

Особливості створення web додатків в Django

1. Основні особливості Django
2. Створення нового проекту Django
3. Запуск веб додатку
4. Міграція баз даних (БД)
5. Створення Django applications
6. Реєстрація застосунку в **settings.py**
7. Створення папки templates та HTML сторінки в ній
8. Створення view функції в **views.py**
9. Розподілення запитів клієнта в файлі **urls.py**
10. Таблиця культових файлів Django

Django - це відкритий фреймворк для розробки веб-додатків, написаний на мові програмування Python. Він надає розробникам набір інструментів та бібліотек для швидкого і зручного створення веб-додатків, зокрема, веб-сайтів і веб-додатків з базами даних.

Основні особливості Django включають:

- Система маршрутизації: Django надає шляхи до веб-сторінок та визначає, який код виконувати при доступі до конкретного URL.
- Об'єктно-реляційна модель: Django надає вбудовану ORM (Object-Relational Mapping), яка дозволяє розробникам працювати з базами даних за допомогою Python-коду, не звертаючись безпосередньо до SQL.
- Адміністративна панель: Django має вбудовану адміністративну панель, яка допомагає легко керувати даними вашого додатка, включаючи створення, редагування та видалення записів у базі даних.
- Шаблони: Django використовує шаблони для відображення даних на сторінках веб-сайту. Це допомагає розділити логіку додатку від його відображення.
- Вбудовані засоби безпеки: Django має вбудовані заходи безпеки, які допомагають захистити додатки від різних атак, таких як впровадження SQL, міжсайтовий скриптинг тощо.
- Розширюваність: Django легко розширюється за допомогою сторонніх розширень (або "пакетів"), які спрощують розробку специфічних для вашого проекту функцій.

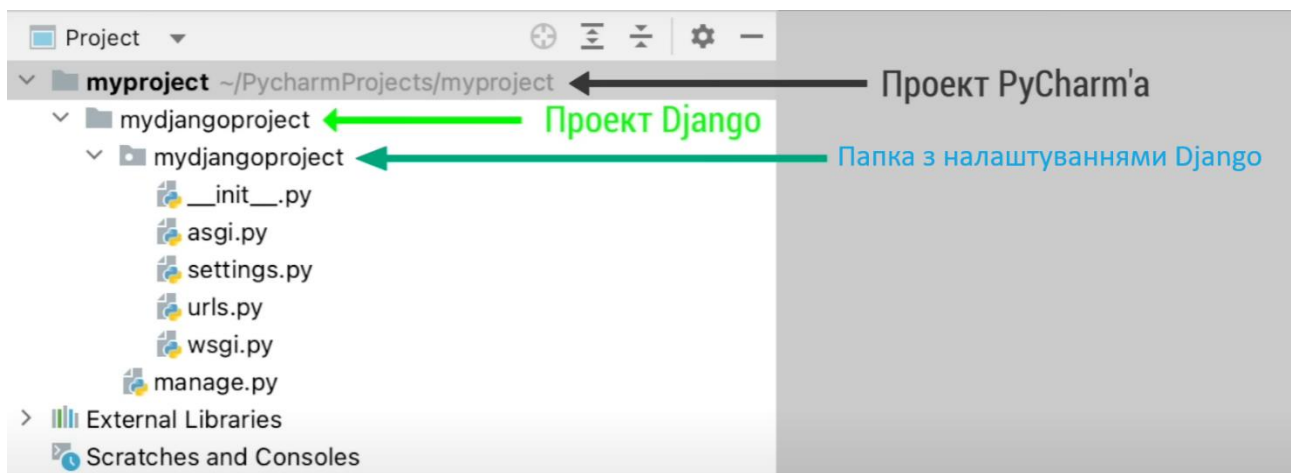
Django дуже популярний серед розробників, оскільки він пропонує швидкий старт для створення веб-додатків та дозволяє дотримуватися кращих практик розробки. Він використовується для створення різноманітних веб-додатків, від простих блогів до складних веб-порталів і електронних комерційних систем.

Створення нового проекту Django

1. Створюємо віртуальне середовище
2. Створюємо проект
3. `pip install Django`

Інший спосіб: PyCharm → Main Menu → Settings → Project: project_name → Python Interpreter → + → Ввести ім'я пакету → Install Package

4. `django-admin startproject mydjangoпроект`



5. Війти в папку проекту: `cd mydjangoпроект` (це коренева папка проекту)

6. Помітимо папку (в PyCharm) проекту як Sources Root:

Папка `mydjangoпроект` → контекстне меню → Mark Directory As → Sources Root

Запуск веб додатку

```
./manage.py runserver
```

django

View [release notes](#) for Django



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.

A screenshot of the PyCharm IDE's Project tool window. The project is named 'myproject' and is located at '~/PycharmProjects/myproject'. The 'mydjangoproject' folder is expanded, showing subfolders and files. The 'db.sqlite3' file is highlighted with a green box. To the right of the file, the text 'База даних' (Database) is written in green. Other items visible include 'manage.py', 'External Libraries', and 'Scratches and Consoles'.

