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 скриптінг 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

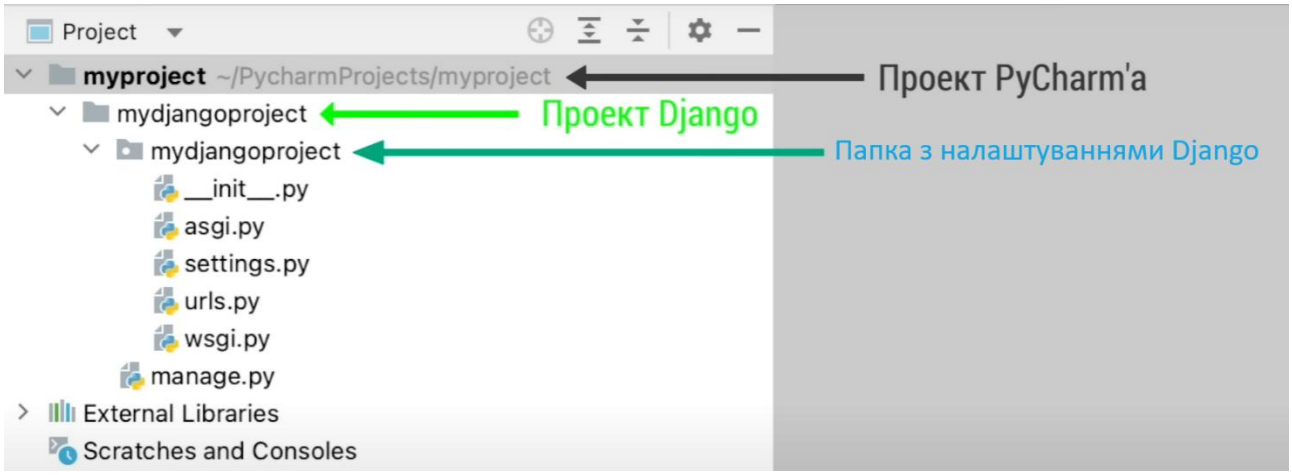
1. Створюємо віртуальне середовище

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 mydjangoproject



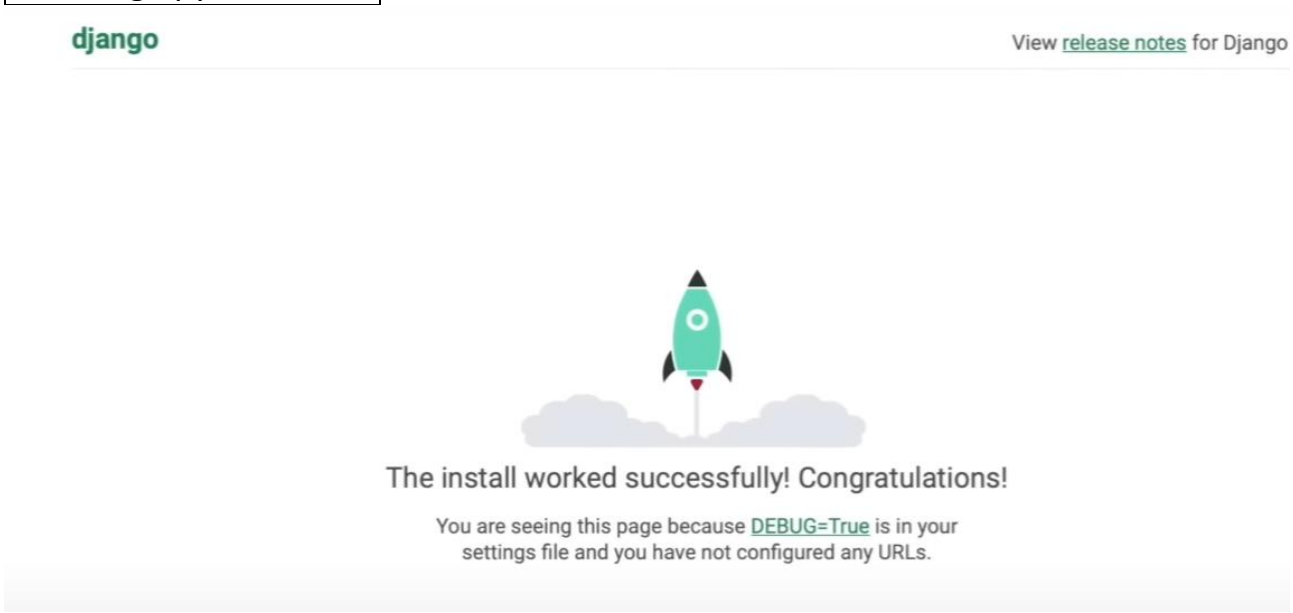
5. Війти в папку проекту: cd mydjangoproject (це коренева папка проекту)

