

PART 01

사전 준비 작업

레포를 clone 받은 후 가상환경 activate하고 (pipenv shell)

- pip install django

- pip install django-allauth

- pip install pillow

그 이후

manage.py가 있는 폴더로 이동한 후

- python manage.py makemigrations

- python manage.py migrate

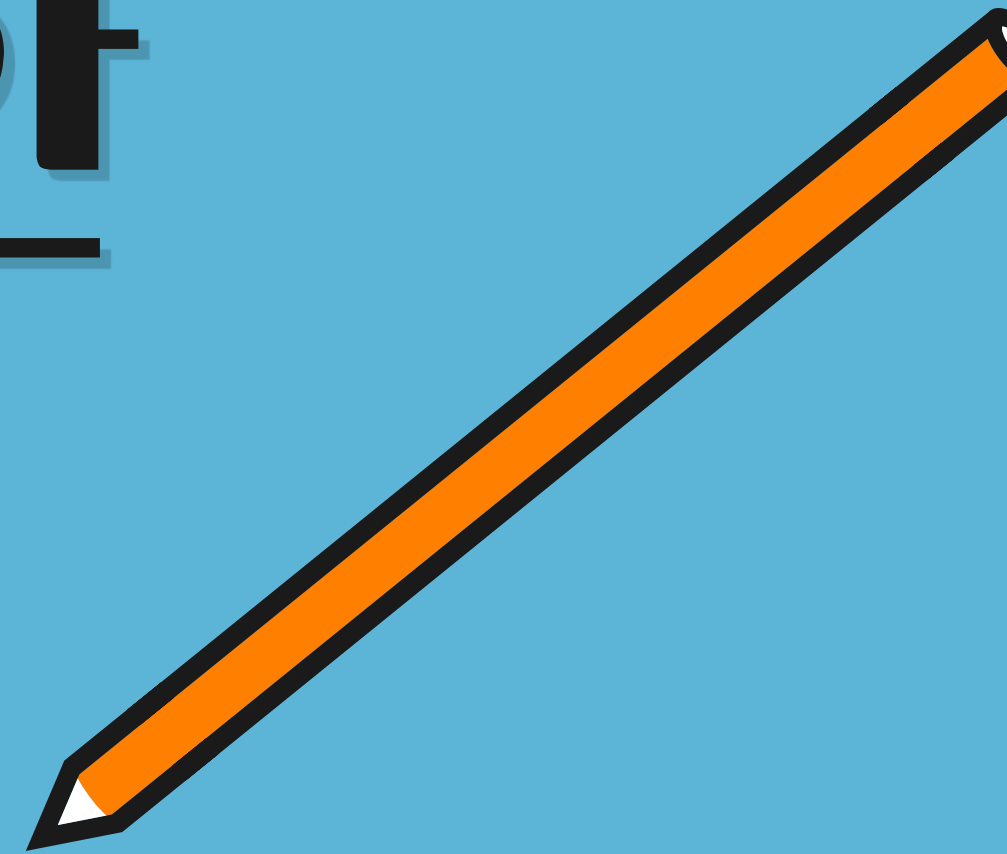
한 후

- python manage.py runserver를 통해 정상적으로 페이지가 뜨는지 확인



