

Authentication

# 11번째 세션

NEXT X LIKELION 서인재

# 세션 목표

인증

1. 회원가입 / 로그인 / 로그아웃 기능 구현
2. 로그인 여부에 따라 다르게 보이는 navbar, 접근 가능한 페이지
3. 글 작성자만 수정, 삭제 권한 부여
4. 카카오 API를 이용한 소셜 로그인

# 오늘의 keyword

게으른 완벽주의자들을 위해서 모든 것을 제공하는 장고

## django.contrib.auth

- 장고 내장된 앱
- 기본적으로 User 모델 제공  
`from django.contrib.auth.models import User`
- 로그인 / 로그아웃 로직 제공

## django-allauth

- 장고에 따로 내장되어 있지 않고 설치 따로 진행
- 이메일 인증 기능 제공
- 소셜 로그인 기능 제공

# 개념

여러가지 활용 방법



## 이번 실습에서는?

- `django.contrib.auth` 를 활용하여 `views.py`를 작성합니다.
- `django-allauth` 를 설치하여 소셜 로그인 기능을 구현합니다.

## 실습 준비

늘 그랬듯이,,, 에러도 살짝 마주치면서

**<Session11.zip> 압축 해제 후**

**\$ cd session11**

**\$ pipenv shell**

**\$ pipenv install django**

**\$ cd project**

**\$ python manage.py makemigrations**

**\$ python manage.py migrate**

**\$ python manage.py createsuperuser**

**\$ python manage.py runserver**

# 회원가입

MTV 작성합시다

models.py

```
from django.db import models
from django.contrib.auth.models import User
```

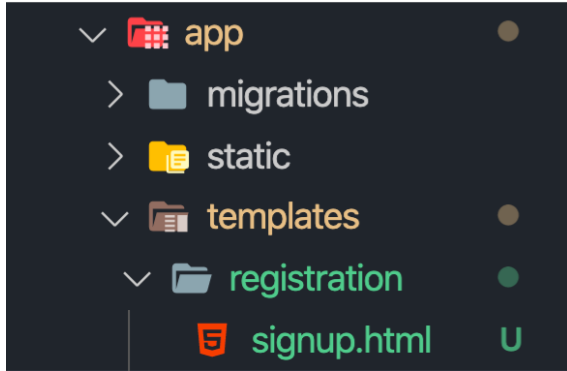
User 모델의 기본 필드

- username
- first\_name
- last\_name
- password
- email
- ...

# 회원가입

MTV 작성합니다

templates/registration/signup.html



```
signup.html U x
project > app > templates > registration > signup.html > ...
1  {% extends 'base.html' %} {% block css %}{% load static %}
2  <link rel="stylesheet" type="text/css" href="{% static 'signup.css' %}" />
3  {% endblock css %} {% block content %}
4
5  <section id="signup">
6      <form method="POST">
7          {% csrf_token %}
8          <input type="text" name="username" placeholder="아이디" />
9          <input type="password" name="password" placeholder="비밀번호" />
10         <button>회원가입하기</button>
11     </form>
12 </section>
13 {% endblock %}
14
```

signup.css 파일 만들고 연결만 해주세요

templates/base.html

```
<li class="navbar__menu__item">
  <a href="{% url 'signup' %}">회원가입</a>
</li>
```



# 회원가입

MTV 작성합시다

views.py

```
from django.contrib.auth.models import User
```

```
def signup(request):  
    if request.method == "POST":  
        username = request.POST["username"]  
        password = request.POST["password"]  
        User.objects.create_user(username=username, password=password)  
        return redirect("home")  
    return render(request, "registration/signup.html")
```

django.contrib.auth 에서 제공하는 메소드

**create\_user(username, password)** : User 객체를 생성, 저장한다.

urls.py

```
from app import views
```

```
path("registration/signup", views.signup, name="signup"),
```