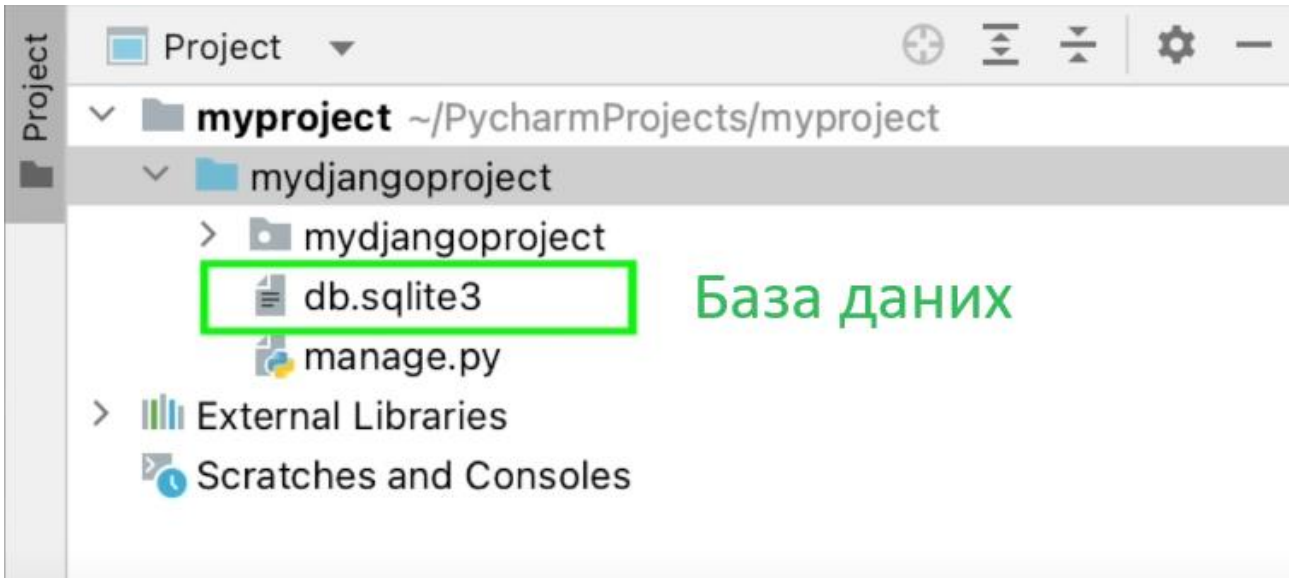
6. Помітимо папку (в PyCharm) проекту як Sources Root:

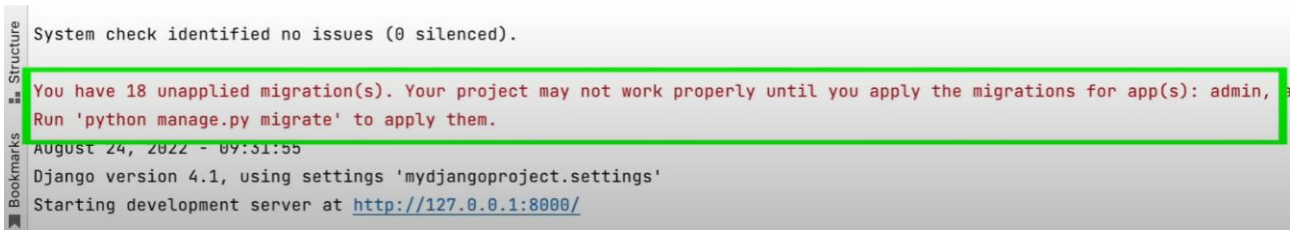
Папка mydjangoproject → контекстне меню → Mark Directory As → Sources Root

Запуск веб додатку

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







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 переконанняся, що дані були правильно перенесені і що нова система працює коректно.

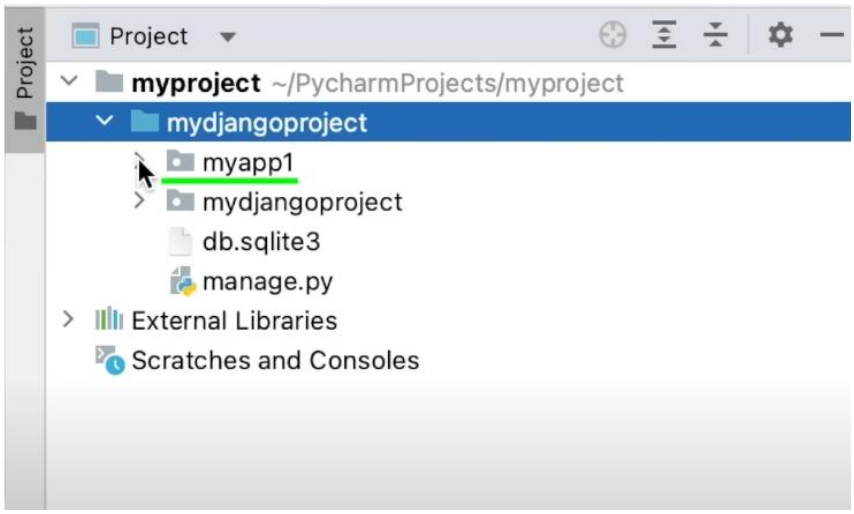
Міграція баз даних може бути необхідною з різних причин, таких як оновлення програмного забезпечення, перенесення даних в хмару, консолідація баз даних, а також для забезпечення безпеки і дотримання вимог щодо зберігання даних. Важливо правильно спланувати та виконати міграцію, щоб уникнути втрати даних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