Ajax를 이용한

좋아요 구현

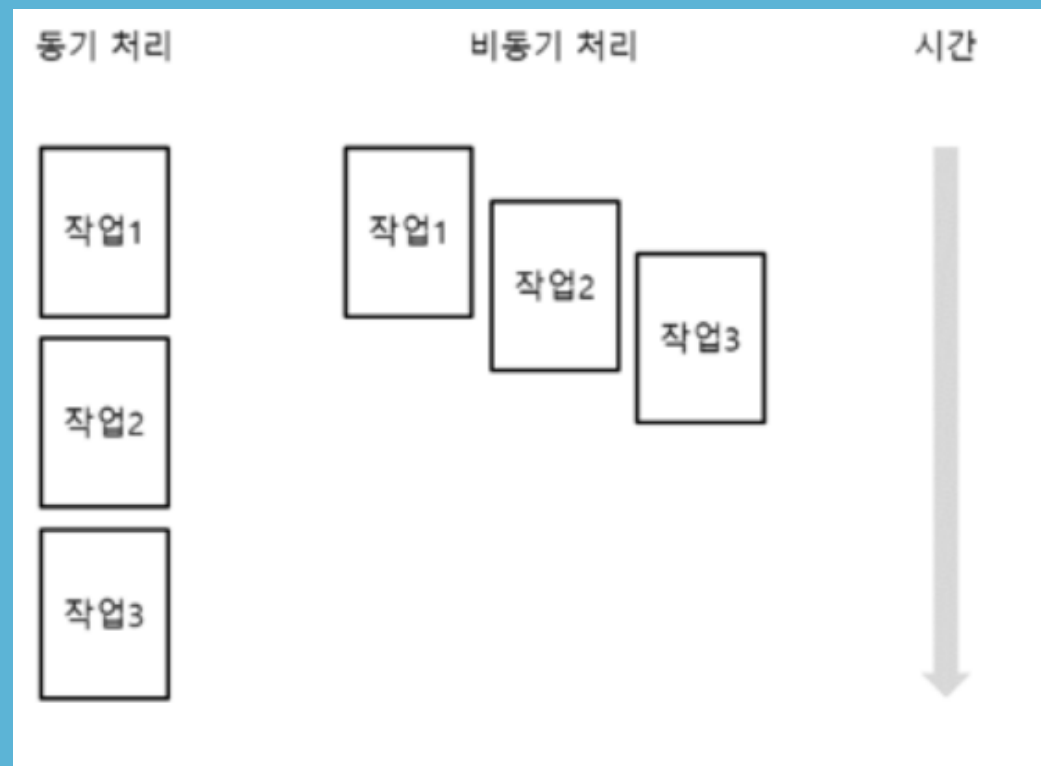


‘Ajax’란?

JavaScript의 라이브러리 중 하나이며 Asynchronous Javascript And Xml(비동기식 자바스크립트와 xml)의 약자이다.

전체 페이지를 새로고침 하지 않고도 페이지의 일부만을 위한 데이터를 로드하는 방식이며

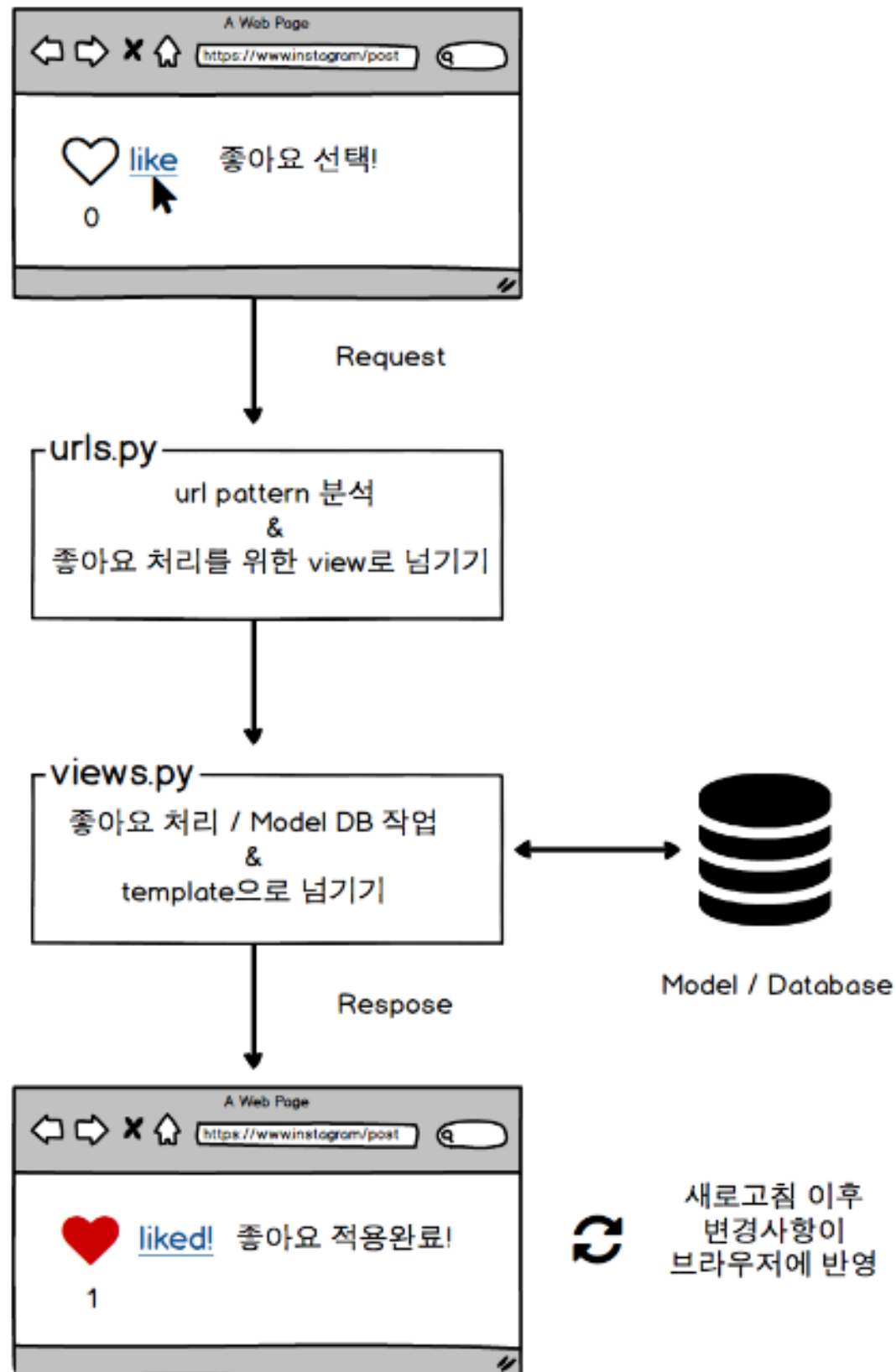
쉽게 말하면 Javascript를 통해서 서버에 데이터를 요청하는 것이다.