# 회원가입

## 예외처리

모델의 `username`은 중복될 수 없다.

```
def signup(request):  
    if request.method == "POST":  
        username = request.POST["username"]  
        password = request.POST["password"]  
        found_user = User.objects.filter(username=username)  
        if len(found_user):  
            error = "이미 아이디가 존재합니다"  
            return render(request, "registration/signup.html", {"error": error})  
        new_user = User.objects.create_user(username=username, password=password)  
        auth.login(request, new_user)  
        return redirect("home")  
    return render(request, "registration/signup.html")
```

`filter()` : 쿼리셋 반환

쿼리셋 : 데이터베이스에서 전달받은 모델의 객체 목록

POST로 보낸 `username`이 User 모델에 이미 있다면, `user`객체가 포함된 쿼리셋을 반환합니다.

`len(found_user) > 0` : 이미 `username`이 있다

`len(found_user) = 0` : 중복되는 `username`이 없다.

# 회원가입

login()

회원가입하면서 동시에 로그인한다

```
from django.contrib.auth.models import User
from django.contrib import auth
```

```
def signup(request):
    if request.method == "POST":
        username = request.POST["username"]
        password = request.POST["password"]
        found_user = User.objects.filter(username=username)
        if len(found_user):
            error = "이미 아이디가 존재합니다"
            return render(request, "registration/signup.html", {"error": error})
        new_user = User.objects.create_user(username=username, password=password)
        auth.login(request, new_user)
        return redirect("home")
    return render(request, "registration/signup.html")
```

session을 쉽게 말해보자면..

=> 일정 시간 동안 유지되는 저장소

django.contrib.auth 에서 제공하는 login()을 사용

login(request, User객체) : 유저 정보를 session에 저장합니다.

만약 login()을 통해 유저 정보를 session에 저장하지 않는다면 ? => 페이지 이동할 때마다 로그인을 해야 합니다.

# 회원가입

## templates.html

```
{% extends 'base.html' %} {% block css %}{% load static %}
<link rel="stylesheet" type="text/css" href="{% static 'signup.css' %}" />
{% endblock css %} {% block content %}

<section id="signup">
  <form method="POST">
    {% csrf_token %} {% if error %}
    <p>{{error}}</p>
    {% endif %}
    <input type="text" name="username" placeholder="아이디" />
    <input type="password" name="password" placeholder="비밀번호" />
    <button>회원가입하기</button>
  </form>
</section>
{% endblock %}
```

# 로그인

MTV

회원가입과 마찬가지로 작성합니다.

templates/registration/login.html

```
{% load static %}
{% extends 'base.html' %}
{% block css %}
<link rel="stylesheet" type="text/css" href="{% static 'login.css' %}" />
{% endblock css %}
{% block content %}
<section id="login">
  <form method="POST">
    {% csrf_token %}
    {% if error %}
    <p>{{error}}</p>
    {% endif %}
    <input type="text" name="username" placeholder="아이디" />
    <input type="password" name="password" placeholder="비밀번호" />
    <button>로그인</button>
  </form>
</section>
{% endblock %}
```

templates/base.html

```
<li class="navbar__menu__item">
  <a href="{% url 'login' %}">로그인</a>
</li>
```

urls.py

```
path("registration/login", views.login, name="login"),
```

# 로그인

MTV

views.py

```
def login(request):  
    if request.method == "POST":  
        username = request.POST["username"]  
        password = request.POST["password"]  
        user = auth.authenticate(request, username=username, password=password)  
        if user is not None:  
            auth.login(request, user)  
            return redirect("home")  
        error = "아이디 또는 비밀번호가 틀립니다"  
        return render(request, "registration/login.html", {"error": error})  
  
    return render(request, "registration/login.html")
```

authenticate() : User인증 함수. 자격 증명이 유효한 경우 User객체를, 그렇지 않은 경우 None을 반환합니다.

User객체를 반환하면 => login()을 사용하여 유저의 정보를 session에 저장하자.

None 반환하면 => 에러메시지를 생성하자.