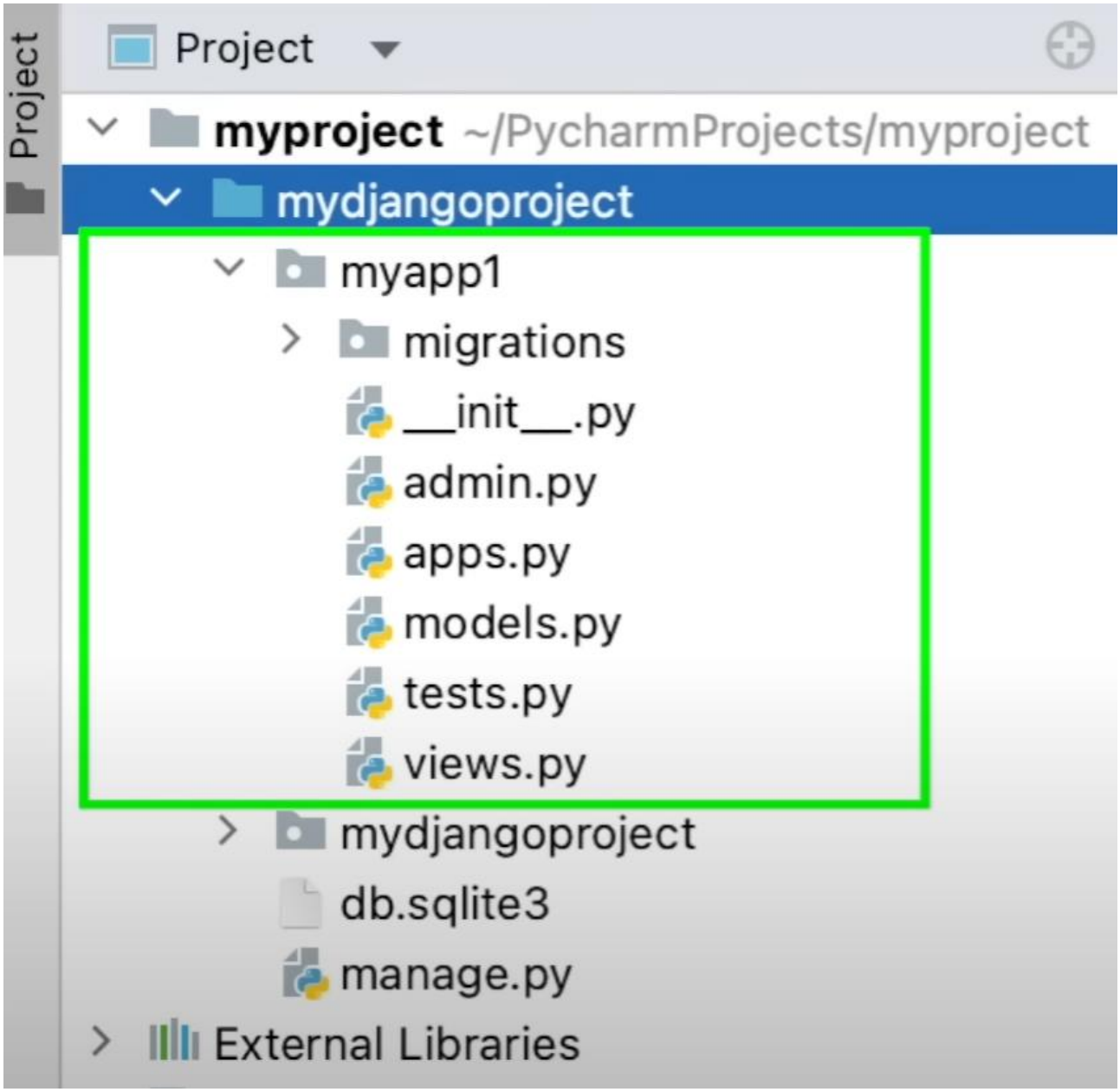
```
./manage.py migrate
```

Створення Django applications

Кожний застосунок відповідає за свій функціонал.

