Вступ: Що таке Django?

Django — це потужний, безкоштовний фреймворк для розробки веб-додатків, створений на мові програмування Python. Його основна філософія — "батарейки в комплекті" (*batteries included*), що означає, що він надає готовий функціонал для багатьох поширених завдань, дозволяючи розробникам швидко створювати складні додатки, не вигадуючи велосипед.

- **Історія:** Django був створений у 2005 році для розробки веб-сайтів газети **Lawrence Journal-World**. Його назвали на честь відомого гітариста Джанго Райнгардта.
- **Популярність:** Фреймворк використовується на багатьох великих сайтах, таких як **Instagram**, **Pinterest** та **Mozilla**.
- **Безпека:** Велика увага приділяється безпеці, що допомагає уникнути таких поширених проблем, як **SQL-ін'єкції** та **CSRF-атаки**.
- **Відкритість:** Django є проектом з відкритим вихідним кодом, який активно розвивається спільнотою.

Архітектура Django: MVT-патерн

Django реалізує архітектурний патерн **MVT (Model-View-Template)**, який є модифікацією класичного патерну MVC.

Уявіть, що ви шеф-кухар у ресторані.

- **Замовлення (Запит):** Клієнт робить замовлення.
- **Офіціант (Диспетчер URL):** Він приймає замовлення і передає його на кухню.
- **Шеф-кухар (View):** Він отримує замовлення, вирішує, що робити (чи потрібно брати інгредієнти з комори), готує страву.
- **Комора (Model):** Це сховище інгредієнтів (даних). Шеф-кухар може брати звідти інгредієнти або додавати нові.
- **Тарілка та подача (Template):** Це готова страва, подана на тарілці. Вона має гарний вигляд і готова для клієнта (HTML-розмітка).

Елемент	Опис	Відповідальність
URL dispatcher	Визначає, який **View** (вигляд) повинен обробити запит, виходячи з URL-адреси.	Маршрутизація запитів.
View (Вигляд)	Функція або клас, що приймає запит, обробляє його, взаємодіє з **Model** та повертає відповідь.	Обробка запитів та бізнес-логіка.

Model (Модель)	Описує структуру даних додатка та взаємодіє з базою даних. Кожен клас моделі — це таблиця в БД.	Робота з даними.
Template (Шаблон)	Файл, що містить HTML-розмітку, яка динамічно генерується **View** для відправки користувачеві.	Відображення даних.

Встановлення та підготовка

Перед початком роботи з Django, необхідно встановити **Python**. Далі виконайте наступні кроки.

1. Встановлення пакетного менеджера pip

Pip — це стандартний менеджер пакетів для Python. Зазвичай він встановлюється разом з Python. Ви можете перевірити його версію:

```
pip -V
```

Якщо pip не встановлено, його можна встановити або оновити за допомогою команд:

```
python -m ensurepip --upgrade
python -m pip install --upgrade pip
```

2. Робота з віртуальним середовищем (venv)

Використання віртуального середовища є **обов'язковим** для професійної розробки. Це ізольоване середовище, яке дозволяє встановлювати пакети для кожного проекту окремо, уникнувши конфліктів залежностей між різними проектами.

1. **Створіть віртуальне середовище:**

```
python -m venv .venv
```

Це створить каталог .venv у вашій поточній директорії.

2. **Активуйте віртуальне середовище:**
   - **Windows (командна строка):**

```
.venv\Scripts\activate.bat
```

   - **Windows (PowerShell):**

```
.venv\Scripts\Activate.ps1
```

   - **Linux/macOS:**

```
source .venv/bin/activate
```

   Після активації, назва середовища ((.venv)) з'явиться на початку командного рядка.
3. Встановіть Django:
   Перебуваючи в активованому середовищі, виконайте команду:

```
pip install django
```

Щоб встановити певну версію, використайте:

```
pip install django~=4.2
```

(Тут ~= означає "найновіша версія 4.2.x")

4. **Перевірте версію:**

```
python -m django --version
```

Ви побачите встановлену версію Django.

Створення першого проєкту та додатку

Проект Django складається з одного або декількох додатків. Проект — це загальна конфігурація, а додаток — це певний функціонал.

1. Створення проєкту

Використовуйте *django-admin startproject*, щоб створити новий проект.

```
# Перейдіть до каталогу, де ви хочете створити проект.
# У нас це буде каталог C:\django
cd C:\django

# Створюємо новий проект Django з назвою "myproject"
# Замість "myproject" ви можете використати будь-яку іншу назву
django-admin startproject myproject
```

Ця команда створить каталог myproject з такою структурою:

Файл/Каталог	**Призначення**
manage.py	**Утиліта командного рядка.** Головний скрипт для управління проектом. З його допомогою ми запускаємо сервер (runserver), створюємо додатки (startapp), працюємо з міграціями та багато іншого.
myproject/	**Каталог проєкту.** Це контейнер для всіх конфігураційних файлів проєкту. Його ім'я має співпадати з назвою проєкту.

init.py	**Пакет Python.** Цей порожній файл вказує, що каталог myproject є Python-пакетом.
settings.py	**Налаштування проєкту.** Тут ми вказуємо, які додатки використовуються, налаштовуємо базу даних, статичні файли, мову та часовий пояс.
urls.py	**Мапа URL-адрес проєкту.** Цей файл містить основну мапу маршрутизації, яка направляє вхідні запити до відповідних додатків.
wsgi.py	**Конфігурація WSGI.** Це стандартний інтерфейс для розгортання веб-додатків на сумісних серверах.
asgi.py	**Конфігурація ASGI.** Це новий стандарт для асинхронних веб-серверів, що дозволяє працювати з WebSockets та іншими асинхронними протоколами.