동기는 요청을 보낸 후 응답(결과물)을 받아야지만 다음 동작이 이루어지는 방식을 의미하며

비동기는 여러 작업을 처리하도록 예약한 뒤 작업이 끝나면 결과를 받는 방식을 의미한다. 비동기로 구성을 하게 된다면 데이터를 받아오기까지 기다리는 시간이 줄어들 것이고 처리 속도가 향상될 것이다.

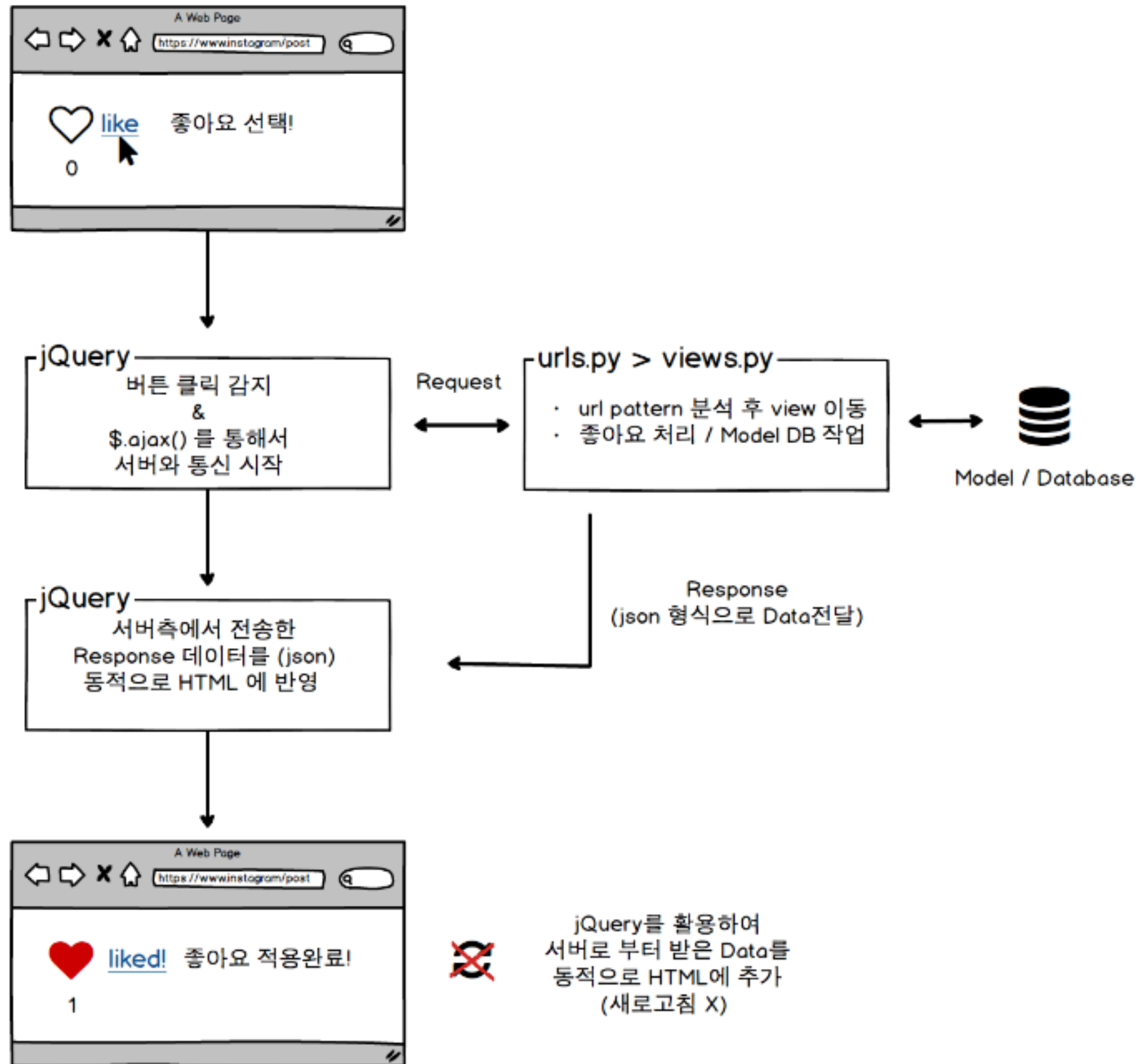
페이지 새로고침 ○ 좋아요 반영하기



기존 구현 방식대로 `views.py`에 함수를 만들고 -> `urls.py`를 지정하고 -> "좋아요" 버튼에 해당 url을 할당하는 방식으로 진행한다면 좌측과 같은 흐름도가 완성이 될 것이다.

url이 변경되어야하기 때문에 "새로고침"의 과정이 필수적이다.