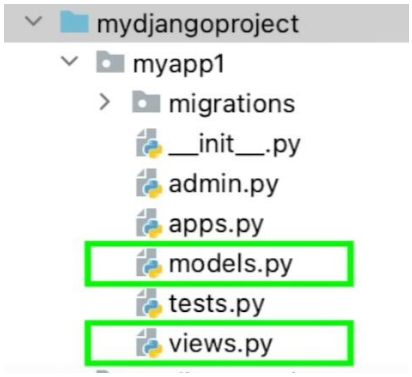
./manage.py startapp myapp1






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

Основний код буде в цих файлах:



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

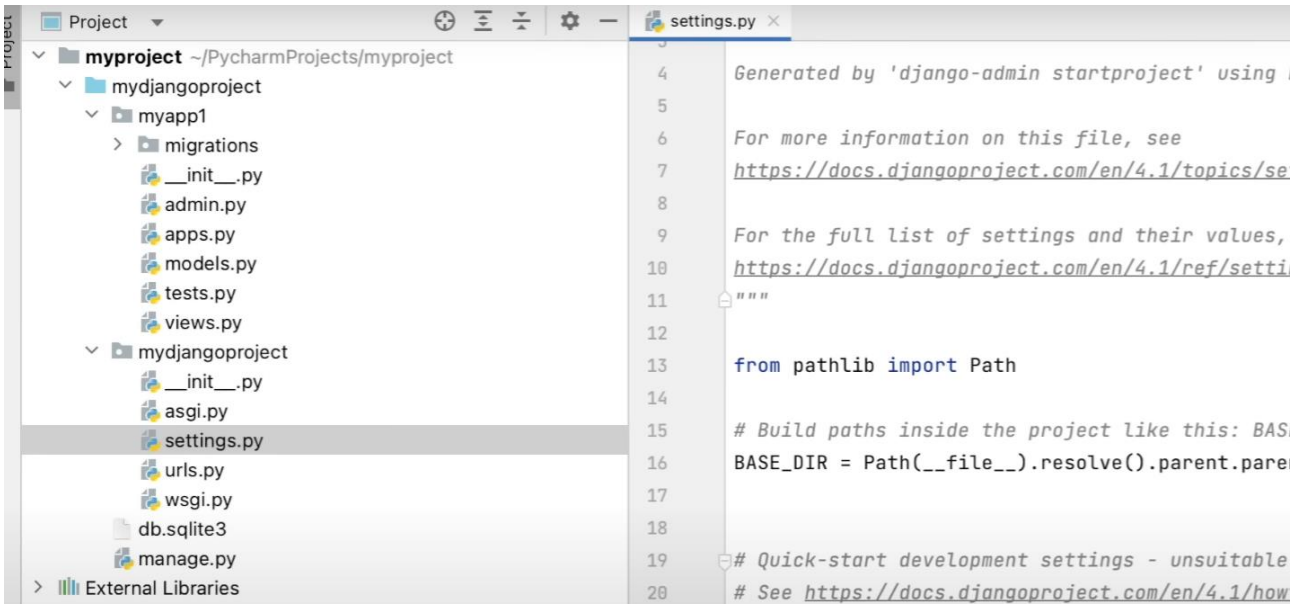
Having trouble? We'd like to help!