# 로그아웃

MTV

views.py

```
def logout(request):  
    auth.logout(request)  
  
    return redirect("home")
```

templates/base.html

```
<li class="navbar__menu__item">  
    <a href="{% url 'logout' %}">로그아웃</a>  
</li>
```

urls.py

```
path("registration/logout", views.logout, name="logout"),
```

# 세션 목표

인증

1. ~~회원가입 / 로그인 / 로그아웃 기능 구현~~
2. 로그인 여부에 따라 다르게 보이는 navbar, 접근 가능한 페이지
3. 글 작성자만 수정, 삭제 권한 부여
4. 카카오 API를 이용한 소셜 로그인

이제 여러분은 회원가입/로그인/로그아웃을 구현할 수 있습니다.

django.contrib.auth에게 고맙다고 합시다.

공식문서에서 django.contrib.auth 를 통해 views.py를 작성하는 방법을 다 알려줍니다.

공식문서 확인 : <https://docs.djangoproject.com/en/4.0/topics/auth/default/#>



# 로그인 여부에 따라 다르게 보이는 navbar

templates/base.html

```
<li class="navbar_menu_item"><a href="{% url 'new' %}">New</a></li>
<li class="navbar_menu_item">
  <a href="{% url 'home' %}">Home</a>
</li>
{% if user.is_authenticated %}
<span class="navbar_menu_item">안녕, {{user.username}}</span>
<li class="navbar_menu_item">
  <a href="{% url 'logout' %}">로그아웃</a>
</li>
{% else %}
<li class="navbar_menu_item">
  <a href="{% url 'login' %}">로그인</a>
</li>
<li class="navbar_menu_item">
  <a href="{% url 'signup' %}">회원가입</a>
</li>
{% endif %}
</ul>
```

# 로그인 여부에 따라 다르게 보이는 navbar

## 템플릿에서 User 사용하기

뷰함수에서 User 객체를 전달하지 않아도 django.contrib.auth 기능으로 인해 템플릿에서 User 객체를 사용할 수 있다.

- `user.is_authenticated` - 현재 사용자가 인증되었는지 여부 (로그인한 상태라면 `true`, 로그아웃 상태라면 `false`)
- `user.is_anonymous` - `is_authenticated`의 반대 경우 (로그인한 상태라면 `false`, 로그아웃 상태라면 `true`)
- `user.username` - 사용자명 (사용자 ID)
- `user.is_superuser` - 사용자가 슈퍼유저인지 여부

# 로그인 여부에 따라 접근 가능한 페이지

로그인한 유저만 새글을 작성할 수 있고, 글의 상세정보에 접근할 수 있도록 만들어 봅시다.

views.py

```
from django.contrib.auth.decorators import login_required
```

```
@login_required(login_url="/registration/login")
def new(request):
    if request.method == "POST":
        title = request.POST["title"]
        content = request.POST["content"]
        new_post = Post.objects.create(
            title=title, content=content, author=request.user
        )
        return redirect("detail", new_post.pk)

    return render(request, "new.html")
```

```
@login_required(login_url="/registration/login")
def detail(request, post_pk):
    post = Post.objects.get(pk=post_pk)
    print(type(post.author), "post.author")
    if request.method == "POST":
        content = request.POST["content"]
        Comment.objects.create(post=post, content=content, author=request.user)

    return redirect("detail", post_pk)
    return render(request, "detail.html", {"post": post})
```

@login\_required(login\_url="/registration/login")

로그인하지 않은 유저면 registration/login으로 redirect시킨다.

# 로그인 여부에 따라 접근 가능한 페이지

## URL 의미

로그인하지 않은 상황에서 새글 작성하기 위해 새 글 작성 페이지로 가면 @login\_required에 의해 로그인 페이지로 redirect 된다.

이때 아래 URL이 다음과 같아진다.

```
127.0.0.1:8000/registration/login?next=/new/
```

의미: 로그인이 완료되면 원래 사용자가 가고자 했던 새글 작성 페이지로 보내주겠다는 의미다.

