«Django без боли: руководство для новичков»

Автор : Эркинов Амантур Эркинович

1. **Введение**
   * Что такое Django?
   * Почему именно Django?
   * Установка и настройка среды разработки.
2. **Первый проект**
   * Создание проекта.
   * Структура проекта Django.
   * Первый запуск приложения.
   * Создание приложения myapp
   * Настройка статических-файлов
   * **Настройка медиа-файлов**
3. **Модели и базы данных**
   * Работа с моделями.
   * Изменение встроенной модели пользователя (Custom User)
   * Миграции.
   * Административная панель Django.
4. **Создание представлений (Views)**
   * Функциональные и классовые представления.
   * Обработка запросов и формирование ответов.
5. **Шаблоны (Templates)**
   * Основы работы с шаблонами.
   * Django Template Language (DTL).
   * Организация интерфейса.
6. **Формы**
   * Стандартные формы Django.
   * Работа с формами и валидацией.
7. **Пользователи и аутентификация**
   * Регистрация, вход и выход.
   * Система прав и групп пользователей.
8. **Работа с внешними библиотеками**
   * Django Rest Framework (DRF).
   * Django Allauth.
9. **Развертывание**
   * Развёртывание на Heroku.
   * Настройка Gunicorn и Nginx.
10. **Заключение**
    * Лучшие практики разработки.
    * Рекомендации для дальнейшего изучения.
11. **Письмо от автора**
12. Привет, дорогой читатель!
13. Если ты открыл эту книгу, значит, у тебя появилось желание понять Django — один из самых популярных фреймворков для веб-разработки. И это уже первый шаг к большому достижению! Позволь мне тебя поздравить: ты на правильном пути.
14. Я написал эту книгу не для гениев программирования и не для тех, кто уже может создать сложное веб-приложение с закрытыми глазами. Эта книга для тех, кто хочет научиться, но боится, что это слишком сложно. Я преподаю Django уже не первый год и знаю, какие страхи испытывают новички: «Это слишком трудно», «Я не пойму», «У меня не получится». Поверь, ты не один такой.
15. Но вот что я тебе скажу: программирование — это не магия. Это просто набор инструкций, которые компьютер понимает. А Django — это инструмент, который делает процесс создания веб-приложений гораздо проще, чем ты думаешь. Всё, что тебе нужно, — это терпение, практика и немного любопытства.
16. Эта книга проведёт тебя за руку от самого начала: установки Django до твоего первого полноценного проекта. Мы будем разбираться с базами данных, создавать страницы, настраивать пользователей и даже готовить приложение к публикации. Без заумных слов и сложных теорий — только практика, понятные примеры и минимум боли.
17. Не бойся ошибаться. Ошибки — это не враги, а твои лучшие учителя. Каждый баг, каждая непонятная строка кода — это шаг вперёд. Главное — не сдаваться.
18. В конце этой книги ты удивишься, насколько далеко ты смог продвинуться. И, кто знает, может быть, именно с Django начнётся твоя большая карьера в мире IT?
19. Удачи тебе! Я верю в тебя. А теперь — пора писать код! 🚀
20. С уважением,  
    Амантур

**Введение**

1. **Что такое Django?**

Django — это высокоуровневый фреймворк для веб-разработки на языке Python, который позволяет создавать мощные, безопасные и масштабируемые веб-приложения с минимальными усилиями. Его главная философия — это принцип «не повторяйся» (DRY — Don't Repeat Yourself) и «все включено» (batteries included), что делает процесс разработки более удобным и эффективным.

С помощью Django вы сможете создавать веб-сайты, блоги, интернет-магазины, социальные сети и многие другие типы приложений. Фреймворк предоставляет множество встроенных инструментов: систему аутентификации, админ-панель, поддержку работы с базами данных, обработку форм и многое другое.

Основные преимущества Django:

* **Простота использования**: Чёткая документация и логичная структура делают фреймворк доступным для новичков.
* **Безопасность**: Django автоматически защищает приложения от наиболее распространённых угроз, таких как SQL-инъекции и межсайтовая подделка запросов.
* **Масштабируемость**: Подходит как для небольших проектов, так и для высоконагруженных систем.

Изучая Django, вы не только освоите веб-разработку, но и научитесь писать чистый, поддерживаемый код. Приступим к созданию вашего первого проекта!