Project

myproject ~/PycharmProjects/myproject

mydjangoproject

mydjangoproject

db.sqlite3 База даних

manage.py

External Libraries

Scratches and Consoles

A screenshot of the PyCharm console output. The text is as follows:

```
System check identified no issues (0 silenced).  
You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,  
Run 'python manage.py migrate' to apply them.  
August 24, 2022 - 09:51:55  
Django version 4.1, using settings 'mydjangoproject.settings'  
Starting development server at http://127.0.0.1:8000/
```

The first two lines of the output are highlighted with a green box.

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin,
Run 'python manage.py migrate' to apply them.

August 24, 2022 - 09:51:55

Django version 4.1, using settings 'mydjangoproject.settings'

Starting development server at <http://127.0.0.1:8000/>

Міграція баз даних (БД)

Потрібна коли ми щось змінюємо в БД.

В самий перший раз Django хоче створити свої службові таблиці, тому просить нас створити міграцію.

Міграція баз даних (БД) - це процес переміщення даних та структури бази даних з одного середовища або системи у інше. Цей процес може включати в себе різні аспекти, такі як:

Переміщення даних: Це означає копіювання і перенесення фактичних даних з однієї БД до іншої. Це включає в себе таблиці, записи, зв'язки між даними тощо.

Переміщення схеми: Це означає створення таблиць, індексів, виділень інших об'єктів бази даних в новій системі так, щоб вони відповідали структурі і схемі попередньої БД.

Актуалізація даних: Іноді, під час міграції, доводиться проводити трансформації або обробку даних для забезпечення відповідності новій схемі чи вимогам додатка.

Тестування та верифікація: Після міграції важливо провести тестування для переконання, що дані були правильно перенесені і що нова система працює коректно.

Міграція баз даних може бути необхідною з різних причин, таких як оновлення програмного забезпечення, перенесення даних в хмару, консолідація баз даних, а також для забезпечення безпеки і дотримання вимог щодо зберігання даних. Важливо правильно спланувати та виконати міграцію, щоб уникнути втрати даних та збоїв в роботі системи.