단순히 redirect("home")으로 하지말고 url의 의도에 맞게 적용해봅시다.

```
def login(request):
    if request.method == "POST":
        username = request.POST["username"]
        password = request.POST["password"]
        user = auth.authenticate(request, username=username, password=password)
        if user is not None:
            auth.login(request, user)
            # return redirect("home")
            return redirect(request.GET.get("next", "/"))
        error = "아이디 또는 비밀번호가 틀립니다"
        return render(request, "registration/login.html", {"error": error})

    return render(request, "registration/login.html")
```

장고에서 request 내용을 뽑아내고 싶을 때,

request.GET : request 데이터를 딕셔너리 형태로 변환

dict.get(key) : 딕셔너리 관련 함수로써 key값을 통해 value를 얻어냄

# 세션 목표

인증

1. ~~회원가입 / 로그인 / 로그아웃 기능 구현~~
2. ~~로그인 여부에 따라 다르게 보이는 navbar~~
3. 글 작성자만 수정, 삭제 권한 부여
4. 카카오 API를 이용한 소셜 로그인

# 글 작성자만 수정, 삭제 권한 부여

models.py

ForeignKey로 사용하기 위해 User import

글, 댓글에 유저 정보를 저장하기 위해 author 필드 추가

```
You, 2 seconds ago | 1 author (You)
1 from django.db import models
2 from django.contrib.auth.models import User
3
4 # Create your models here.
You, 2 seconds ago | 1 author (You)
5 class Post(models.Model):
6     title = models.CharField(max_length=200)
7     content = models.TextField()
8     author = models.ForeignKey(User, models.CASCADE, related_name="posts")
9
10     def __str__(self):
11         return self.title
12
13
14
You, 2 seconds ago | 1 author (You)
15 class Comment(models.Model):
16     post = models.ForeignKey(Post, on_delete=models.CASCADE, related_name="comments")
17     content = models.TextField()
18     author = models.ForeignKey(User, models.CASCADE, related_name="comments")
19
20
21
22
```

# 글 작성자만 수정, 삭제 권한 부여

models.py

모델에 변경사항이 있다면? `python manage.py makemigrations`  
`python manage.py migrate`

It is impossible to add a non-nullable field 'author' to comment without specifying a default. This is because the database needs something to populate existing rows.

Please select a fix:

- 1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
- 2) Quit and manually define a default value in models.py.

Select an option:

이미 등록되어 있는 게시물에 author가 저장되어야 하는데 어떤 값을 넣어야 할 지 몰라서 생깁니다.

## Solution

기존 게시물에 추가될 author에 임의의 계정 정보를 추가합니다.

1번을 누르면 계속 눌러줍니다.

완료되면 `python manage.py migrate` 하여 DB에 변경사항을 적용합니다.

# 글 작성자만 수정, 삭제 권한 부여

views.py

새 글 또는 댓글을 작성할 때 유저의 정보가 담겨야 합니다.

views.py 를 수정해줍니다.

```
@login_required(login_url="/registration/login")
def new(request):
    if request.method == "POST":
        title = request.POST["title"]
        content = request.POST["content"]
        new_post = Post.objects.create(
            title=title, content=content, author=request.user
        )
        return redirect("detail", new_post.pk)

    return render(request, "new.html")
```

```
@login_required(login_url="/registration/login")
def detail(request, post_pk):
    post = Post.objects.get(pk=post_pk)
    print(type(post.author), "post.author")
    if request.method == "POST":
        content = request.POST["content"]
        Comment.objects.create(post=post, content=content, author=request.user)

    return redirect("detail", post_pk)
    return render(request, "detail.html", {"post": post})
```

request.user 를 통해 User객체에 접근할 수 있습니다.