### 1.2 Почему именно Django?

Django выделяется среди других фреймворков благодаря своему богатому функционалу, удобству использования и высокой производительности. Вот несколько причин, почему Django становится выбором многих разработчиков по всему миру:

1. **Полный набор инструментов из коробки**: Django предлагает множество встроенных функций, таких как аутентификация пользователей, управление сессиями, обработка форм, работа с базами данных и админ-панель. Вам не нужно тратить время на интеграцию сторонних решений — всё необходимое уже включено.
2. **Простота в освоении**: Благодаря чёткой структуре проекта и подробной документации, Django становится доступным даже для начинающих программистов. Разработка веб-приложений не требует глубоких знаний архитектуры, а логика фреймворка интуитивно понятна.
3. **Высокий уровень безопасности**: Безопасность в Django стоит на первом месте. Фреймворк защищает от таких угроз, как SQL-инъекции, межсайтовая подделка запросов (CSRF), межсайтовый скриптинг (XSS) и кликджекинг. Эти встроенные механизмы позволяют разработчикам сосредоточиться на логике приложения, не беспокоясь о безопасности.
4. **Масштабируемость и гибкость**: Django подходит как для небольших проектов, так и для крупных корпоративных систем с высокой нагрузкой. Его модульная структура позволяет легко расширять функциональность, а поддержка различных баз данных делает его универсальным решением.
5. **Активное сообщество и множество ресурсов**: Django имеет огромное сообщество разработчиков по всему миру, которые создают плагины, расширения и пишут обучающие материалы. Если у вас возникнут трудности, вы всегда сможете найти ответ в официальной документации или на форумах.
6. **Популярность в индустрии**: Django используется такими крупными компаниями, как Instagram, Pinterest, Disqus и Spotify. Этот факт подтверждает надёжность и производительность фреймворка на практике.
7. **Соответствие принципам Python**: Django наследует все преимущества Python: читаемость кода, простота синтаксиса и широкие возможности для автоматизации процессов. Это делает фреймворк удобным инструментом как для разработки MVP (минимально жизнеспособного продукта), так и для крупных долгосрочных проектов.

Изучая Django, вы не только освоите веб-разработку, но и научитесь писать чистый, поддерживаемый код. Приступим к созданию вашего первого проекта!

* 1. **Установка и настройка среды разработки**

Для начала работы с Django необходимо установить Python, Django и настроить среду разработки. Вот пошаговое руководство:

1. **Установка Python**
   * Перейдите на официальный сайт Python: <https://www.python.org/downloads/>.
   * Скачайте последнюю стабильную версию Python.
   * Установите Python, не забыв поставить галочку напротив "Add Python to PATH".
2. **Установка виртуального окружения**
   * Откройте командную строку (Terminal/Command Prompt).
   * Введите команду:

python -m venv myenv

* + Активируйте окружение:
    - На Windows: myenv\Scripts\activate
    - На Mac/Linux: source myenv/bin/activate

1. **Установка Django**
   * В активированном окружении выполните команду:

pip install django

* + Проверьте успешную установку:

python -m django --version

1. **Установка среды разработки**
   * Рекомендуем установить Visual Studio Code: <https://code.visualstudio.com/>.
   * Установите расширения "Python" и "Django".

Теперь у вас есть всё необходимое для начала работы с Django. В следующем разделе мы создадим ваш первый проект!

**2. Первый проект**

**2.1 Создание проекта**

Теперь создадим первый Django-проект с помощью самой простой команды. Следуйте инструкциям:

1. **Создание проекта в текущей директории**
   * Откройте командную строку в папке, где хотите разместить проект.
   * Введите команду:

django-admin startproject myproject .

* + Обратите внимание на точку в конце команды — она указывает Django создать проект прямо в текущей папке.

1. **Структура проекта** После выполнения команды структура вашего проекта будет выглядеть так:
2. ./
3. manage.py
4. myproject/
5. \_\_init\_\_.py
6. settings.py
7. urls.py
8. asgi.py

wsgi.py

1. **Запуск сервера разработки**
   * Запустите сервер разработки следующей командой:

python manage.py runserver

* + Если всё сделано правильно, в браузере по адресу http://127.0.0.1:8000/ появится приветственная страница Django.

Поздравляю! Вы успешно создали свой первый Django-проект с помощью команды django-admin startproject myproject ..

