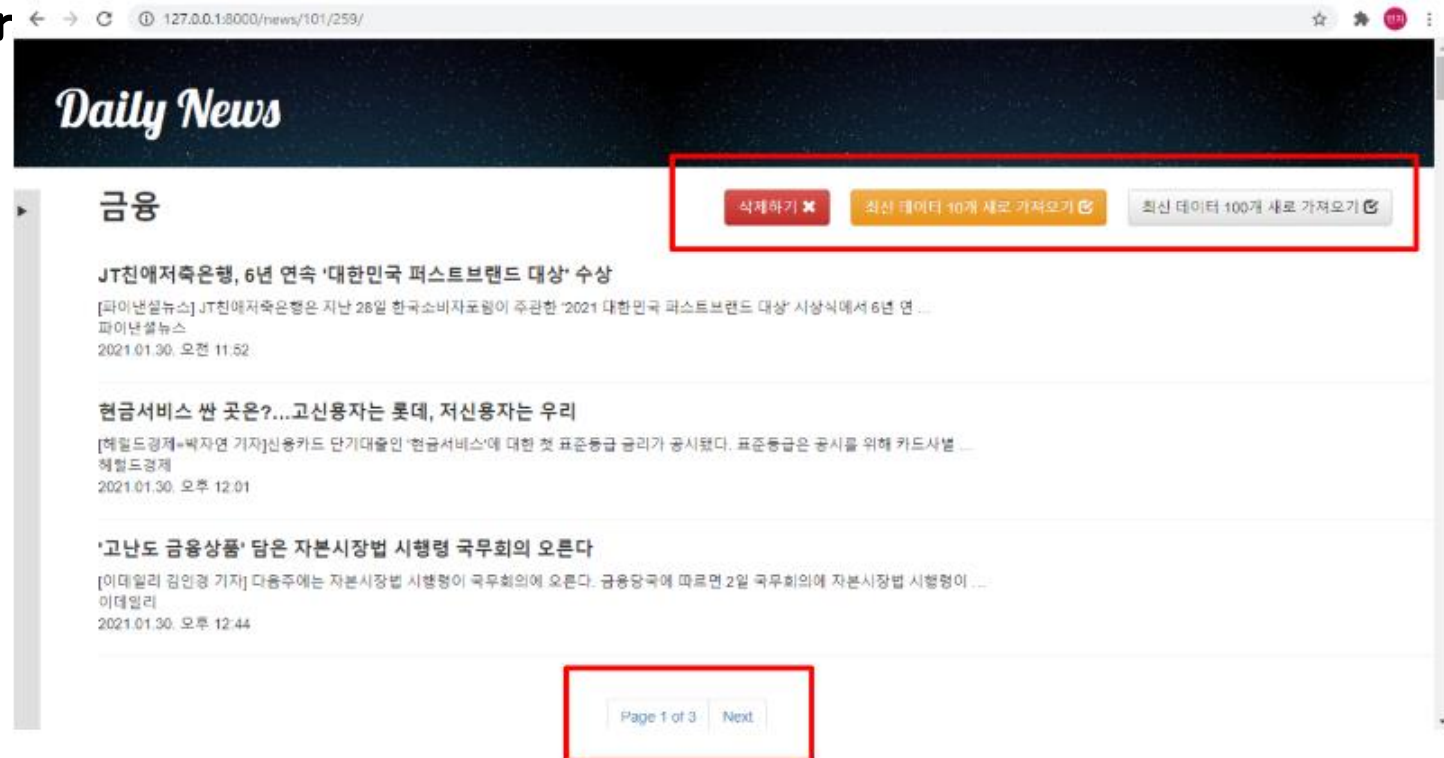
- /news/⟨int: sid1⟩/⟨int: sid2⟩
- Sidebar에서 카테고리 하위 토픽을 클릭하면 해당 토픽 페이지로 이동한다.
- 경제, IT/과학 카테고리의 경우 news_detail.html을 사용