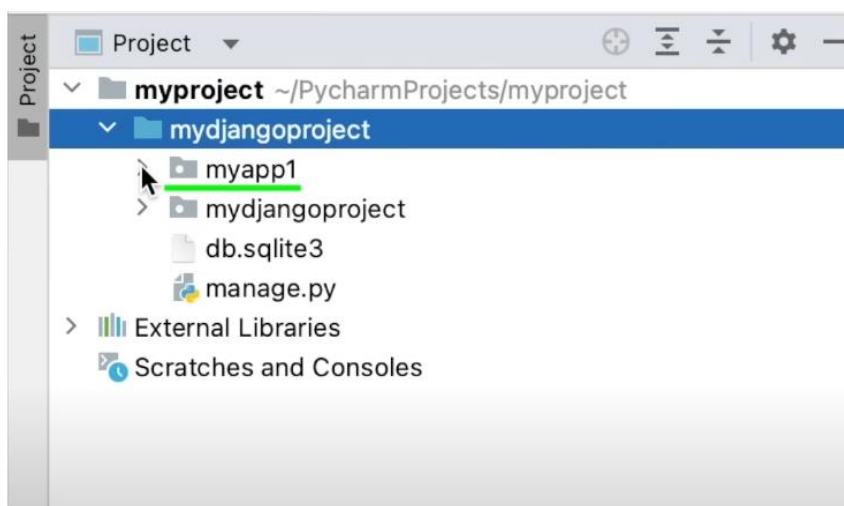
```
./manage.py migrate
```

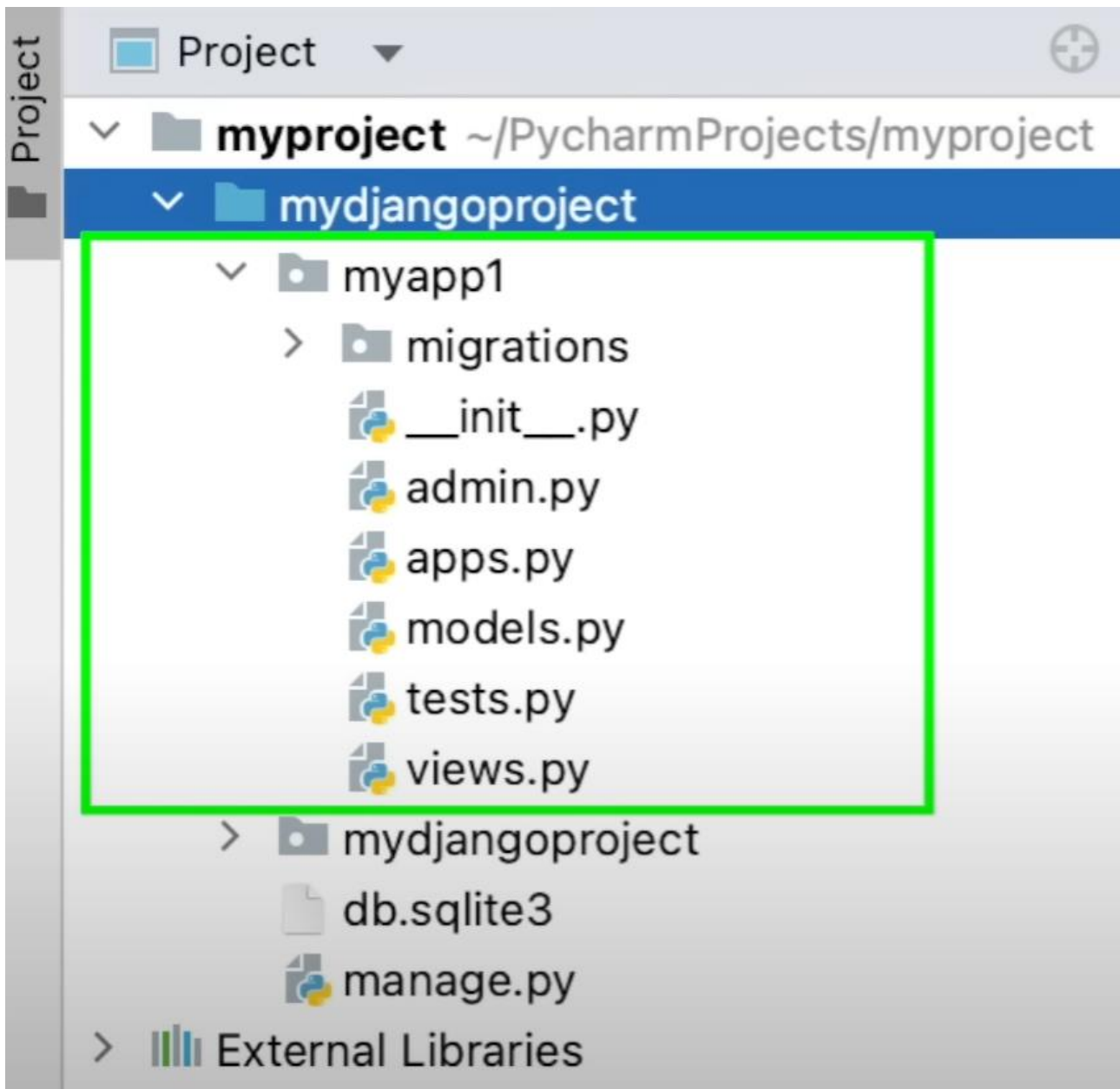
Створення Django applications



Кожний застосунок відповідає за свій функціонал.

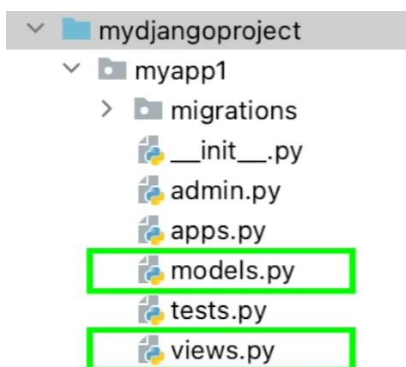
```
./manage.py startapp myapp1
```





```
./manage.py startapp myapp2  
./manage.py startapp myapp3
```

Основний код буде в цих файлах:



Please take a few minutes to complete the **Django Developers Survey 2023**.
Your feedback will help guide future efforts.

Django documentation

Everything you need to know about Django.

First steps

Are you new to Django or to programming? This is the place to start!

- **From scratch:** [Overview](#) | [Installation](#)
- **Tutorial:** [Part 1: Requests and responses](#) | [Part 2: Models and the admin site](#) | [Part 3: Views and templates](#) | [Part 4: Forms and generic views](#) | [Part 5: Testing](#) | [Part 6: Static files](#) | [Part 7: Customizing the admin site](#) | [Part 8: Adding third-party packages](#)
- **Advanced Tutorials:** [How to write reusable apps](#) | [Writing your first patch for Django](#)

Getting help

Having trouble? We'd like to help!

Support Django!



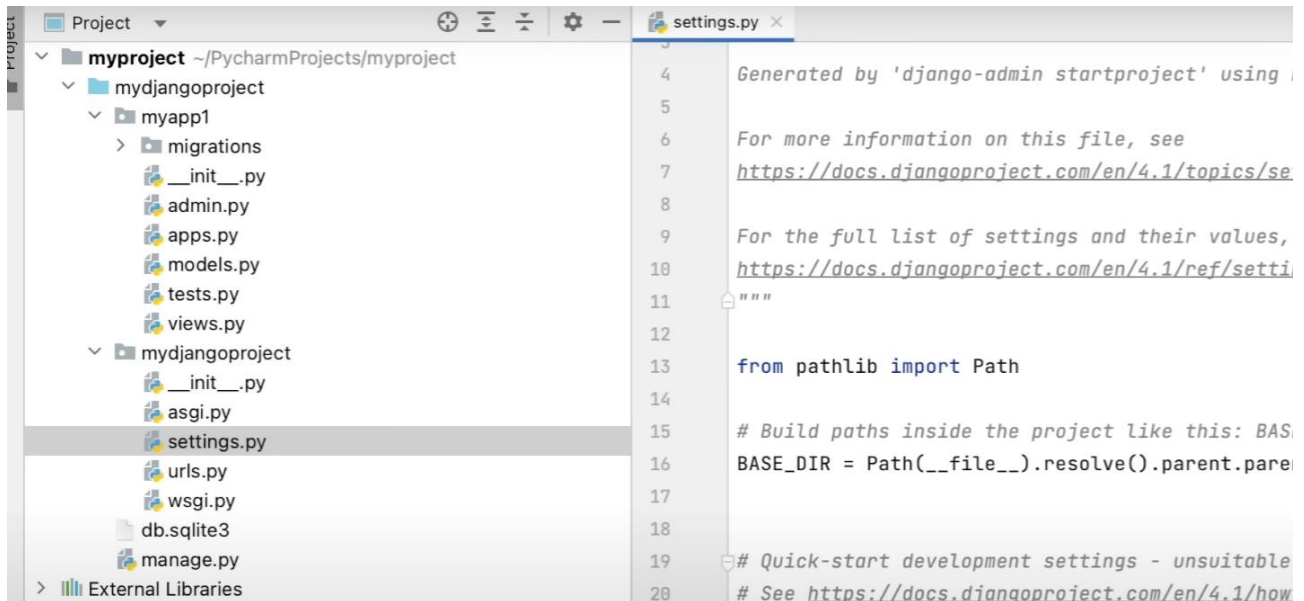
LAAC Technology donated to the Software Foundation to support development. Donate today!