# 글 작성자만 수정, 삭제 권한 부여

templates

detail.html

```
{% extends "base.html" %} {% block content %}
<section id="detail">
  <div>
    You, 3 days ago • 세션 초기세팅 ...
    <div>{{ post.title }}</div>
    <div>{{ post.content }}</div>
  </div>
  <a href="{% url 'home' %}">홈으로</a>
  {% if user.is_authenticated and post.author.pk == user.pk %}
  <a href="{% url 'edit' post.pk %}">수정하기</a>
  <a href="{% url 'delete' post.pk %}">삭제하기</a>
  {% endif %}
  <!-- -->
  {% for comment in post.comments.all %}
  <li>{{ comment.content }}</li>
  {% if user.is_authenticated and comment.author.pk == user.pk %}
  <a href="{% url 'delete_comment' post.pk comment.pk %}">댓글삭제</a>
  {% endif %} {% endfor %}
  <!-- -->
  {% if user.is_authenticated %}
  <form method="POST">
    {% csrf_token %}
    <input type="text" name="content" placeholder="댓글을 입력하세요" />
    <button type="submit">댓글 쓰기</button>
  </form>
  {% endif %}
</section>
{% endblock content %}
```

```
{% if user.is_authenticated and post.author.pk == user.pk %}
```

user.is\_authenticated : 현재 사용자(User 객체)가 로그인 되었는지

post.author.pk : post.author를 하면 User 모델을 참조하고, 그 모델에서의 pk

user.pk : User 객체의 pk

댓글 삭제의 경우:

post.author.pk 가 아니라 comment.author.pk로 접근해야 합니다.

# 세션 목표

인증

1. ~~회원가입 / 로그인 / 로그아웃 기능 구현~~
2. ~~로그인 여부에 따라 다르게 보이는 navbar~~
3. ~~글 작성자만 수정, 삭제 권한 부여~~
4. 카카오 API를 이용한 소셜 로그인

# 카카오 API를 이용한 소셜 로그인

## django-allauth

장고에 내장된 기능이 아니기 때문에 설치를 해줘야 합니다. 공식문서에서 설치방법과 코드가 잘 나와있으니 잘 복붙해봅시다.

공식 문서: <https://django-allauth.readthedocs.io/en/latest/installation.html>

Step1. 패키지 설치

```
$ pipenv install django-allauth
```

# 카카오 API를 이용한 소셜 로그인

## django-allauth

### Step2. settings.py 설정

```
AUTHENTICATION_BACKENDS = [  
    # Needed to login by username in Django admin, regardless of `allauth`  
    "django.contrib.auth.backends.ModelBackend",  
    # `allauth` specific authentication methods, such as login by e-mail  
    "allauth.account.auth_backends.AuthenticationBackend",  
]  
# Application definition  
  
INSTALLED_APPS = [  
    "django.contrib.admin",  
    "django.contrib.auth",  
    "django.contrib.contenttypes",  
    "django.contrib.sessions",  
    "django.contrib.messages",  
    "django.contrib.staticfiles",  
    "app",  
    "django.contrib.sites",  
    "allauth",  
    "allauth.account",  
    "allauth.socialaccount",  
    "allauth.socialaccount.providers.kakao",  
    "allauth.socialaccount.providers.google",  
]  
  
SITE_ID = 1  
  
LOGIN_REDIRECT_URL = "/"
```

# 카카오 API를 이용한 소셜 로그인

django-allauth

Step3. urls.py 설정

```
from django.contrib import admin
from django.urls import path, include
from app import views

urlpatterns = [
    path("admin/", admin.site.urls),
    path("", views.home, name="home"),
    path("new/", views.new, name="new"),
    path("detail/<int:post_pk>", views.detail, name="detail"),
    path("edit/<int:post_pk>", views.edit, name="edit"),
    path("delete/<int:post_pk>", views.delete, name="delete"),
    path(
        "delete_comment/<int:post_pk>/<int:comment_pk>",
        views.delete_comment,
        name="delete_comment",
    ),
    path("registration/signup", views.signup, name="signup"),
    path("registration/login", views.login, name="login"),
    path("registration/logout", views.logout, name="logout"),
    path("accounts/", include("allauth.urls")),
]
```

