Введение в Django

Что такое Django?

Django — это высокоуровневый фреймворк для веб-разработки на языке Python. Его главная цель — облегчить разработку сложных веб-приложений, предоставляя множество встроенных инструментов и библиотек. Django помогает создавать веб-сайты быстрее, безопаснее и с минимальными усилиями благодаря философии "Don't Repeat Yourself" (DRY) — принципу, который направлен на сокращение дублирования кода.

Ключевые особенности Django

- **Модульность:** Django разделяет проект на логические части: модели (данные), представления (логика) и шаблоны (интерфейс). Этот подход называется паттерном MVC (Model-View-Controller) или MVT в контексте Django (Model-View-Template).
- **Встроенная панель администратора:** Django автоматически генерирует административную панель для управления данными.
- **ORM (Object-Relational Mapping):** Django позволяет взаимодействовать с базой данных через Python-код, избегая написания SQL-запросов.
- **Безопасность:** Django обеспечивает высокую безопасность, помогая разработчикам избежать распространенных уязвимостей, таких как SQL-инъекции, межсайтовые запросы и XSS-атаки.

- **Гибкость:** Django подходит для создания небольших проектов, а также крупных, сложных систем.

Процесс создания веб-приложения на Django

Шаг 1: Установка Django

Для начала нужно установить Django с помощью pip:

```
pip install django
```

Шаг 2: Создание проекта

Django позволяет быстро создать проект с основной структурой, выполнив команду:

```
django-admin startproject myproject
```

Эта команда создаст директорию myproject, которая содержит следующие файлы:

- manage.py — команда для управления проектом (запуск сервера, миграции и т.д.).
- myproject/ — директория с настройками проекта, включая файлы settings.py, urls.py.

Шаг 3: Запуск сервера разработки

После создания проекта можно запустить встроенный сервер для разработки:

```
python manage.py runserver
```

Сервер запускается по адресу http://127.0.0.1:8000/.

Шаг 4: Создание приложения

В Django проект может состоять из одного или нескольких приложений. Для создания приложения выполните команду:

```
python manage.py startapp shop
```

Это создаст директорию shop, которая будет содержать файлы для моделей, представлений и шаблонов.

Шаг 5: Миграции и базы данных

Django использует систему миграций для работы с базами данных. После создания модели нужно выполнить команды для создания соответствующих таблиц в базе данных:

```
python manage.py makemigrations
python manage.py migrate
```

Шаг 6: Создание суперпользователя

Для доступа к панели администратора создайте суперпользователя:

```
python manage.py createsuperuser
```

Структура проекта Django

После создания проекта Django у вас появилась структура с несколькими важными файлами и директориями. Рассмотрим их подробнее, чтобы лучше понять их функции и роль в проекте:

1. manage.py

Этот файл находится в корне вашего проекта. Его основная цель — это инструмент для управления вашим проектом Django. С помощью manage.py можно выполнять различные команды, такие как:

- Запуск локального сервера разработки:

  ```
  python manage.py runserver
  ```

- **Применение миграций:**

  ```
  python manage.py migrate
  ```

- **Создание приложений:**

  ```
  python manage.py startapp myapp
  ```

- **Создание суперпользователя для административной панели:**

  ```
  python manage.py createsuperuser
  ```

Проще говоря, manage.py служит интерфейсом между проектом и Django-фреймворком, позволяя вам взаимодействовать с проектом через консоль.

2. name_project (основная директория вашего проекта)

Эта папка содержит конфигурационные файлы и основную логику работы проекта.

- __init__.py

Это пустой файл, который сообщает Python, что данная папка является модулем. Это важно, чтобы Python мог правильно импортировать содержимое этой директории как модуль.

- asgi.py

Этот файл нужен для настройки ASGI (Asynchronous Server Gateway Interface), который позволяет Django работать с асинхронными запросами. ASGI — это более современный интерфейс между сервером и приложением, поддерживающий WebSockets и другие асинхронные протоколы. Если вы работаете над проектом, где важно асинхронное взаимодействие (например, чат в реальном времени), этот файл будет использоваться для настройки.

- wsgi.py

Этот файл необходим для настройки WSGI (Web Server Gateway Interface) — интерфейса, который обеспечивает взаимодействие между сервером и вашим приложением. WSGI — это стандарт, который позволяет Django работать с веб-серверами (например,

Apache или Nginx). Когда вы развертываете ваш проект на сервере, wsgi.py будет отвечать за запуск вашего приложения.

- settings.py

Один из самых важных файлов в Django-проекте. Этот файл содержит все настройки проекта, такие как:

- Подключение к базе данных.
- Настройки приложений, используемых в проекте.
- Настройки локализации и часовых поясов.
- Безопасность проекта (настройка ключа шифрования, разрешенные хосты и т.д.).
- Пути к статическим и медиа-файлам. Этот файл можно считать "центром управления" проектом. Пример конфигурации базы данных в settings.py:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

- urls.py

Этот файл отвечает за маршрутизацию в Django. Здесь вы указываете, какие URL-адреса ведут к каким представлениям. Например, вы можете настроить URL для отображения страницы с товарами:

Роль каждого файла

- **manage.py**: интерфейс для управления проектом.
- **__init__.py**: помечает папку как модуль Python.
- **asgi.py и wsgi.py**: конфигурационные файлы для взаимодействия с серверами.
- **settings.py**: хранит все настройки проекта.
- **urls.py**: отвечает за маршрутизацию в проекте.
- **db.sqlite3**: файл базы данных SQLite (если используется база по умолчанию).
- **.venv**: директория виртуальной среды для изоляции зависимостей проекта.