Browse

- Prev: [Django documentation contents](#)
- Next: [Getting started](#)
- [Table of contents](#)
- [General Index](#)
- [Python Module Index](#)

Реєстрація застосунку в settings.py

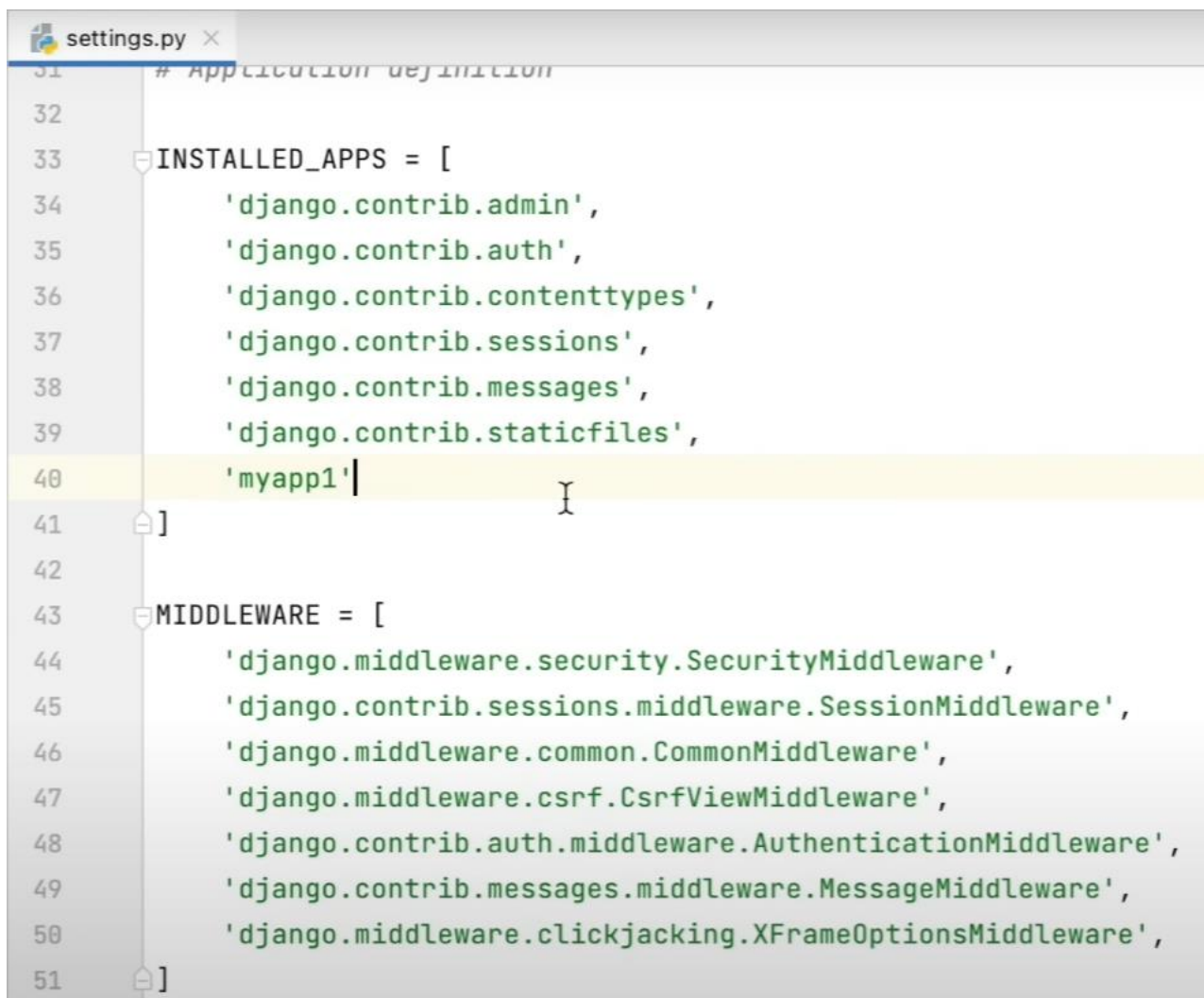
Settings.py – головний файл налаштувань проекту. Це звичайний Python файл.



