추가학습 (#hashtag 구현하기)

자료	DB
구분	DB
과목	

Hashtag 구현을 연습해보자.

모델 관계 설정 (M:N)

- BD에 저장할 Hashtag 모델을 정의한 후에
- 기존 Movie 테이블에 ManyToManyField 를 사용해 Hashtag와의 중개테이블을 추가 해 본다.

```
# movies/models.py

class Hashtag(models.Model):
    content = models.TextField(unique=True)

    def __str__(self):
        return self.content



class Movie(models.Model):
    hashtags = models.ManyToManyField(Hashtag, blank=True)
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
                             on_delete=models.CASCADE)

    like_users = models.ManyToManyField(settings.AUTH_USER_MODEL,
                                        related_name='like_movies')
    title = models.CharField(max_length=20)
    description = models.TextField()
```

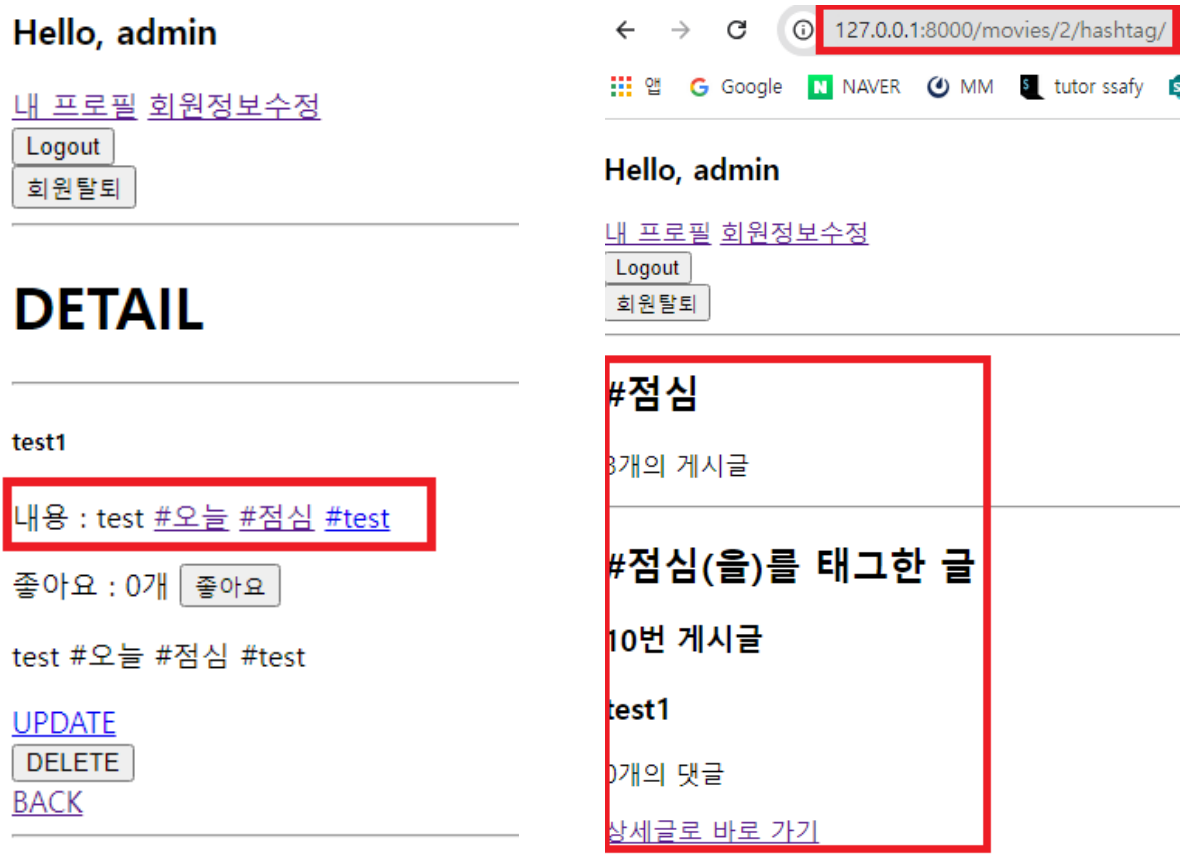
- Hashtag 모델을 살펴보면 `unique=True` 옵션을 넣어 주었다.
    - unique arguments가 `True` 인 경우 이 필드는 테이블 전체에서 고유한 값임을 의미한다.
- def __str__(self):
  매직매소드를 사용해서
  `Hashtag` 객체를 문자열로 표현하겠다는 의미 이다. 예를 들어, `Hashtag` 를 조회하면 이 객체의 내