페이지 새로고침 X
좋아요 반영하기
(jQuery / Ajax 활용)



하지만 Ajax를 이용하여 비동기 처리를 구현한다면, 처리 흐름은 좌측과 같아진다. 즉, 동적으로 HTML에 추가함에 의해 새로고침이 필요없어지는 것이다.

+ -> JQuery는 Ajax와 관련된 여러 편리한 기능을 제공하는 라이브러리로 보통 Ajax는 JQuery와 함께 사용하게 된다.

+ -> JSON은 JavaScript Object Notation의 약자로 자바 스크립트 객체 표기법을 말한다. key와 value 값이 쌍으로 구성된 형태의 객체 표기법으로 클라이언트와 서버 사이에 정보를 교환하기 위한 목적으로 사용된다.

{key1: value1, key2: value2, ... }

PART 02

기존 프로젝트 분석

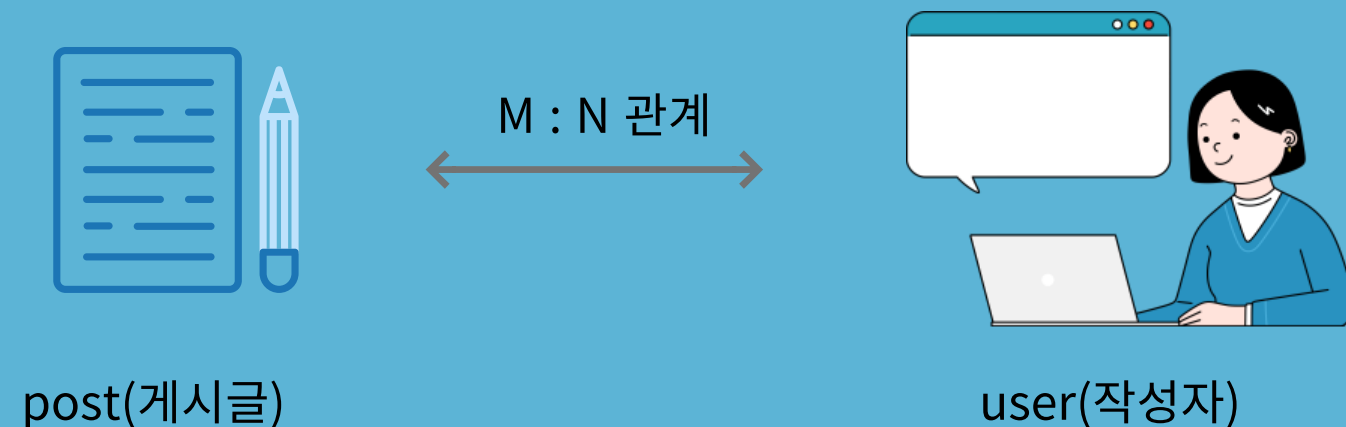
```
class Post(models.Model):
    title = models.CharField(max_length=50, null=False)
    content = models.TextField()
    view_count = models.IntegerField(default = 0)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    image = models.ImageField(upload_to='images/', null=True)
    user = models.ForeignKey(User, on_delete=models.CASCADE, null=True)
    like_user_set = models.ManyToManyField(User, blank=True, related_name='likes_user_set', through='Like')
    # dislike_user_set = models.ManyToManyField(User, blank=True, related_name='dislikes_user_set', through='Dislike')

    @property
    def like_count(self):
        return self.like_user_set.count()

    # @property
    # def dislike_count(self):
    #     return self.dislike_user_set.count()
```

- title = 게시글 제목
 - content = 게시글 내용
 - view_count = 조회수
 - created_at = 작성 날짜
 - updated_at = 수정날짜
 - image = 이미지
 - user = 작성자
 - like_user_set = 해당 게시글을 좋아요 한 user 집합
 - like_count = like_user_set에 포함된 학생의 수를 count해주는 메소드.
- @property 데코레이터로 인해 메소드를 마치 필드인 것 처럼 취급할 수 있게 해준다.
- 즉, html template에서
< .. src = {{post.like_user_set}}>와 같이 불러올 수 있게 된다.

=> like_user_set의 ManyToManyField?



User는 여러 개의 게시물을 좋아요 할 수 있고, Post는 여러 명의 사용자에게 좋아요를 받을 수 있으므로 N:N 관계를 갖는다. Post와 ManyToMany 관계를 맺는 User모델을 파라미터로 넣어주고, 아무도 좋아요를 하지 않을 수도 있으니 'blank=True' 옵션을 걸어주는 것이다.

ManyToMany field를 사용할 때 realated_name을 지정해준다면 해당 게시물을 좋아요한 사용자를 얻기 위해 "some_post.likes_user_set.all()"처럼 사용할 수 있게 된다.