Browse

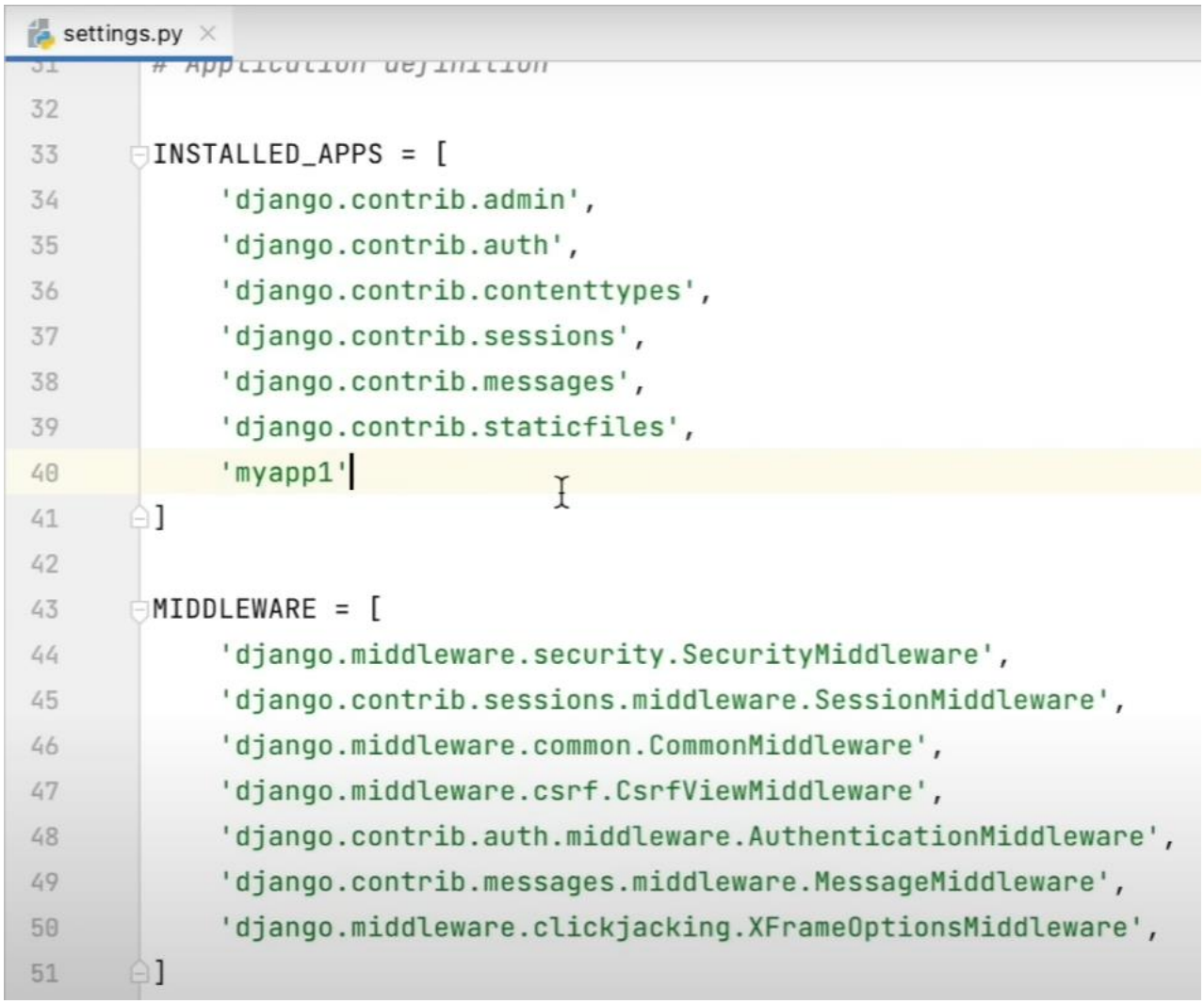
- Prev: Django documentation contents
- Next: Getting started
- Table of contents
- General Index
- Python Module Index

Реєстрація застосунку в settings.py

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



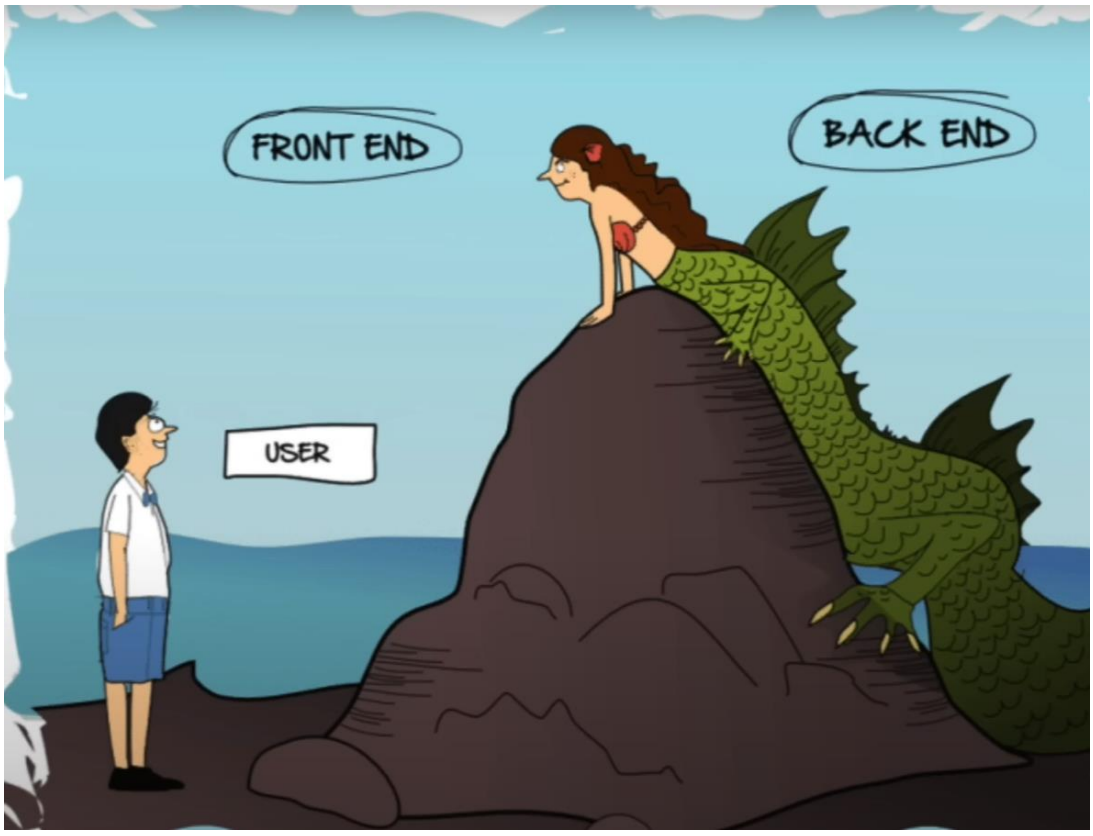
Щоб Django міг працювати з застосунком, його необхідно зареєструвати в settings.py