용은 문자열 형태로 출력이 될 것이다. 그렇게 되면 DB에서 해시태그를 확인할 때 객체의 형태가 아닌 문자열의 형태로 쉽게 확인이 가능 하다.

URL 및 view 작성

우리는 디테일 페이지에 있는 Hash tag를 클릭하면

해당 tag와 관련된 게시물이 나오는 페이지를 제작 하려고 한다.




```
# movies/urls.py

from django.urls import path
from . import views

app_name = 'movies'

urlpatterns = [
    ...,
```

```
    path('<int:hash_pk>/hashtag/', views.hashtag, name='hashtag'),
]
```

```
# movies/views.py

from .models import Hashtag
from django.shortcuts import get_object_or_404

@login_required
def hashtag(request, hash_pk):
    hashtag = get_object_or_404(Hashtag, pk=hash_pk)
    movies = hashtag.movie_set.order_by('-pk')
    context = {
        'hashtag': hashtag,
        'movies': movies,
    }
    return render(request, 'movies/hashtag.html', context)
```

- Hashtag 객체를 DB에서 가져오는데 만약에 해당 pk에 해당하는 해시태그가 없다면 404 오류 페이지를 반환하도록 코드를 작성했다. 그리고
- 조회된 해시태그에 연결된 영화게시글을 `movie_set`을 통해 가져온다. 나는 -pk로 정렬해서 가장 최근에 생성된 영화가 먼저 나오게 했다.

자 이제 새 영화관련 게시글을 생성하면 Hashtag도 넣을 수 있도록

Create 함수 그리고 Update 함수를 수정해 보자.

CREATE 함수 수정

```
# movies/views.py

@login_required
@require_http_methods(['GET', 'POST'])

def create(request):
    if request.method == 'POST':
        form = MovieForm(request.POST)
        if form.is_valid():
            movie = form.save(commit=False)
            movie.user = request.user
            movie.save()
```

```
            for word in movie.description.split():
                if word.startswith('#'):
                    hashtag, created = Hashtag.objects.get_or_create(content=word)
                    movie.hashtags.add(hashtag)

            return redirect('movies:detail', movie.pk)
        else:
            form = MovieForm()
        context = {
            'form': form,
        }
        return render(request, 'movies/create.html', context)
```

붉은 글씨의 for구문을 살펴보자.

- movie.description 은 movie모델의 필드로, 영화의 내용을 담고 있는 문자열이다.

  여기서 split()을 이용해서 공백을 기준으로 나누어 리스트로 만들었다.

- `#`으로 시작하는 단어를 찾아서 우리가 DB에 만든 Hashtag 모델 그리고 게시글에 데이터 추가하는데
  - `get_or_create()` 메서드 활용했다. 이는 모델 객체를 생성할 때 이미 있는 객체라면 가져오고(검색) 없으면 새로 생성하는 메서드 이다.
  - (hashtag, created) 튜플의 형태로 저장하는데
    - hashtag 부분에는 검색 또는 새로 생성된 객체가 저장되며
    - created 에는 새 객체의 생성 여부를 지정하는 boolean 값이 저장이 된다.

  `get_or_create` 메서드를 간단히 정리하면 아래의 두 가지 경우로 설명이 가능하다.

  1. (content=word)에 들어간 조건에 해당하는 인스턴스가 존재 한다면 데이터베이스에서 해당 인스턴스를 꺼내와 object에 대입하고, flag는 False가 된다.
  2. 조건에 해당하는 인스턴스가 존재하지 않는다면, 해당 조건을 만족하는 인스턴스를 생성하여 Objesct에 대입하고, flag는 True가 된다.