PART 02

기존 프로젝트 분석

```
class Like(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    post = models.ForeignKey(Post, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        unique_together = (('user', 'post'))

    def __str__(self):
        return f'Like {self.user.username} {self.post.title}'
```

- user = 좋아요를 누른 사용자
- post = 좋아요가 눌린 대상

- user, post 세트를 고유한 필드 set으로 지정하기 위해
class Meta 코드를 추가

=> Meta 클래스는 모델에 대한 다양한 사항을 정의하는데 사용하는 선택사항

PART 03

items/views.py 수정

사용할 모듈 불러오기

```
from django.views.decorators.http import require_POST
from django.http import HttpResponse
import json
```

```
# 2. 사용할 모듈 불러오기
# 2-1 POST 형식의 HTTP 통신만 받기
from django.views.decorators.http import require_POST
# 2-2 response를 변환하는 가장 가본 함수, html 파일, 이미지 등 다양한 응답
from django.http import HttpResponse
# 2-3 딕셔너리를 json 형식으로 바꾸기 위해
import json

def main(request):
    items = Post.objects.all()
```


items/views.py 수정

@require_POST POST 형식의 HTTP 통신이고

@login_required login이 되어있다면

```
def like_toggle(request, post_id):
```

```
    post = get_object_or_404(Post, pk = post_id)
```

```
    post_like, post_like_created = Like.objects.get_or_create(user=request.user, post=post)
```

해당 메소드는 (object, created) 튜플 형식의 반환값을 내놓는데,
created=True라는 뜻은 인스턴스가 해당 메소드에 의해 새로 생성되었다
는 뜻이고

created= False는 인스턴스가 기존에 존재하여, DB에서 꺼내왔다는 의미

```
    if not post_like_created:
```

```
        post_like.delete()
```

```
        result = "like_cancel"
```

```
    else:
```

```
        result = "like"
```

```
    context = {
```

```
        "like_count" : post.like_count,
```

```
        "result" : result
```

```
    }
```

```
    return HttpResponse(json.dumps(context), content_type = "application/json")
```

```
#삭제하기
```

```
def delete(request, post_id):
```

```
    post = get_object_or_404(Post, pk=post_id)
```

```
    post.delete()
```

```
    return redirect('main')
```

```
# 3. like_toggle 함수 작성하기
```

```
@require_POST # POST 형식의 HTTP 통신이고
```

```
@login_required # 사용자가 login 한 상태이면
```

```
def like_toggle(request, post_id):
```

```
    post = get_object_or_404(Post, pk = post_id)
```

```
    post_like, post_like_created = Like.objects.get_or_create(user=request.user, post=post)
```

```
    if not post_like_created:
```

```
        post_like.delete()
```

```
        result = "like_cancel"
```

```
    else:
```

```
        result = "like"
```

```
    context = {
```

```
        "like_count" : post.like_count,
```

```
        "result" : result
```

```
    }
```

```
    return HttpResponse(json.dumps(context), content_type = "application/json")
```

items/urls.py 수정

```
path('like_toggle/<int:post_id>/', views.like_toggle, name="like_toggle"),
```

```
from . import views

app_name="items"
urlpatterns = [
    path('', views.main, name="main"),
    path('new/', views.new, name="new"),
    path('create/', views.create, name="create"),
    path('show/<int:post_id>/', views.show, name="show"),
    path('delete/<int:post_id>/', views.delete, name="delete"),

    # 1. like_toggle url 연결하기
    path('like_toggle/<int:post_id>/', views.like_toggle, name="like_toggle"),
]
```