프로젝트 설명

news_detail.html

-  : 최신 데이터를 페이지 기준에서 => 날짜 기준으로 100개 가져와서 DB에 저장
-  : 최신 데이터를 10개만 가져와서 DB에 저장한다.
-  : 현재 카테고리의 토픽(아래 이미지에서는 금융) 뉴스만 DB에서 삭제한다.
-  : Django > Paginator



프로젝트 설명

views.py: news_detail()

- DB update와 delete를 위한 UDForm으로 parsing 메소드 (BeautifulSoup)를 사용해서 뉴스를 크롤링한다.

```
def news_detail(request, sid1, sid2):
    mylist = [str(sid1), str(sid2)]
    if request.method == 'POST':
        UDform = UDForm(request.POST)
        if UDform.is_valid():
            # 최신 데이터 10개 가져오기
            if UDform.cleaned_data['UD'] == 'update':
                contents = parsing(sid1, sid2)
                for data in contents:
                    form = Letter(
                        category=str(sid1),
                        topic=str(sid2),
                        title= data[0],
                        letter_link=data[1],
                        published_date=data[2],
                        preview= data[3],
                        writer=data[4]
                    )
                    form.save()
```

최신 데이터 10개 새로 가져오기

```
news > forms.py > UDForm
1 from django import forms
2
3 class SearchForm(forms.Form):
4     news_topic = forms.CharField(label='Topic keyword',max_length=10)
5     news_title = forms.CharField(label='Search keyword', max_length=20)
6
7 #update, delete form
8 class UDForm(forms.Form):
9     UD = forms.CharField(label='UD',max_length=10)
```

프로젝트 설명

views.py: news_detail()

- 최신 100개의 경우, 10개에서 사용하는 메소드와 달리 new100_parsing() 메소드를 사용한다.
- DB delete는 페이지에 해당하는 topic을 찾아서 삭제한다.

```
92 # 최신 데이터 100개 가져오기
93 elif UDform.cleaned_data['UD'] == 'new100':
94     contents = new100_parsing(sid1,sid2)
95     for data in contents:
96         form = Letter(
97             category=str(sid1),
98             topic=str(sid2),
99             title= data[0],
100             letter_link=data[1],
101             published_date=data[2],
102             preview= data[3],
103             writer=data[4]
104         )
105         form.save()
106 # 데이터 전부 삭제
107 elif UDform.cleaned_data['UD'] == 'delete':
108     Letter.objects.filter(topic=sid2).delete()
109
110 else:
111     print('GET')
112 News = Letter.objects.filter(topic= str(sid2),created_date__lte = timezone.now()).order_by
113     ('-created_date')
```

최신 데이터 100개 새로 가져오기

삭제하기

프로젝트 설명

views.py: news_detail()

- Django에서 제공하는 Paginator를 사용해서 DB에서 조회한 데이터(News)를 페이지 당 3개의 데이터로 나누었다.
- Previous, Next 페이지가 있을 때에만 활성화해준다.