Створення папки templates та HTML сторінки в ній

Кожен сайт можна поділити на дві частини:

- Front End
- Back End



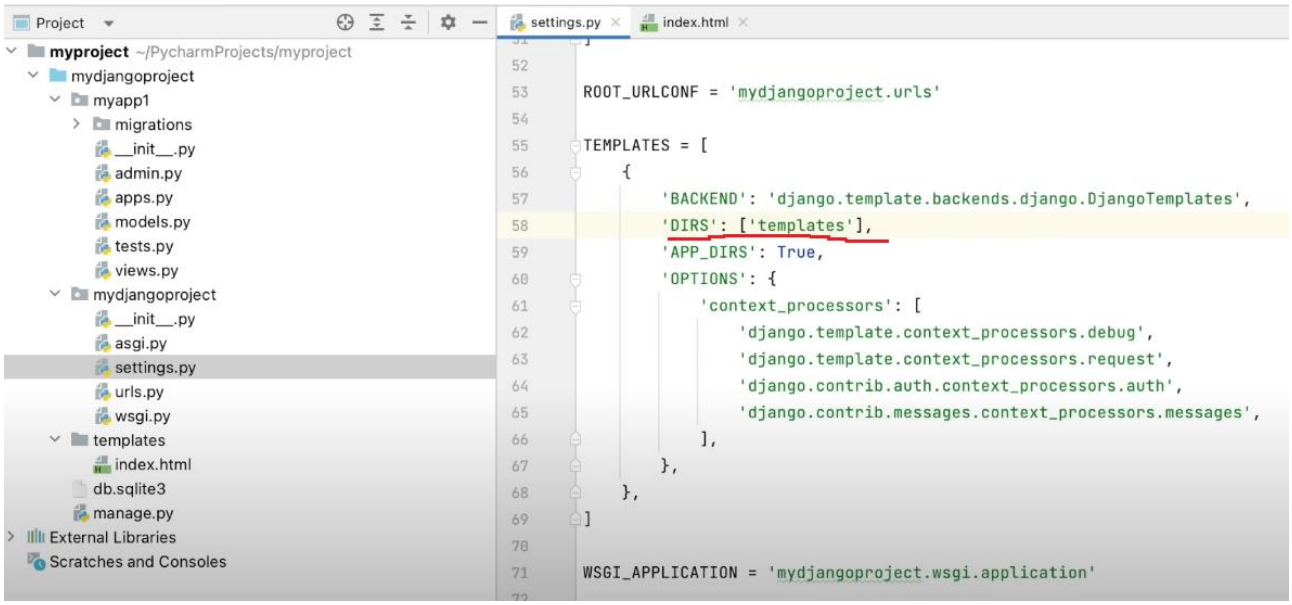
Всі Django шаблони (Front End) прийнято зберігати в папці **templates**

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

Зареєструємо папку **templates** в settings.py

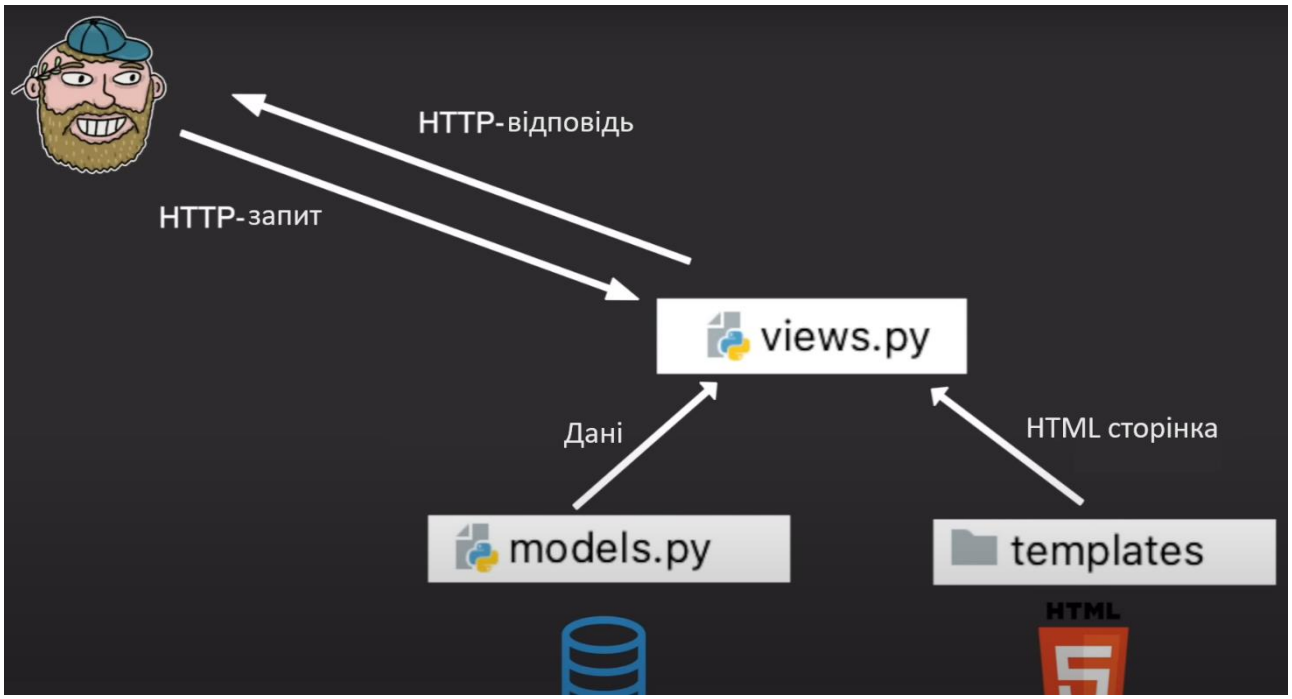
Просто вставити папку в проект не вийде!!!