items/templates/items/show.html 수정

- 좋아요 버튼 노출

```
<span href="#" class="like" name="{{post.id}}" value="Like">
  {% if user in post.like_user_set.all %} 해당 user가 해당 post를 좋아요 한 user라면
    <div class="alt-service-icon">
      <i class="fas fa-heart" id="like-icon" style="color:red;"></i>
    </div>
    딱 찬 하트를 보여주고
  {% else %} 아니라면
    <div class="alt-service-icon">
      <i class="far fa-heart" id="like-icon" style="color:red;"></i>
    </div>
    빈 하트를 보여준다
  {% endif %}
  <span id="count-{{ post.id }}" class="alt-services-title font-alt">
    좋아요
    {% if post.like_count == 0 %}
      0개
    {% else %}
      {{ post.like_count }}개
    {% endif %}
  </span>
</span>
```

```
{% if post.user == user %}
  <a href="{% url 'items:delete' post.pk %}"> 삭제하기 </a>
{% endif %}

<!--4. 좋아요 버튼 노출-->

<span href="#" class="like" name="{{post.id}}" value="Like">

  {% if user in post.like_user_set.all %}
    <div class="alt-service-icon">
      <i class="fas fa-heart" id="like-icon" style="color:red;"></i>
    </div>
  {% else %}

    <div class="alt-service-icon">
      <i class="far fa-heart" id="like-icon" style="color:red;"></i>
    </div>

  {% endif %}

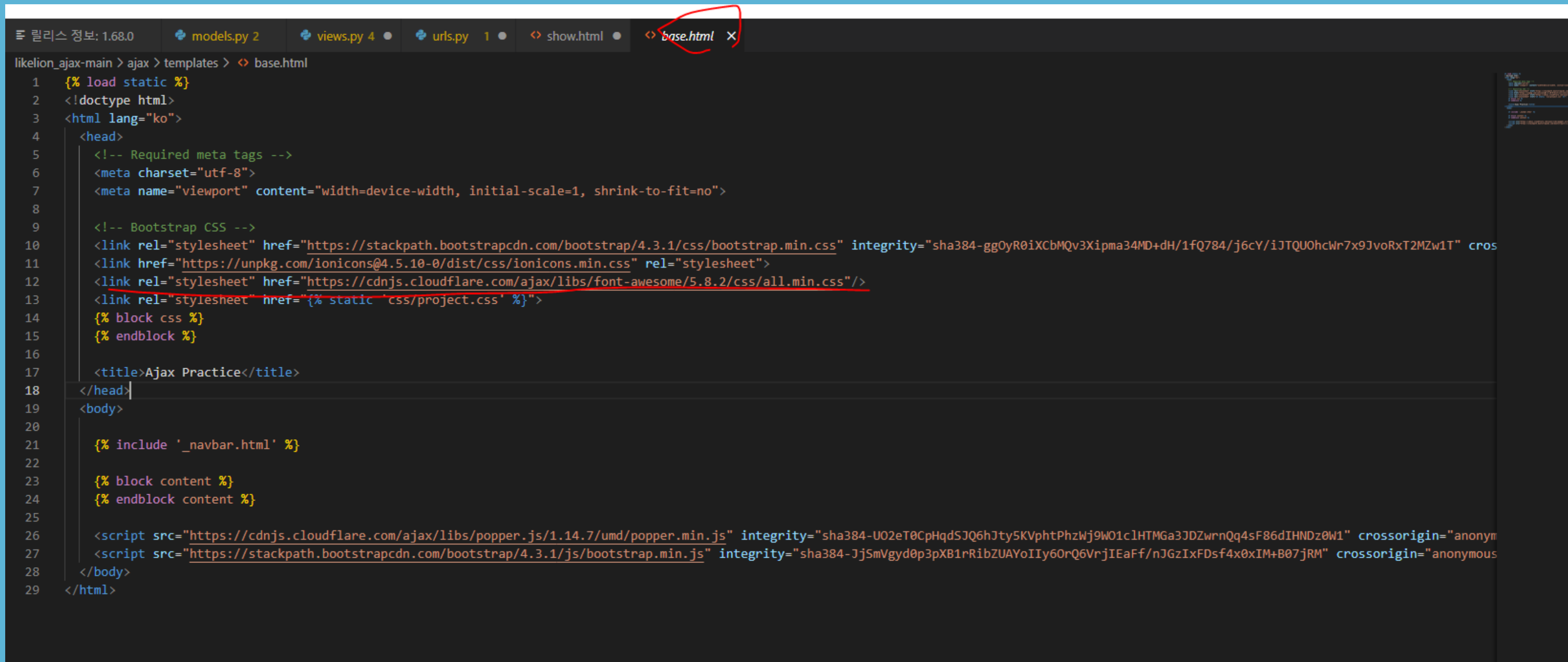
  <span id="count-{{ post.id }}" class="alt-services-title font-alt">
    좋아요
    {% if post.like_count == 0 %}
      0개
    {% else %}
      {{ post.like_count }}개
    {% endif %}
  </span>
</span>

</div>
```

PART 03

items/templates/items/show.html 수정 - 좋아요 버튼 노출

빨간 하트, 빈 하트가 화면에 노출되는 이유는 "font-awesome"을 사용했기 때문에



```
likelion_ajax-main > ajax > templates > <> base.html
1  {% load static %}
2  <!doctype html>
3  <html lang="ko">
4  <head>
5      <!-- Required meta tags -->
6      <meta charset="utf-8">
7      <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
8
9      <!-- Bootstrap CSS -->
10     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
11     <link href="https://unpkg.com/ionicons@4.5.10-0/dist/css/ionicons.min.css" rel="stylesheet">
12     <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.8.2/css/all.min.css"/>
13     <link rel="stylesheet" href="{% static 'css/project.css' %}">
14     {% block css %}
15     {% endblock %}
16
17     <title>Ajax Practice</title>
18 </head>
19 <body>
20
21     {% include '_navbar.html' %}
22
23     {% block content %}
24     {% endblock content %}
25
26     <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-U02eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>
27     <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEaEfF/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
28 </body>
29 </html>
```