추가된 테이블 확인

SITES		
Sites	+ Add	Change
SOCIAL ACCOUNTS		
Social accounts	+ Add	Change
Social application tokens	+ Add	Change
Social applications	+ Add	Change

Step4. \$ python manage.py migrate

# 카카오 API를 이용한 소셜 로그인

django-allauth

Step5. Sites 수정

Change site

**http://127.0.0.1:8000**

Domain name:

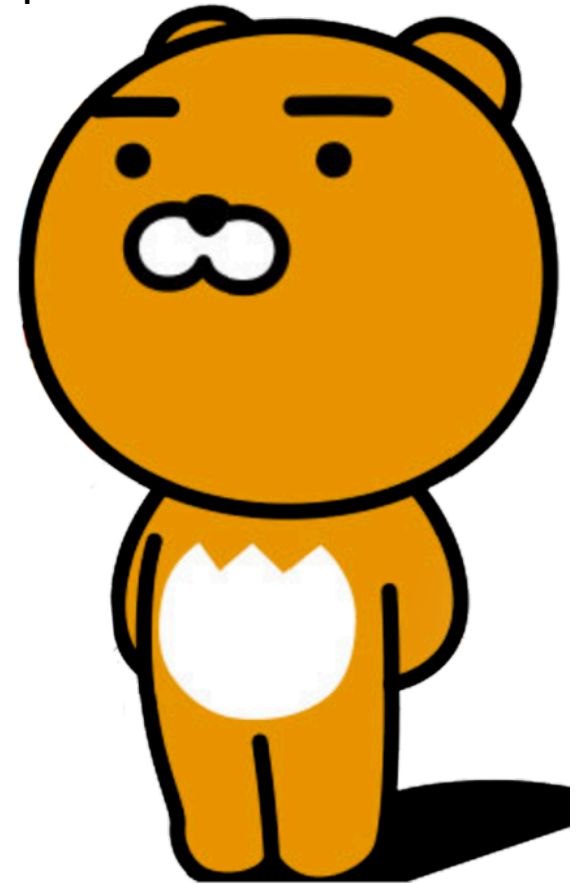
Display name:

[Delete](#)

# 카카오 API를 이용한 소셜 로그인

django-allauth

이제 카카오톡 연결해봅시다



# 카카오 API를 이용한 소셜 로그인

django-allauth

Step1. <https://developers.kakao.com/> 접속

Step2. 회원가입/로그인

Step3. 내 어플리케이션

Step4. 어플리케이션 추가하기



# 카카오 API를 이용한 소셜 로그인

django-allauth

애플리케이션을 추가해줍니다.

The screenshot shows the Django Admin interface for managing social applications. A modal titled '애플리케이션 추가하기' (Add Application) is open. The modal contains the following fields and options:

- 앱 아이콘 (App Icon):** Includes an '이미지 업로드' (Image Upload) button and a '파일 선택' (File Select) button. Below these, it specifies supported formats: 'JPG, GIF, PNG' and a size limit: '권장 사이즈 128px, 최대 250KB'.
- 앱 이름 (App Name):** A text input field with the placeholder text '내 애플리케이션 이름' (My Application Name).
- 사업자명 (Business Name):** A text input field with the placeholder text '사업자 정보와 동일한 이름' (Same as business information).

Below the input fields, there are two lines of text providing instructions:

- 입력된 정보는 사용자가 카카오 로그인을 할 때 표시됩니다.
- 정보가 정확하지 않은 경우 서비스 이용이 제한될 수 있습니다.

At the bottom of the modal, there are two buttons: '취소' (Cancel) and '저장' (Save).

# 카카오 API를 이용한 소셜 로그인

django-allauth

REST API 키는 나중에 어드민 페이지에 등록해줄겁니다.