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.

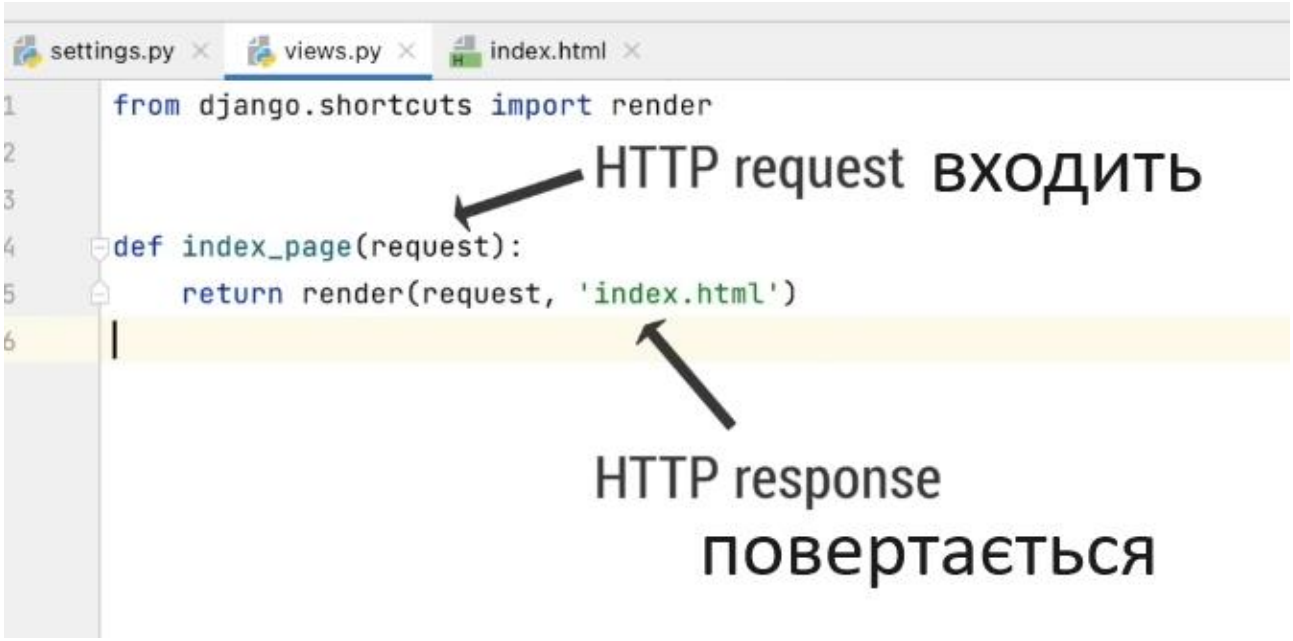


Створення view функції в views.py

view-функція обробляє запит користувача і повертає йому результат.