PART 03

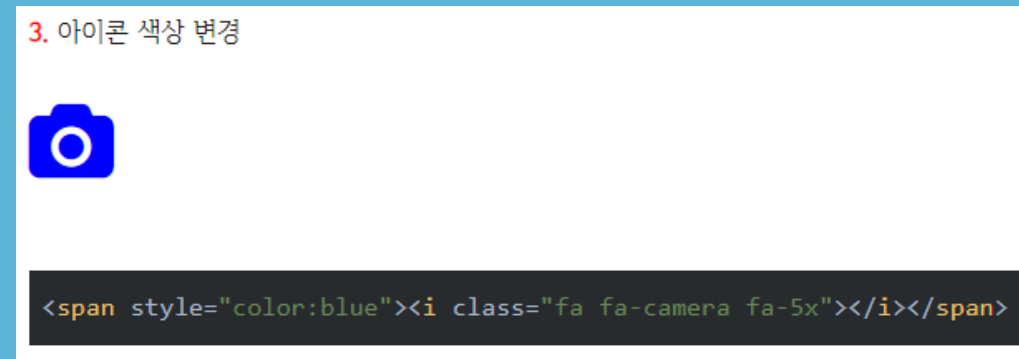
items/templates/items/show.html 수정 - 좋아요 버튼 노출

font awesome은 웹 아이콘 폰트를 모아놓은 라이브러리.

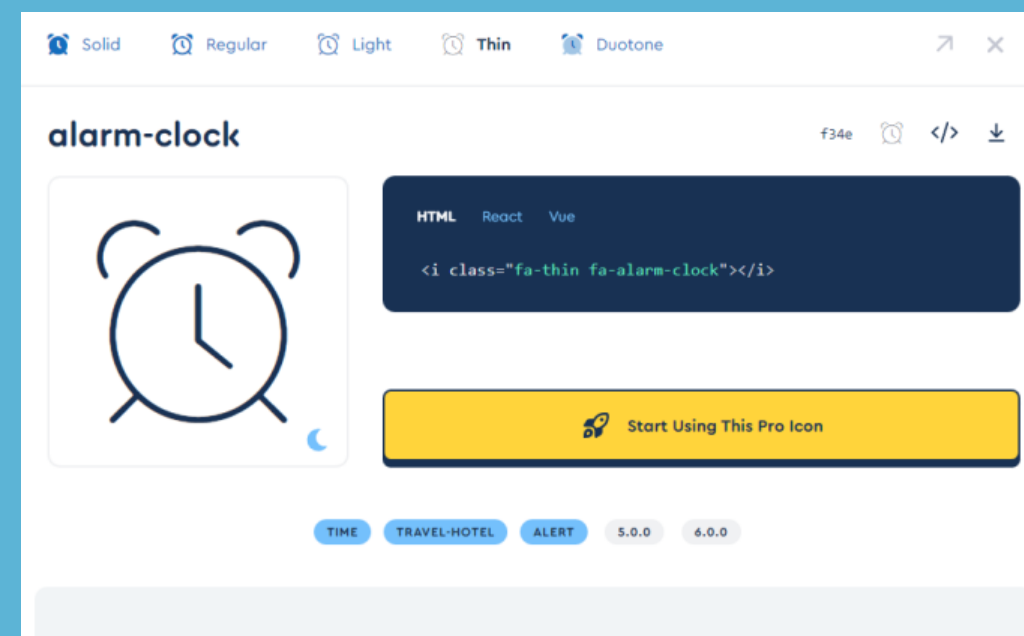
위와 같이 헤더에 코드를 삽입하고 특정 클래스 명으로 호출하여 사용할 수 있다.



<https://coding-factory.tistory.com/192>
=> font-awesome 아이콘 사용 방법



<https://fontawesome.com/icons>
=> font-awesome 사이트



items/templates/items/show.html 수정

- ajax 적용하기

```
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script>
    $(".like").click(function(){ like 버튼 클릭 감지
    var pk=$(this).attr('name')
    $.ajax({ ajax로 서버와 통신
    type:'POST',
    url:"{% url 'items:like_toggle' post.pk %}",
    data: {'pk':pk, 'csrfmiddlewaretoken': '{{csrf_token}}'},
    dataType:'json',
    success:function(response){
        if(response.like_count==null){ like 요청이 실패한다면
            alert('로그인이 필요합니다.');" 경고창을 띄우고
            window.location.replace('/accounts/login/'); 로그인 페이지로 이동
        }
        else{
            if(response.result=='like'){ result='like'이면 (새로 like를 생성하는 것이면)
                $('#like-icon').removeClass();
                $('#like-icon').addClass('fas fa-heart'); icon을 짝 찬 하트로 변경
            }
            else{ 그게 아니면
                $('#like-icon').removeClass();
                $('#like-icon').addClass('far fa-heart'); icon을 빈하트로 변경
            }
            $('#count-{{post.id}}').html('좋아요'+response.like_count+"개"); 좋아요 개수 업데이트
        }
    },
    error:function(request, status, error){
        alert("로그인이 필요합니다.")
        alert('code:'+request.status+'\n'+message:'+request.responseText+"\n"+"error:"+error);
    },
    })
})
</script>
```

```
<!-- 5. ajax 적용하기-->
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
<script>
    $(".like").click(function(){
    var pk=$(this).attr('name')
    $.ajax({
    type:'POST',
    url:"{% url 'items:like_toggle' post.pk %}",
    data: {'pk':pk, 'csrfmiddlewaretoken': '{{csrf_token}}'},
    dataType:'json',
    success:function(response){
        if(response.like_count==null){
            alert('로그인이 필요합니다.');"
            window.location.replace('/accounts/login/');"
        }
        else{
            if(response.result=='like'){
                $('#like-icon').removeClass();
                $('#like-icon').addClass('fas fa-heart');"
            }
            else{
                $('#like-icon').removeClass();
                $('#like-icon').addClass('far fa-heart');"
            }
            $('#count-{{post.id}}').html('좋아요'+response.like_count+"개");
        }
    },
    error:function(request, status, error){
        alert("로그인이 필요합니다.")
        alert('code:'+request.status+'\n'+message:'+request.responseText+"\n"+"error:"+error);
    },
    })
})
</script>

{% endblock %}
```