     즉, created(boolean flag)는 `get_or_create` 메서드에 의해 생성이 되면 True, 생성되지 않고 데이터베이스에서 꺼내오면 False가 저장 된다.

     사실 created를 사용하여 해시태그 생성 여부를 확인하고 어떤 추가 기능을 넣을 필요가 없다면, created를 생략해도 상관없다. created는 당장은 필요 없는 부분 이지만, `get_or_create` 메서드를 설명하는데 참고로 알고 있었으면 좋겠다는 생각에 작성해 보았다.

- add()를 통해서 해시태그 내용을 중개테이블에 Data를 추가 해 주었다.

Update 함수 수정

- 기존에 있던 hashtag 삭제 후 create와 동일한 작업 수행

```
# movies/views.py

@login_required
@require_http_methods(['GET', 'POST'])
def update(request, pk):
    movie = get_object_or_404(Movie, pk=pk)
    if request.user == movie.user:
        if request.method == 'POST':
            form = MovieForm(request.POST, instance=movie)
            if form.is_valid():
                form.save()

                movie.hashtags.clear()

                for word in movie.description.split():
                    if word.startswith('#'):
                        hashtag, created = Hashtag.objects.get_or_create(content=word)
                        movie.hashtags.add(hashtag)

                return redirect('movies:detail', movie.pk)
        else:
            form = MovieForm(instance=movie)
    else:
        return redirect('movies:index')
    context = {
        'movie': movie,
        'form': form,
    }
    return render(request, 'movies/update.html', context)
```

Update 함수에서는 Create함수와 다른점은 movie.hashtags.clear() 를 추가 한 점이다.

Clear()를 통해서 중개테이블에서 영화게시글과 해시태그와의 관계만 끊는다. 즉, 영화게시글과 해시테그 사이의 연결 정보만 중개 테이블에서 삭제가 된다.

하지만, 해시태그 자체는 데이터베이스에 그대로 남아 있다.

예를들어 보자.

위에서 작성했었던 models.py 안에 클래스를 보면서 아래 설명을 보자.

```python
# movies/models.py

class Hashtag(models.Model):
    content = models.TextField(unique=True)

    def __str__(self):
        return self.content


class Movie(models.Model):
    hashtags = models.ManyToManyField(Hashtag, blank=True)
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
                             on_delete=models.CASCADE)

    like_users = models.ManyToManyField(settings.AUTH_USER_MODEL,
                                        related_name='like_movies')
    title = models.CharField(max_length=20)
    description = models.TextField()
```

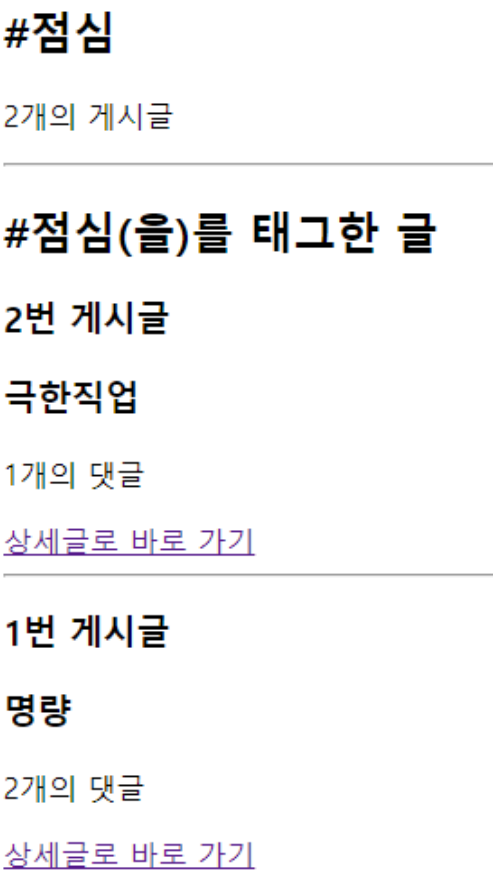
- 예를 들어 영화 게시글에 #액션, #스릴러라는 해시태그가 붙어 있었다고 해보자.
- `movie.hashtags.clear()`를 하면, 이 영화 게시글에서 #액션, #스릴러 연결이 끊기지만.
  - (Movie 모델의 hashtags 필드의 내용은 모두 삭제되지만)
- #액션, #스릴러 해시태그 자체는 삭제되지 않고 DB에 그대로 남겨 둔다는 뜻이다.
  - (Hashtag 모델의 DB에는 정보를 남겨둔다)
- 그래서 나중에 이 해시태그들을 다시 연결하거나, 다른 영화 게시글에 쓸 수도 있다.