2. Запуск сервера розробки

Після створення проекту, його можна запустити та перевірити.

```
# Перейдіть до каталогу вашого проекту, де знаходиться manage.py
cd myproject

# Запускаємо вбудований сервер розробки Django.
# Це дозволить вам бачити ваш сайт локально у браузері.
python manage.py runserver
```

Сервер буде запущено на адресі http://127.0.0.1:8000/. Відкрийте її у браузері. Ви повинні побачити стандартну вітальну сторінку Django.

3. Створення додатку

Щоб додати функціонал, потрібно створити додаток.

```
# Переконайтесь, що ви перебуваєте в каталозі проекту, де знаходиться manage.py
# Створюємо додаток з назвою "hello"
python manage.py startapp hello
```

Це створить каталог hello з файлами:

Файл/Каталог	**Призначення**
migrations/	**Міграції бази даних.** Тут зберігаються скрипти, які описують зміни у структурі бази даних, що дозволяють синхронізувати її з моделями.
admin.py	**Налаштування адмін-панелі.** У цьому файлі ви реєструєте свої моделі, щоб вони відображалися в автоматично згенерованій адмін-панелі.
apps.py	**Конфігурація додатку.** Містить клас, який конфігурує цей додаток. Зазвичай цей файл не потребує змін.
models.py	**Моделі даних.** Тут ви описуєте структуру даних вашого додатку (наприклад, Article, User, Product). Кожен клас моделі відповідає таблиці в базі даних.
tests.py	**Тести.** Файл для написання автоматичних тестів, які перевіряють коректність роботи вашого додатку.
views.py	**Логіка додатку.** Містить функції, які отримують запити від користувачів, обробляють їх, взаємодіють з моделями

	та шаблонами і повертають відповідь.

4. Реєстрація додатку

Щоб Django "побачив" ваш додаток, потрібно зареєструвати його у файлі settings.py.

```
# файл: myproject/settings.py

# Відкрийте цей файл і знайдіть масив INSTALLED_APPS.
# Додайте назву нашого нового додатку до цього списку.
# Важливо: назва додатку має бути у лапках та з комою в кінці.

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'hello', # Наш новий додаток
]
```

5. Налаштування View та URL-адрес

Крок 5.1: Створення View

View — це функція, яка отримує запит від користувача та створює відповідь.

```
# файл: hello/views.py

# Імпортуємо HttpResponse, щоб повертати просту текстову відповідь.
from django.http import HttpResponse

# Створюємо функцію з назвою 'index', яка приймає один аргумент 'request'.
# 'request' - це об'єкт, який містить інформацію про запит користувача.
def index(request):
    # Повертаємо об'єкт HttpResponse з текстом, який побачить користувач у
браузері.
    return HttpResponse("Ласкаво просимо до нашого першого веб-додатку на
Django!")
```

Крок 5.2: Налаштування маршрутизації (URL-адрес)

Тепер нам потрібно "сказати" Django, що коли користувач заходить на певний URL, він має викликати нашу функцію index.

```
# файл: myproject/urls.py

# Імпортуємо функцію path для створення маршрутів
# Імпортуємо модуль views з нашого додатку 'hello'
from django.urls import path
from hello import views

# urlpatterns - це список, який містить усі маршрути нашого проекту
urlpatterns = [
    # Визначаємо маршрут для кореня нашого сайту ('')
    # Другий аргумент - це функція views.index, яка буде обробляти цей маршрут
    # name='home' - це унікальне ім'я для маршруту, яке може бути корисним у майбутньому
    path('', views.index, name='home'),
]
```

Після цих змін знову запустіть сервер командою *python manage.py runserver*, і ви побачите свій перший сайт, створений на Django!

Практичні завдання для закріплення матеріалу

1. **Додайте другу сторінку:**
   - У файлі hello/views.py створіть нову функцію about(request), яка повертатиме HttpResponse з текстом "Це сторінка про нас.".
   - У файлі myproject/urls.py додайте новий маршрут path('about/', views.about, name='about'), щоб сторінка була доступна за адресою http://127.0.0.1:8000/about/.
2. **Створіть новий додаток:**
   - Створіть додаток з назвою pages за допомогою команди python manage.py startapp pages.
   - Зареєструйте його у файлі myproject/settings.py.
   - Створіть у ньому файл urls.py та views.py і додайте свій власний маршрут та

сторінку.

3. **Спробуйте інший текст:**
    - Змініть текст, який повертає функція index у файлі hello/views.py. Запустіть сервер і переконайтеся, що зміни відобразилися в браузері.

Підсумок:
Ми крок за кроком створили і запустили перший проєкт Django. Розібралися, як:

1. Створювати проєкт та додаток.
2. Розуміти призначення кожного файлу.
3. Реєструвати додаток у налаштуваннях проєкту.
4. Писати першу функцію View.
5. Налаштовувати маршрутизацію (URL-адреси), щоб зв'язати URL-адресу з функцією View.

Тепер ви можете спробувати створити інший додаток або додати ще одну сторінку до існуючого.