PART 03

결과 확인

<상세보기 페이지>

글 제목 : adf

글 내용 : adf

작성 시각 : 2022년 7월 18일 3:01 오후

조회 수: 4

작성자 : alswn11

[삭제하기](#)



좋아요1개

로그인 한 경우

127.0.0.1:8000의 메시지

로그인이 필요합니다.

확인

<상세보기 페이지>

글 제목 : adf

글 내용 : adf

작성 시각 : 2022년 7월 18일 3:01 오후

조회 수: 5

작성자 : alswn11



좋아요 1개

로그인 안 한 경우

PART 04

내 좋아요 목록 만들기 - items/views.py 수정

```
# 4. my_like 함수 작성하기
def my_like(request, user_id):
    user = User.objects.get(id = user_id)
    like_list = Like.objects.filter(user = user)
    context = {
        'like_list' : like_list,
    }
    return render(request, 'items/my_like.html', context)
```

해당 user가 좋아요 한 Like 객체 목록 가져오기

```
# 4. my_like 함수 작성하기
def my_like(request, user_id):
    user = User.objects.get(id = user_id)
    like_list = Like.objects.filter(user = user)
    context = {
        'like_list' : like_list,
    }
    return render(request, 'items/my_like.html', context)
```

LIKE

>>>>>>

user = 좋아요 한 사용자
post = 좋아요 된 게시물

PART 04

내 좋아요 목록 만들기 - items/urls.py 수정

```
path('my_like/<int:user_id>', views.my_like, name='my_like'),
```

```
# 2. my_like url 연결하기
```

```
path('my_like/<int:user_id>', views.my_like, name='my_like'),
```

내 좋아요 목록 만들기

- ajax/templates/_navbar.html 수정

```
<li class="nav-item">  
  <a class="nav-link" href="{% url 'items:my_like' request.user.id %}">내 좋아요 목록</a>  
</li>
```

```
    <a class="nav-link" href="{% url 'account_logout' %}">로그아웃</a>  
</li>  
<!--좋아요 한 목록 링크 추가-->  
<li class="nav-item">  
  <a class="nav-link" href="{% url 'items:my_like' request.user.id %}">내 좋아요 목록</a>  
</li>  
{% else %}  
  <li class="nav-item">  
    <a class="nav-link" href="{% url 'account_login' %}">로그인</a>
```

PART 04

내 좋아요 목록 만들기

- items/templates/items/my_like.html 추가

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
    <div class="container">
        <p style="font-weight:bold;"> 내 좋아요 목록 </p>
        <div class="row">
            {% for like in like_list %}
                <div class="col-md-3 col-6 item_list__card">
                    <div class="card">
                        {% if like.post.image %}
                            <a href="#">
                                
                            </a>
                        <div class="card-body">
                            <h5 class="card-title">{{ like.post.title }}</h5>
                            <a href="{% url 'items:show' like.post.pk %}" class="btn btn-outline-primary">보러가기</a>
                        </div>
                        {% else %}
                            <div class="card-body">
                                <h5 class="card-title">{{ like.post.title }}</h5>
                                <a href="{% url 'items:show' like.post.pk %}" class="btn btn-outline-primary">보러가기</a>
                            </div>
                        {% endif %}
                    </div>
                </div>
            {% endfor %}
        </div>
    </div>
{% endblock content %}
```

[Ajax Practice](#) [Home](#) [alswn1212님](#) [새글 쓰기](#) [로그아웃](#) [내 좋아요 목록](#)

👉 내 좋아요 목록 👉

adf

보러가기

PART 05

과제



Check Point 01

- ☐ 싫어요 버튼 만들기
(ajax, font-awesome
활용)

Check Point 02

- ☐ 개인 sns-project에 좋
아요, 싫어요 버튼 추가
하기

Check Point 03

- ☐ [프론트엔드]
javascript5 듣기
- ☐ [기획 디자인]
온라인 강의 수강