자 이제 해시태그를 클릭했을 때

해당 해시태그를 사용한 게시물들을 모아서 보여주는 페이지 작성해 보자.

Template 작성

- movies/templates/movies 폴더에 hashtag.html 파일을 새로 생성하자.

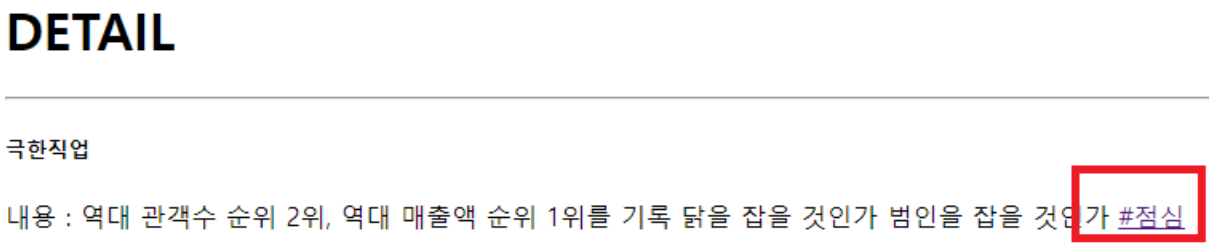



```
# movies/hashtag.html

{% extends 'base.html' %}

{% block content %}
<div>
  <h2>{{ hashtag.content }}</h2>
  <p>{{ movies|length }}개의 게시글</p>
</div>

<hr />

<div>
  <h2>{{ hashtag.content }}(을)를 태그한 글</h2>
  {% for movie in movies %}
    <h3>{{ movie.pk }}번 게시글</h3>
    <h3>{{ movie.title }}</h3>
    <p>{{ movie.comment_set.all|length }}개의 댓글</p>
    <a href="{% url 'movies:detail' movie.pk %}">상세글로 바로 가기</a>
    <hr />
```

```
 {% endfor %}
</div>
{% endblock %}
```

자 그다음에는 영화관련 게시판의 게시글 중에서

`해시태그` 에 해당되는 부분만 따로 링크로 연결해 보자.



사용자 정의 템플릿 태그

해시태그에 해당되는 부분만 따로 링크로 연결하기 위해서 Custom template filter 가 필요하다.

커스텀 템플릿 필터란? 주어진 데이터를 사용자의 입맛에 맞도록 변형해서 사용할 수 있도록

DTL에서 필터를 새로 만드는 것을 말한다. (고급스킬 ㅋㅋ)

사용방법은

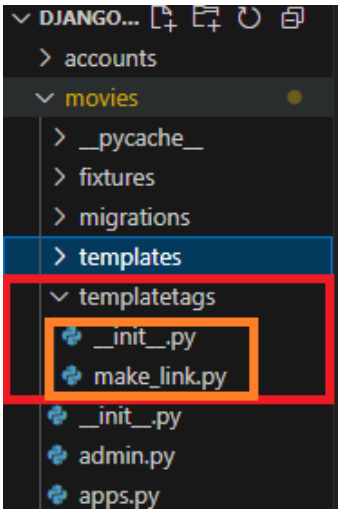
1. 커스텀 필터를 정의 해 줄 것이다.
2. register.filter 데코를 이용해서 Django 템플릿에 해당 필터를 등록을 해준다.
3. 사용하고자 하는 템플릿에서 사용하면 된다.