내 애플리케이션 > 앱 설정 > 요약 정보

앱 설정

요약 정보

일반

비즈니스

앱 키

플랫폼

팀 관리

제품 설정

카카오 로그인

동의항목

가편가인

APP

ID 463310

OWNER

Web

앱 키

네이티브 앱 키	
REST API 키	
JavaScript 키	
Admin 키	

# 카카오 API를 이용한 소셜 로그인

django-allauth

플랫폼에 들어가서 Web 부분에 도메인을 입력해주세요. 배포하게 되면 배포된 사이트의 도메인으로 설정해줍니다.

내 애플리케이션 > 앱 설정 > 플랫폼

앱 설정

요약 정보

일반

비즈니스

앱 키

**플랫폼**

팀 관리

제품 설정

카카오 로그인

동의항목

간편가입

iOS

iOS 플랫폼 등록

Web

삭제

수정

사이트 도메인	http://localhost:8000
---------	-----------------------

• 카카오 로그인 사용 시 Redirect URI를 등록해야 합니다. [등록하러 가기](#)

# 카카오 API를 이용한 소셜 로그인

django-allauth

활성화 설정을 ON으로 해주고 Redirect URI 를 설정해줍니다.

### 활성화 설정

상태 ON

카카오 로그인 API를 활용하면 사용자들이 번거로운 회원 가입 절차 대신, 카카오톡으로 서비스를 시작할 수 있습니다.  
상태가 OFF일 때도 카카오 로그인 설정 항목을 변경하고 서버에 저장할 수 있습니다.  
상태가 ON일 때만 실제 서비스에서 카카오 로그인 화면이 연결됩니다.

### OpenID Connect 활성화 설정

상태 OFF

카카오 로그인의 확장 기능인 OpenID Connect를 활성화합니다.  
이 설정을 활성화하면 카카오 로그인 시 사용자 인증 정보가 담긴 ID 토큰을 액세스 토큰과 함께 발급받을 수 있습니다.

### Redirect URI

삭제

수정

Redirect URI
<div>http://127.0.0.1:8000/accounts/kakao/login/callback/ http://localhost:8000/accounts/kakao/login/callback/</div>

- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
- REST API로 개발하는 경우 필수로 설정해야 합니다.

# 카카오 API를 이용한 소셜 로그인

django-allauth

- 일반
- 비즈니스
- 앱 키
- 플랫폼
- 팀 관리
- 제품 설정
- 카카오 로그인
- 동의항목**
- 간편가입
- 카카오톡 채널
- 개인정보 국외이전
- 연결 끊기
- 사용자 프로퍼티
- 보안
- 고급
- 카카오링크

카카오 로그인 ON

[동의 화면 미리보기](#)

## 동의항목

카카오 로그인으로 서비스를 시작할 때 동의 받는 항목을 설정합니다. 미리 보기를 통해 사용자에게 보여질 화면을 확인할 수 있습니다. 사업자 정보를 등록하여 비즈 앱으로 전환하고 비즈니스 채널을 연결하면 권한이 없는 동의 항목에 대한 검수 신청을 할 수 있습니다.

[비즈니스 설정 바로가기](#)

## 개인정보 보호

항목 이름	ID	상태
프로필 정보(닉네임/프로필 사진)	profile	● 필수 동의 [수집] <a href="#">설정</a>
카카오톡 채널 추가 상태 및 내역	plusfriends	○ 권한 없음
카카오계정(이메일)	account_email	● 사용 안함 <a href="#">설정</a>

# 카카오 API를 이용한 소셜 로그인

django-allauth

보안코드의 활성화 상태를 사용함으로 바꿔주세요

일반

비즈니스

앱 키

플랫폼

팀 관리

제품 설정

카카오 로그인

동의항목

간편가입

카카오톡 채널

개인정보 국외이전

연결 끊기

사용자 프로퍼티

**보안**

고급

카카오링크

카카오 로그인 ON

Client Secret

삭제

토큰 발급 시, 보안을 강화하기 위해 Client Secret을 사용할 수 있습니다. (REST API인 경우에 해당)

코드		
활성화 상태	사용함	설정

# 카카오 API를 이용한 소셜 로그인

django-allauth

어드민 페이지에 등록합니다.