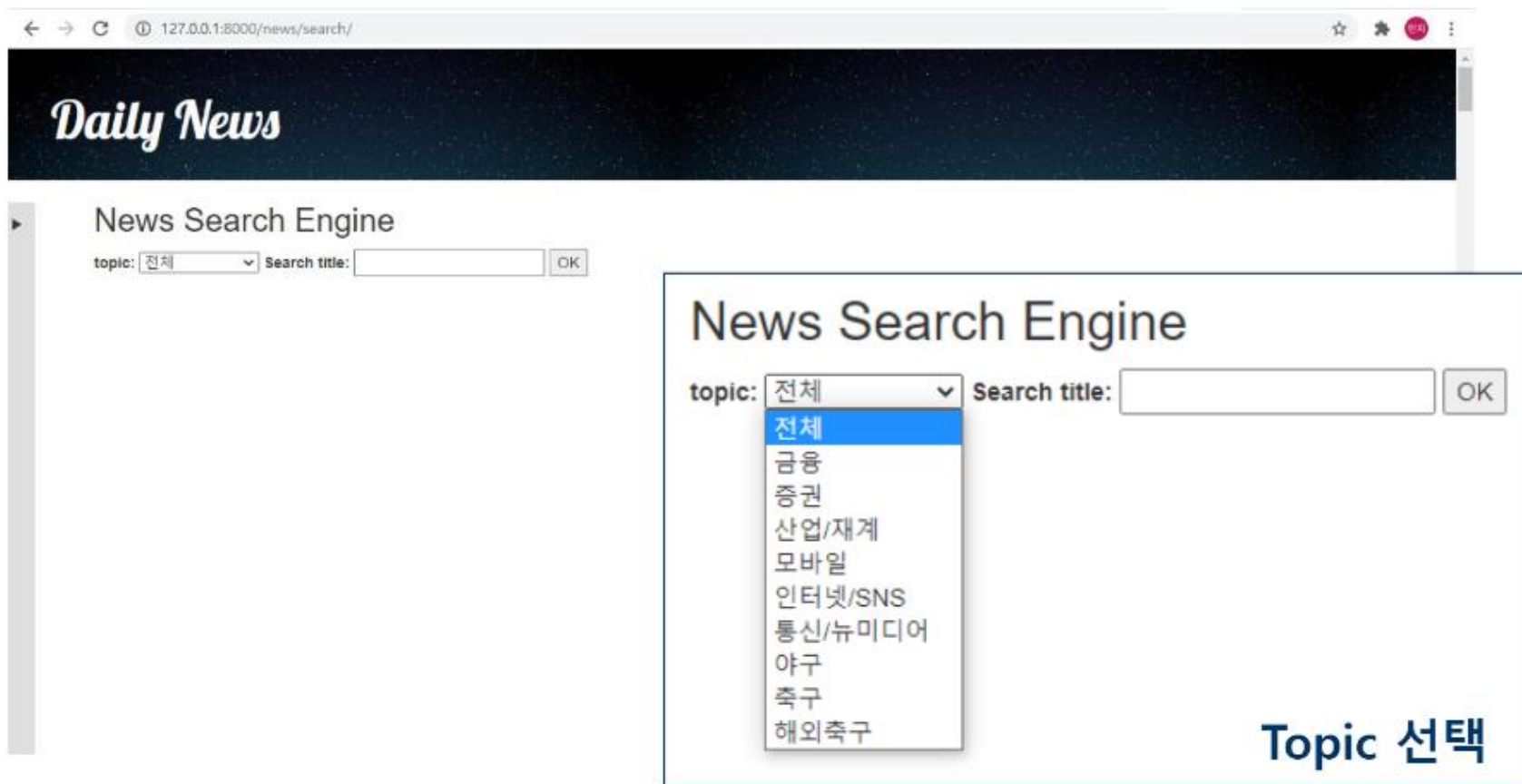
```
114     # pagination
115     paginator = Paginator(News,3)
116     page_no = request.GET.get('page')
117     try:
118         News_p = paginator.page(page_no)
119     except PageNotAnInteger:
120         News_p = paginator.page(1)
121     except EmptyPage:
122         News_p = paginator.page(paginator.num_pages)
123
124     return render(request, 'news/news_detail.html' ,{'message': News_p, 'key':mylist})
125
```

[Previous](#)[Page 2 of 3](#)[Next](#)

프로젝트 설명

news_search.html

- /news/search
- 뉴스 topic을 선택해서 제목의 키워드로 검색을 제공해준다.



프로젝트 설명

news_search.html

- 전체 topic에서 '현금' 검색

News Search Engine

topic: 전체 ▼ Search title: 현금 OK



News Search Engine

topic: 전체 ▼ Search title: OK

금융

현금서비스 싼 곳은?...고신용자는 롯데, 저신용자는 우리

[헤럴드경제=박자연 기자]신용카드 단기대출인 '현금서비스'에 대한 첫 표준등급 금리가 공시됐다. 표준등급은 공시를 위해 카드사별 ...
헤럴드경제

2021.01.30. 오후 12:01

인터넷/SNS

인터넷가입 비교사이트, 인터넷설치 현금 당일 지원하는 곳이 유리

[스포츠서울 김수지기자] 최근 5인 이상 집합금지가 이어지며 코로나 환자가 다소 누그러지며 소강상태에 접어들었으나 일부 지역에서 ...
스포츠서울