The screenshot shows the PyCharm IDE interface. On the left, a project tree displays the structure: myproject (~/PycharmProjects/myproject) containing mydjango project, which includes a myapp1 sub-project with migrations, __init__.py, admin.py, apps.py, models.py, tests.py, and views.py. The main settings.py file is highlighted. The right pane shows the content of settings.py, which includes a docstring, imports, and configuration comments.

```
4 Generated by 'django-admin startproject' using
5
6 For more information on this file, see
7 https://docs.djangoproject.com/en/4.1/topics/se
8
9 For the full list of settings and their values,
10 https://docs.djangoproject.com/en/4.1/ref/setti
11 """
12
13 from pathlib import Path
14
15 # Build paths inside the project like this: BASE
16 BASE_DIR = Path(__file__).resolve().parent.pare
17
18
19 # Quick-start development settings - unsuitable
20 # See https://docs.djangoproject.com/en/4.1/how
```

Щоб Django міг працювати з застосунком, його необхідно зареєструвати в settings.py



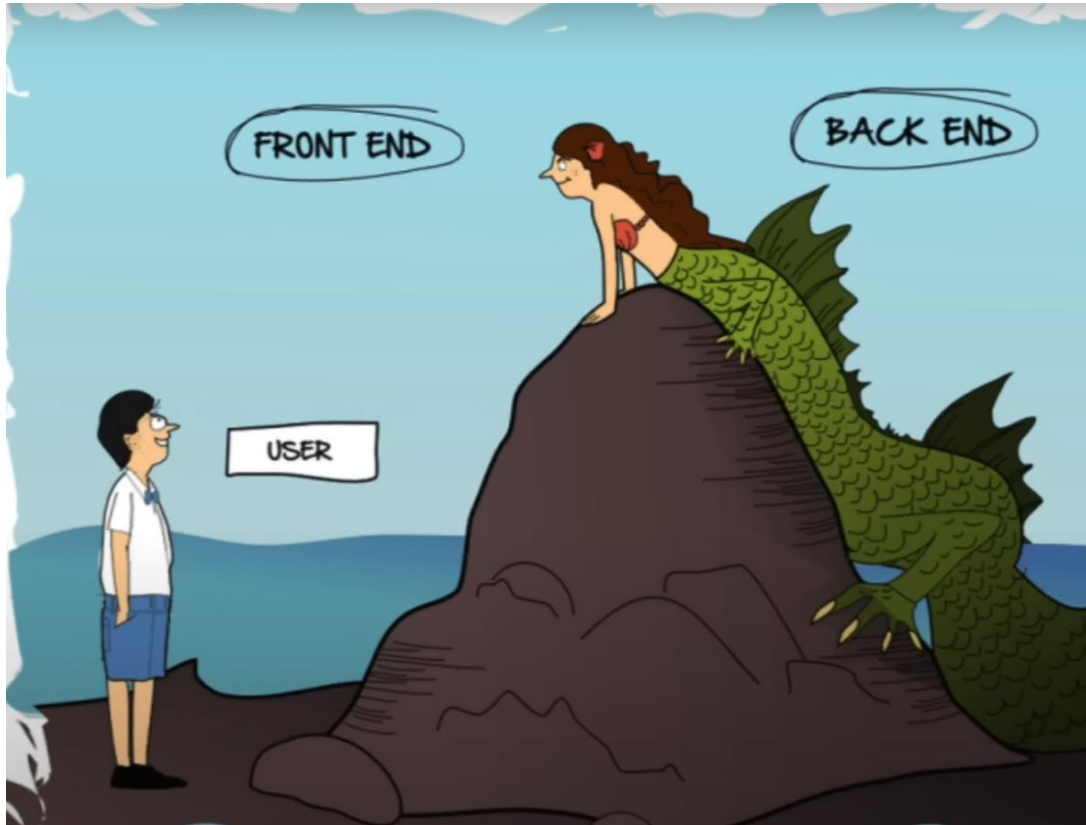
The screenshot shows a close-up of the settings.py file in the PyCharm IDE. The code defines the INSTALLED_APPS list, which includes several Django core applications and the custom application 'myapp1'. The 'myapp1' entry is highlighted in yellow, and the cursor is positioned at the end of the list.

```
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'myapp1'
41 ]
42
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
48     'django.contrib.auth.middleware.AuthenticationMiddleware',
49     'django.contrib.messages.middleware.MessageMiddleware',
50     'django.middleware.clickjacking.XFrameOptionsMiddleware',
51 ]
```