Provider:	Kakao ▾
Name:	<input type="text" value="kakao"/>
Client id:	<div>REST API KEY</div> <small>App ID, or consumer key</small>
Secret key:	<div>CLIENT SECRET CODE</div> <small>API secret, client secret, or consumer secret</small>
Key:	<input type="text"/> <small>Key</small>
Sites:	<div><div>Available sites</div><div><input type="text" value="Filter"/></div></div> <div><div>Chosen sites</div><div><input type="text" value="http://127.0.0.1:8000"/></div></div>

# 카카오 API를 이용한 소셜 로그인

django-allauth

소셜로그인을 적용하고 나면 인증 백엔드가 여러개 구성됩니다.  
따라서 로그인할 때 user정보를 어느 백엔드에서 가져와야 할지를 설명해주어야 합니다.

```
def login(request):
    if request.method == "POST":
        username = request.POST["username"]
        password = request.POST["password"]
        user = auth.authenticate(request, username=username, password=password)
        if user is not None:
            auth.login(
                request, user, backend="django.contrib.auth.backends.ModelBackend"
            )
            # return redirect("home")
            return redirect(request.GET.get("next", "/"))
        error = "아이디 또는 비밀번호가 틀립니다"
        return render(request, "registration/login.html", {"error": error})

    return render(request, "registration/login.html")
```



# 카카오 API를 이용한 소셜 로그인

django-allauth

templates/base.html

```
<html lang="en">
<head>
  {% load static %} {% load socialaccount %}{% providers_media_js %}
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <!-- fontawesome-link code -->
  <script
    src="https://kit.fontawesome.com/62169c8271.js">
```

```
<li class="navbar__menu__item">
  <a href="{% provider_login_url 'kakao' %}">카카오 로그인</a>
</li>
{% endif %}
```

# 카카오 API를 이용한 소셜 로그인

django-allauth

소셜 로그인 버튼을 누르면?

Messages:

- Successfully signed in as user.

Menu:

- [Sign In](#)
- [Sign Up](#)

## Sign In Via Kakao

You are about to sign in using a third party account from Kakao.

Continue

의도한 카카오 로그인 페이지가 바로 뜨지 않는다 ㅜㅜ

# 카카오 API를 이용한 소셜 로그인

django-allauth

나름 솔루션을 찾아보았습니다.

```
</li>
<!-- <li class="navbar__menu_item">
  <a href="{% provider_login_url 'kakao' %}">카카오 로그인</a>
</li> -->
<form
  class="login"
  method="POST"
  action="{% provider_login_url 'kakao' %}"
>
  {% csrf_token %} {{ form.as_p }} {% if redirect_field_value %}
  <input
    type="hidden"
    name="{{ redirect_field_name }}"
    value="{{ redirect_field_value }}"
  />
  {% endif %}
  <button class="primaryAction" type="submit">카카오 로그인</button>
</form>
```

참고자료 : <https://jellyho.com/blog/27/>

# 카카오 API를 이용한 소셜 로그인

django-allauth

Q. 한번 로그인한 이후에,

로그아웃 하고 다시 로그인 시도하면 카카오 로그인 UI가 안뜨고 바로 자동로그인 돼요.

A. <https://accounts.kakao.com/>

여기 들어가서 로그아웃 해줍니다.

# 과제

수고 많으셨습니다! ☺



저번 세션 과제 파일에 이어서 진행해주시면 됩니다

오늘 배웠던 것들을 복습합시다!

필수 !

1. navbar에 회원가입/로그인/로그아웃 버튼 만들고 기능 구현하기
2. 로그인 여부에 따라서 페이지 접근 권한 막기 & navbar 다르게 보이기
3. 작성자만 글 수정, 글과 댓글 삭제할 수 있도록

Option !

- 카카오 API 말고 다른 API 적용해보기 ex) 구글, 네이버 등등
- 추가 공부해보기

Keyword : 인증방식, OAuth