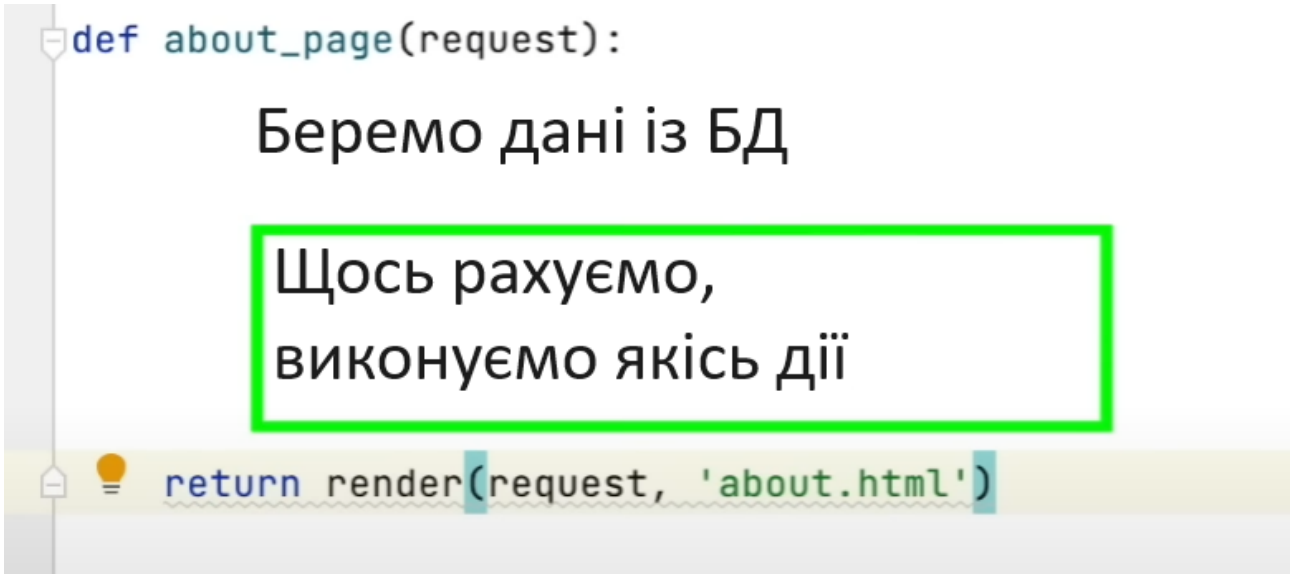


Приклад простішої **views функції**, яка отримає якийсь запит від користувача і повертає html сторінку



views функції, які повертають сторінки index.html і about.html

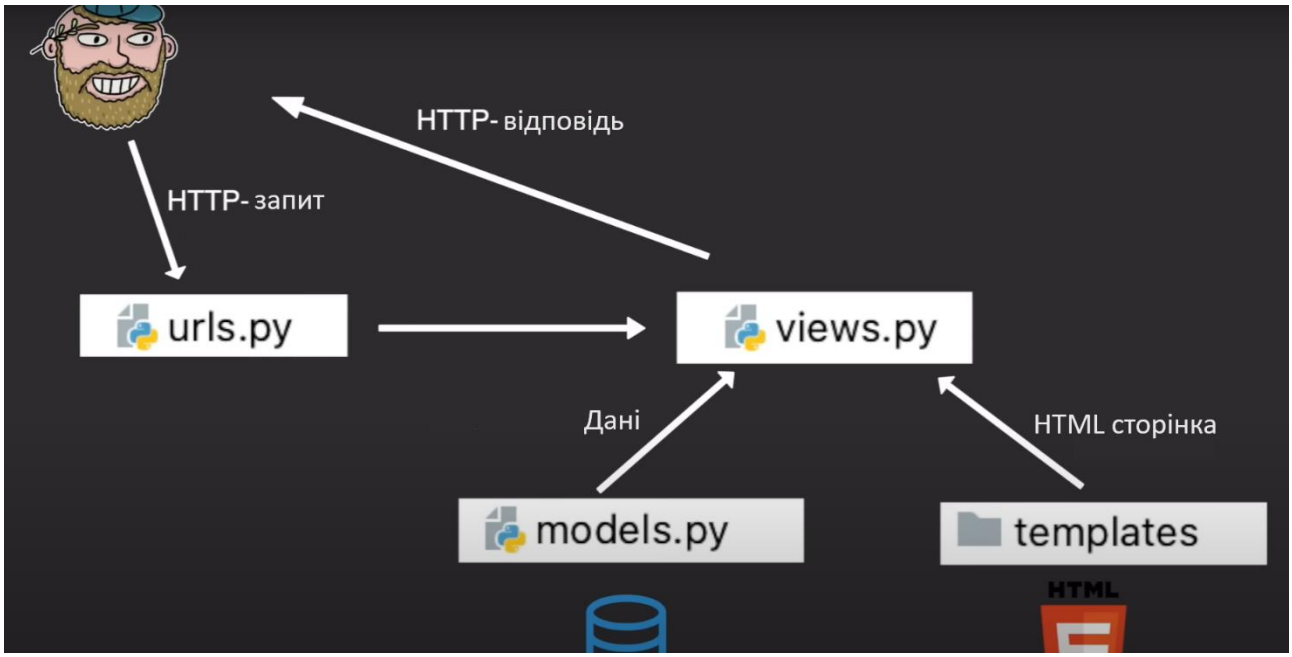
В **views функції** можна формувати динамічні web сторінки, в залежності від змісту БД.



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