Створення папки templates та HTML сторінки в ній

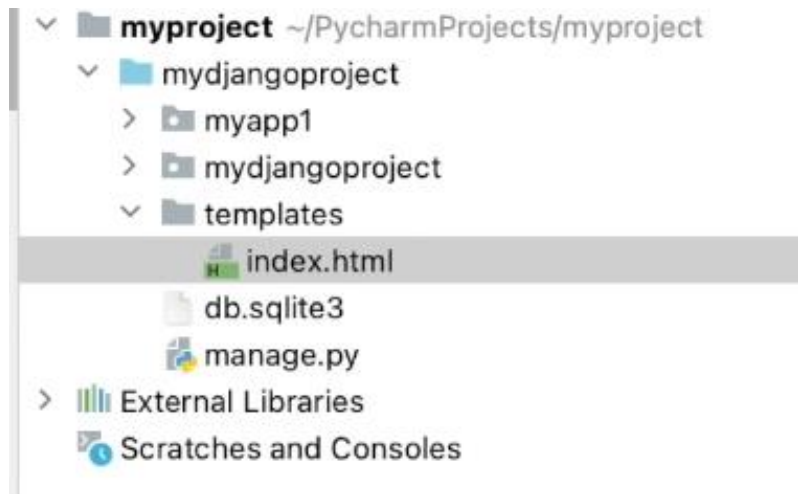
Кожен сайт можна поділити на дві частини:

- Front End
- Back End



Всі Django шаблони (Front End) прийнято зберігати в папці **templates**

Її потрібно створити окремо і в ней веб сторінку *index.html*

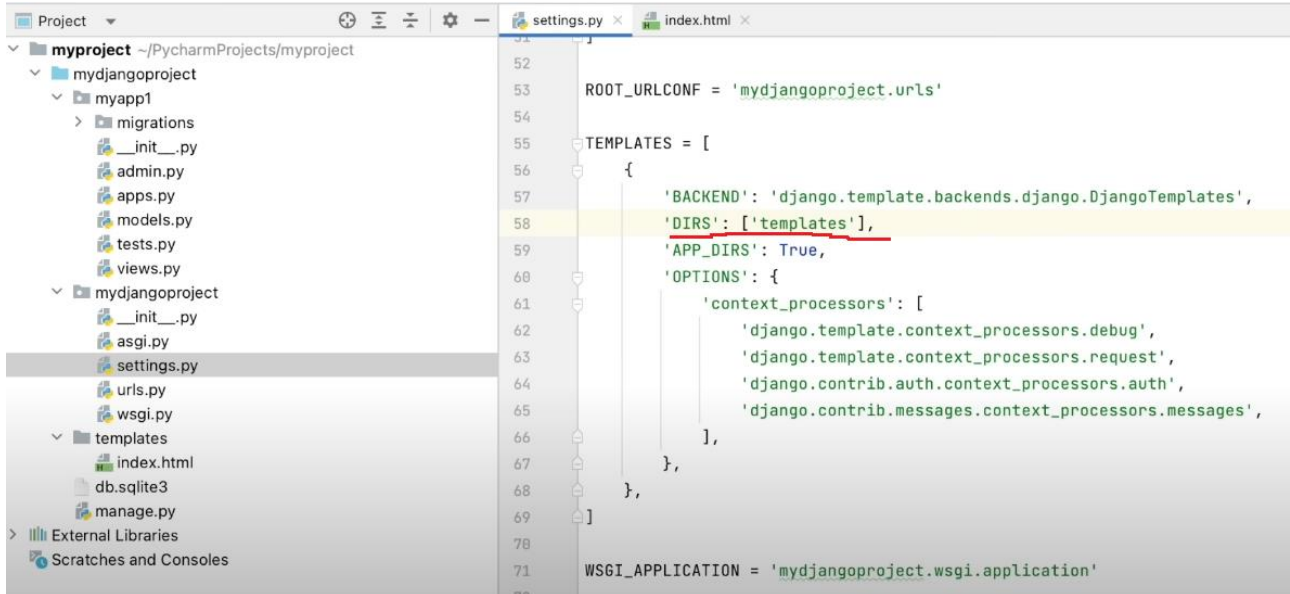


```
settings.py x index.html x
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6 </head>
7 <body>
8     <h1>HELLO, WORLD!!!!!!</h1>
9 </body>
10 </html>
```

Зареєструємо папку **templates** в settings.py

Просто вставити папку в проект не вийде!!!