일단은 아래 실습을 모두 따라한 다음 아래 공식문서도 확인해 보도록 하자.

Django
The web framework for perfectionists with deadlines.
https://docs.djangoproject.com/ko/4.2/howto/custom-template-tags/

1. Templatetags 패키지 생성하기

- movies 앱 폴더 안에 templatetags 이름으로 폴더를 새로 만들고 `__init__.py`, 그리고 `make_link.py` 이름으로 파일을 새로 만든다.



- `__init__.py`
  - 해당 폴더가 파이썬 패키지라는 것을 명시해주는 것
  - 안에 내용이 없어도 상관 없음
- `make_link.py`
  - Custom template filter 를 정의할 것이다.
  - Django 템플릿에서 사용할 커스텀 필터를 이 파일에 정의할 것이다.
  - 이후 이 필터를 템플릿에서 `{% load make_link %}`로 불러와 사용하게 된다.

pycache 는 나중에 서버를 켜면 자동으로 생성되므로 지금 신경쓰지 않도록 하자.

2. 커스텀 템플릿 필터 작성하기

- 커스텀 필터의 인자로 들어오는 데이터에서 '#' 문자와 그 뒤에 오는 문자를 구분하여 반환하는 내용이 담긴 함수를 작성 할 것이다.
- 먼저 Django의 template 모듈을 import 하자.

```
# movies/templatetags/make_link.py
```

```
from django import template

register = template.Library()
```

- register = template.Library()를 통해서 Django 템플릿에 커스텀 필터를 등록할 수 있는 객체를 생성한다.

  template.Library() 메서드는 우리가 커스텀 할 필터를 Django 템플릿에 추가해 줌으로써 Django가 해당 태그와 필터를 템플릿에서 인식할 수 있도록 하는 것이다.

3. 필터를 통해서 적용 할 함수를 정의하기

```
# movies/templatetags/make_link.py

from django import template

register = template.Library()


@register.filter
def hashtag_link(word):
    content = word.description + ' '
    hashtags = word.hashtags.all()
    for hashtag in hashtags:
        content = content.replace(hashtag.content + ' ',
            f'<a href="/movies/{hashtag.pk}/hashtag/">{hashtag.content}</a> ')
    return content
```

hashtag_link 함수를 정의 했다.

이 함수는 영화 게시글에 포함된 해시태그에 링크를 다는 역할을 할 것이다.

위의 함수가 하는 일은 다음과 같다.

1. `word.description`에 있는 텍스트를 가져온다. (예: `"이 영화 너무 재밌다 #액션 #스릴러"`).
2. 해당 객체에 연결된 모든 `hashtags`를 가져온다.
3. 그 해시태그들 각각에 대해:
   - `description` 안에 있는 `#해시태그`를
   - `<a href="...">해시태그</a>` 형식의 링크로 변환할 것이다.

자 이제 코드 한줄한줄씩 살펴보자.

- def hashtag_link(word)
  - word 매개변수를 통해서 해시태그와 관련된 영화의 정보를 담은 객체를 저장한다.
  - 매개변수에 들어갈 인자값은 나중에 detail 템플릿에서 커스텀필터를 통해 전달 될 것이다.
- content = word.description + ' '
  - 이 부분은 영화의 게시글 끝에 ' ' 공백을 추가해서 해시태그의 구분을 돕도록 할 것이다.
- hashtags = word.hashtags.all()
  - 중개테이블에서 해당 영화 게시글에서 사용된 모든 해시태그를 가져와서
- f'{hashtag.content}
  - a 태그를 이용해서 해시태그를 클릭했을 때 이동 할 링크를 생성한다. 그리고
- content = content.replace(hashtag.content + ' ',
  - 를 통해서 공백을 구분으로 찾은 해시태그를 대체한다.
- 마지막으로 태그를 단 content객체를 반환한다.

Templatetags 폴더를 추가하고 난 후에는 서버를 재시작 해야 정상적으로 적용된다.