2021.01.30. 오전 12:00

프로젝트 설명

news_search.html

- 금융 topic에서 '현금' 검색

The screenshot displays a web application titled "News Search Engine". It features a search interface with a "topic:" dropdown menu set to "금융" (Finance) and a "Search title:" input field containing the text "현금" (Cash). An "OK" button is visible next to the search field. Below the search bar, a list of search results is shown, including a headline "현금서비스 싼 곳은?...고신용자는 롯데, 저신용자는 우리" (Cash service is cheap where?...high credit users are Lotte, low credit users are Woori) and a sub-headline "[헤럴드경제=박자연 기자]신용카드 단기대출인 '현금서비스'에 대한 첫 표준등급 금리가 공시됐다. 표준등급은 공시를 위해 카드사별 ...". The date and time of the search are "2021.01.30. 오후 12:01".

A blue arrow points from the search results to a second, identical screenshot of the "News Search Engine" interface. This second screenshot shows the same search results, but the "topic:" dropdown menu is now set to "전체" (All) instead of "금융".

프로젝트 설명

news_search.html

```
news > templates > news > news_search.html
16
17 <label for="news_topic">topic:
18 </label>
19 <select name="news_topic">
20 <option value="default" selected="selected">전체</option>
21 <option value="259"> 금융 </option>
22 <option value="258"> 증권 </option>
23 <option value="261"> 산업/제계 </option>
24 <option value="731"> 모바일 </option>
25 <option value="226"> 인터넷/SNS </option>
26 <option value="227"> 통신/뉴미디어 </option>
27 <option value="kbaseball"> 야구 </option>
28 <option value="kfootball"> 축구 </option>
29 <option value="wfootball"> 해외축구 </option>
30 </select>
31 <label for="news_title">Search title: </label>
32 <input
33 id="news_title"
34 type="text"
35 name="news_title"
36 value="{ search_keywo
37 <input type="submit" v
38 <br/><br/>
```

- Select box로 topic을 선택
- Text input에서 검색 키워드를 입력
- news_search(request)로 POST submit
- news_topic, news_title로 Form을 만들고 Queryset으로 검색

```
news > forms.py > UDForm
```

```
1 from django import forms
2
3 class SearchForm(forms.Form):
4     news_topic = forms.CharField(label='Topic keyword',max_length=10)
5     news_title = forms.CharField(label='Search keyword', max_length=20)
6
7 #update, delete form
8 class UDForm(forms.Form):
9     UD = forms.CharField(label='UD',max_length=10)
```

프로젝트 설명

views.py: news_search()

```
175 def news_search(request):
176     context = {}
177     if request.method == 'POST':
178         form = SearchForm(request.POST)
179         if form.is_valid():
180             topic = form.cleaned_data['news_topic']
181             _title = form.cleaned_data['news_title']
182             news_list = []
183             if topic == 'default':
184                 news_list = Letter.objects.filter(title__icontains=_title)
185             else:
186                 news_func = lambda __x__: Letter.objects.filter(Q(topic__icontains=__x__) & Q
187                     (title__icontains=_title)).distinct()
188                 news_list = news_func(topic)
189             context['form'] = form
190             context['search_keyword'] = _title
191             context['result_list'] = news_list
192             return render(request, 'news/news_search.html', {'search_list': context})
193
194     else:
195         form = SearchForm()
196
197     return render(request, 'news/news_search.html', {'search_list': context})
```

개선점 및 소감

1. 개선점

- Front가 정리되지 못했다. 추후 Front Framework를 공부필요!
- 최신 데이터 100개의 경우, Template에 가져오기까지 시간이 오래걸린다. 크롤링 방법과 코드 최적화에 개선이 필요하다.

2. 소감

- 처음 구현해보는 웹 MVC (Django의 경우 MTV)
- HTML&CSS를 크롤링하면서 자연스럽게 공부할 수 있었다.
- 정적 웹페이지와 동적 웹페이지에 대해 알 수 있었고 이에 따라 크롤링 방법 또한 다르게 해서 경험해볼 수 있었다.