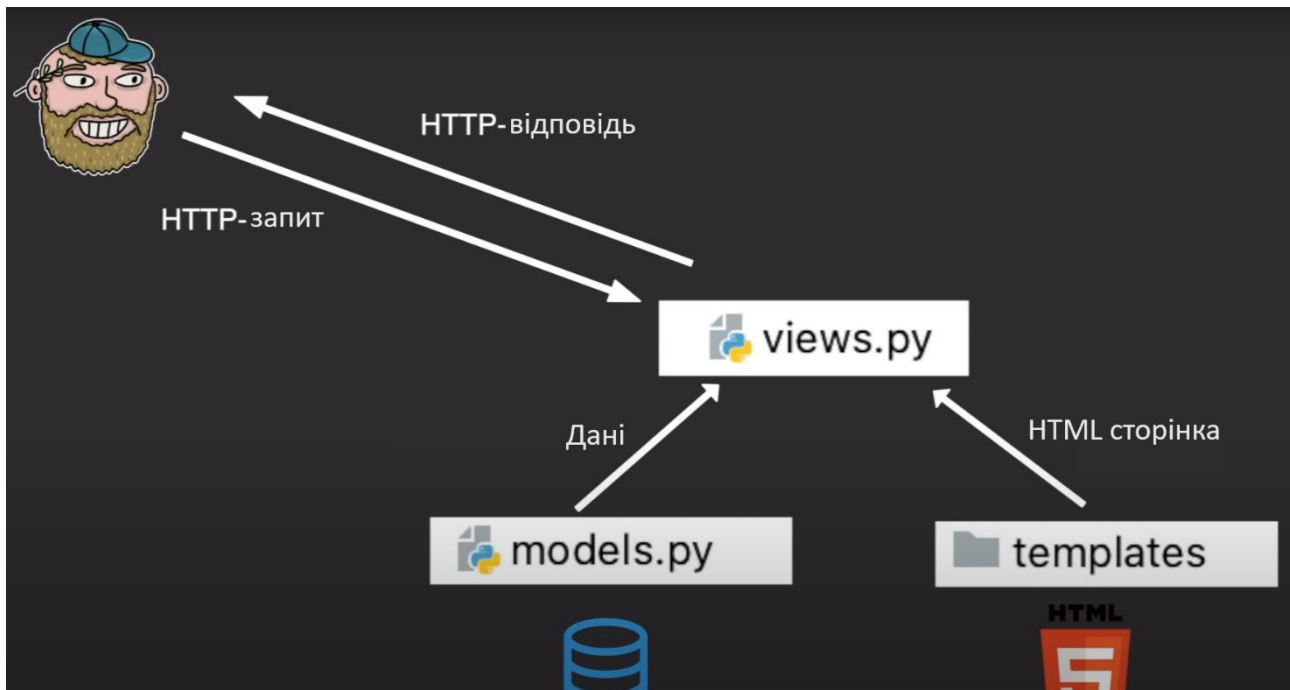
Для того що її бачив Django і знав, що в ній знаходиться (в нашому випадку це шаблони), її необхідно зареєструвати в файлі *settings.py* у відповідному розділі.



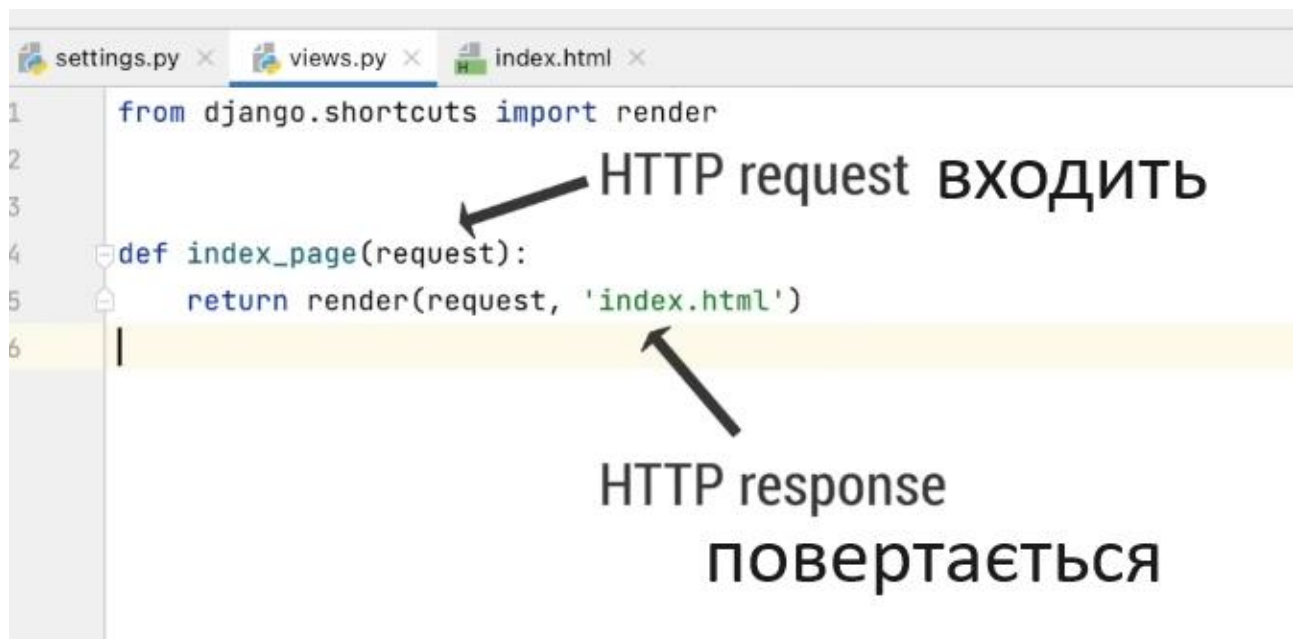
```
52
53 ROOT_URLCONF = 'mydjangoproject.urls'
54
55 TEMPLATES = [
56     {
57         'BACKEND': 'django.template.backends.django.DjangoTemplates',
58         'DIRS': ['templates'],
59         'APP_DIRS': True,
60         'OPTIONS': {
61             'context_processors': [
62                 'django.template.context_processors.debug',
63                 'django.template.context_processors.request',
64                 'django.contrib.auth.context_processors.auth',
65                 'django.contrib.messages.context_processors.messages',
66             ],
67         },
68     },
69 ]
70
71 WSGI_APPLICATION = 'mydjangoproject.wsgi.application'
```

Створення view функції в views.py

view-функція обробляє запит користувача і повертає йому результат.