서버를 끄고 migration migrate를 하고 서버를 다시 켜자.

4. 작성한 커스텀 템플릿 태그 적용하기

- `detail.html`에 load 태그를 통해 우리가 직접 제작한 템플릿 태그 불러 올 것이다.

```
# movies/templates/movies/detail.html


{% extends 'base.html' %}
{% load make_link %}

{% block content %}
  <h1>DETAIL</h1>
  <hr />
  <div>
    <h5>{{ movie.title }}</h5>
    <p>내용 : {{ movie|hashtag_link|safe }}</p>

    <form action="{% url 'movies:likes' movie.pk %}" method="POST">
    <span>좋아요 : {{ movie.like_users.count }}개</span>
```

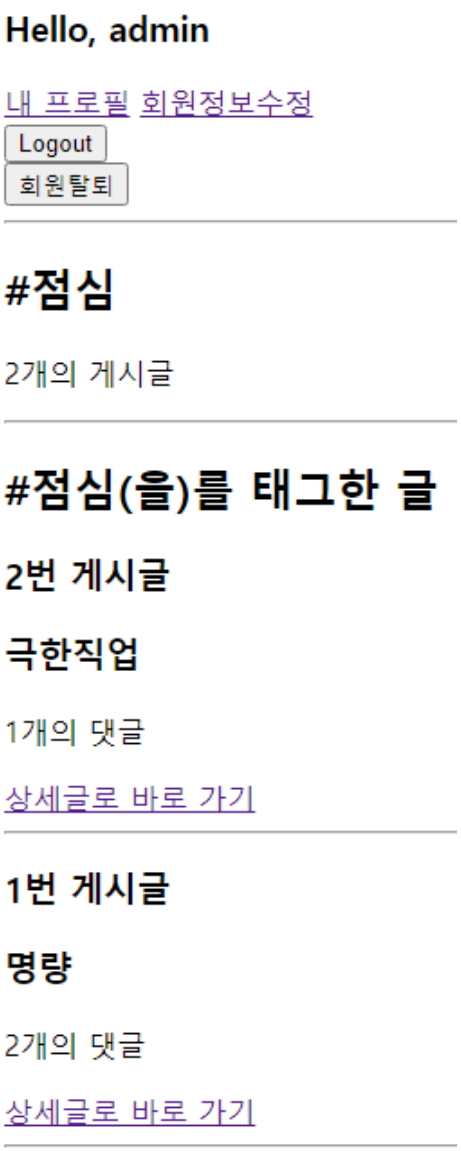
```
...
```

- `{% load make_link %}` 를 통해서 우리가 커스텀해서 만든 탬플릿필터를 불러온다.
- `{{ movie|hashing_link|safe }}` 를 살펴보자.
  - movie 객체를 해당 커스컴 필터의 인자값으로 넘겨준다. (hashtag_link 함수로 전달)
  - `hashtag_link` 필터를 통해 HTML `<a>` 태그가 삽입된 문자열이 반환되고,
  - `safe` 필터를 이용해서 링크의 형태로 렌더링이 된다.
    - safe 필터는 Django 템플릿에서 HTML 내용을 안전하게 렌더링하기 위해 사용되는데 만약에 safe 필터를 사용하지 않으면 Django는 보안상의 이유로 텍스트를 자동으로 이스케이프 처리해서, HTML 태그가 그냥 텍스트로 변환이된다.
    - 그러나 `safe` 필터를 붙이면, Django가 해당 문자열을 **신뢰된 HTML**로 간주해서 그대로 a태그의 형태로 출력이 된다. 따라서 링크기능을 넣으려면 safe 필터를 반드시 넣어줘야 제대로 기능이 작동이 된다.

DETAIL

극한직업

내용 : 역대 관객수 순위 2위, 역대 매출액 순위 1위를 기록 닭을 잡을 것인가 범인을 잡을 것인가 #점심



서버 켜서 확인해 보자.

이상으로 해시태그 기능을 구현해 보았다. 화이팅 !!

(completed_hashtag)_django_test.zip