Приклад простішої **views функції**, яка отримує якийсь запит від користувача і повертає html сторінку



The screenshot shows a code editor with three tabs: settings.py, views.py, and index.html. The views.py file contains the following code:

```
1 from django.shortcuts import render
2
3
4 def index_page(request):
5     return render(request, 'index.html')
6
```

Annotations in Ukrainian are present:

- An arrow points to the `request` parameter in the function signature, with the text "HTTP request ВХОДИТЬ" (HTTP request ENTERS).
- Another arrow points to the `'index.html'` string in the `render` function call, with the text "HTTP response повертається" (HTTP response RETURNS).

views функції, які повертають сторінки [index.html](#) і [about.html](#)



The screenshot shows a code editor with three tabs: settings.py, views.py, and index.html. The views.py file contains the following code:

```
1 from django.shortcuts import render
2
3
4 def index_page(request):
5     return render(request, 'index.html')
6
7
8 def about_page(request):
9     return render(request, 'about.html')
```

The `def about_page(request):` block is highlighted with a green box.

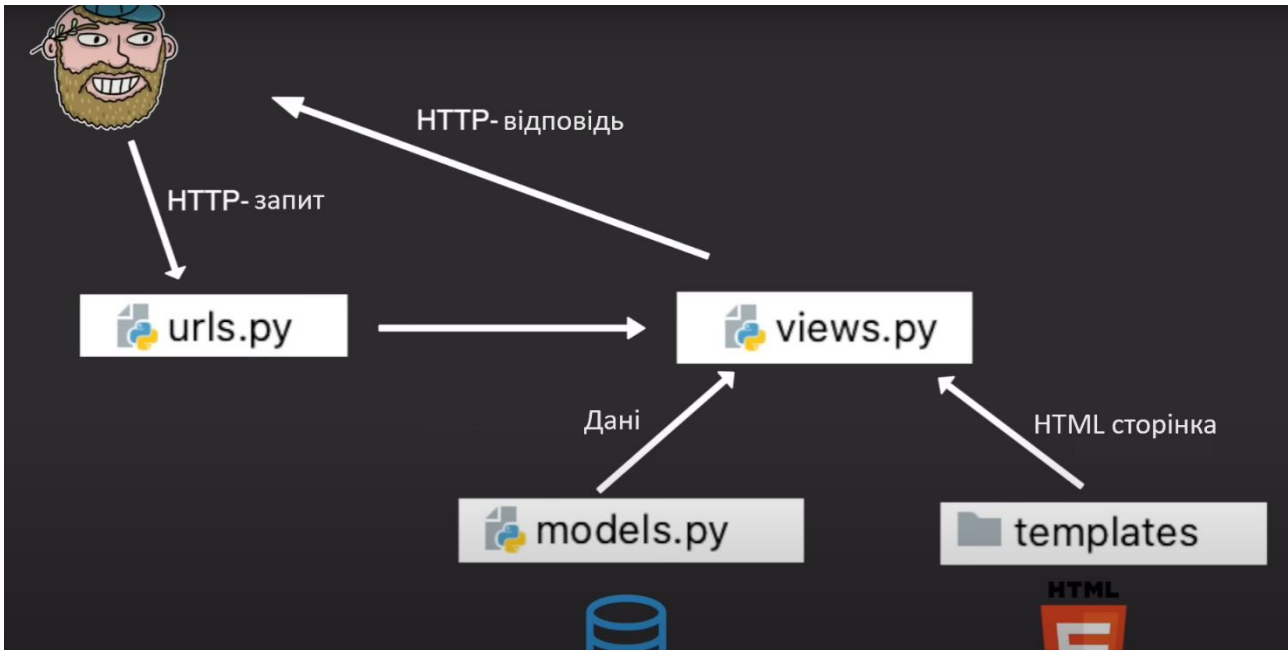
В **views функції** можна формувати динамічні web сторінки, в залежності від змісту БД.

```
def about_page(request):  
    Беремо дані із БД  
    Щось рахуємо,  
    виконуємо якісь дії  
    return render(request, 'about.html')
```

В **urls.py** визначається до якої конкретно views-функції іде звернення (в проекті зазвичай views-функції багато)

Коли приходить запит від користувача, Django спочатку в **urls.py** шукає яка view-функція повинна обробити цей запит. Потім відправляє запит саме цій функції.

Якщо Django не знайшов відповідної view-функції, то повертає **404**.



Приклад. При введенні адреси сайту відкрити сторінку [index_page](#):

```
from django.contrib import admin
from django.urls import path
from myapp1.views import index_page

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', index_page)
]
```

Приклад 2. При додаванні до адреси /about
(http://адреса_сайту/about) відкрити сторінку [about_page](#):

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('', index_page),  
    path('about/', about_page),  
]
```

Таблиця культових файлів Django

Код Django розділено на різні файли. І у кожного файлу своя відповідальність.

manage.py	Запускає сервер, організовує всі процеси
settings.py	Зберігає всі паролі, налаштування, знає де що лежить
urls.py	Направляє http запити до потрібних view-функцій
models.py	Взаємодіє з БД
views.py	З'єднує дані і шаблони. Приймає http запити і відправляє http відповіді
templates	фронтенд

Результат роботи проекту



HELLO, WORLD!!!!