### 2.2 Подробное описание структуры проекта

* **manage.py** — вспомогательный скрипт для управления проектом: запуск сервера, создание миграций, запуск тестов и т.д.
* **myproject/** — основная папка проекта, содержащая важные настройки и конфигурации.
  + **init.py** — файл, делающий папку модулем Python.
  + **settings.py** — основной файл настроек Django, в котором прописываются базы данных, приложения, middleware и прочее.
  + **urls.py** — файл, определяющий маршруты URL-адресов для вашего приложения.
  + **asgi.py** — точка входа для асинхронных серверов.
  + **wsgi.py** — точка входа для традиционных веб-серверов.

1. **Запуск сервера разработки**
   * Запустите сервер разработки следующей командой:

python manage.py runserver

* + Если всё сделано правильно, в браузере по адресу http://127.0.0.1:8000/ появится приветственная страница Django.

Поздравляю! Вы успешно создали свой первый Django-проект с помощью команды django-admin startproject myproject ..

### 2.3 Первый запуск приложения

После создания проекта и настройки среды давайте запустим наше первое приложение. Следуйте инструкциям:

1. **Запуск сервера разработки**
   * Убедитесь, что находитесь в корневой папке проекта.
   * Введите команду:

python manage.py runserver

* + Если всё прошло успешно, вы увидите сообщение о запуске сервера с адресом <http://127.0.0.1:8000/>.

1. **Проверка работы приложения**
   * Откройте браузер и перейдите по адресу <http://127.0.0.1:8000/>.
   * Если всё сделано правильно, на экране появится стандартная приветственная страница Django с сообщением "The install worked successfully! Congratulations!".
2. **Остановка сервера**
   * Для остановки сервера нажмите сочетание клавиш Ctrl+C в командной строке.

Поздравляем! Вы успешно запустили своё первое Django-приложение. В следующих разделах мы разберём, как добавить своё собственное приложение и настроить маршрутизацию.

### 2.4 Создание приложения myapp

Теперь создадим наше первое приложение в проекте. В Django приложения — это модули, которые отвечают за определённые части функционала. Чтобы создать приложение с именем myapp, выполните следующие шаги:

1. **Создание приложения**
   * В командной строке в корневой папке проекта выполните команду:

python manage.py startapp myapp

1. **Структура созданного приложения** После выполнения команды структура папки myapp будет следующей:
2. myapp/
3. migrations/
4. \_\_init\_\_.py
5. \_\_init\_\_.py
6. admin.py
7. apps.py
8. models.py
9. tests.py

views.py

1. **Регистрация приложения в проекте**
   * Откройте файл settings.py.
   * Найдите список INSTALLED\_APPS.
   * Добавьте в него строку:

'myapp',

Теперь приложение myapp готово к работе! В следующем разделе мы разберём, как создать первое представление и отобразить его в браузере.

### 2.5 Настройка статических файлов

Для работы со статическими файлами (CSS, JavaScript, изображения) в Django выполните следующие шаги:

1. **Создание структуры для статических файлов**
   * В корне проекта создайте папку static:

mkdir static

1. **Настройки в settings.py**
   * Откройте файл settings.py и добавьте следующие строки:
   * STATIC\_URL = '/static/'
   * STATICFILES\_DIRS = [
   * BASE\_DIR / 'static',

]

1. **Подключение статических файлов в шаблонах**
   * В файле шаблона добавьте тег загрузки статических файлов:
   * {% load static %}

<link rel="stylesheet" href="{% static 'css/style.css' %}">

1. работки:
   * from django.conf import settings
   * from django.conf.urls.static import static

urlpatterns += static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

Теперь ваши статические файлы готовы к использованию в проекте!

* 1. **Настройка медиа-файлов**

Для работы с медиа-файлами (загружаемыми пользователем изображениями, документами и другими файлами) выполните следующие шаги:

* 1. **Создание структуры для медиа-файлов**
     + В корне проекта создайте папку media:

mkdir media

* 1. **Настройки в settings.py**
     + Откройте файл settings.py и добавьте следующие строки:
     + MEDIA\_URL = '/media/'

MEDIA\_ROOT = BASE\_DIR / 'media'

* 1. **Подключение медиа-файлов в urls.py**
     + В файле urls.py добавьте следующий код после статических файлов:

urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

Теперь ваши статические и медиа-файлы готовы к использованию в проекте!

### 3.1Работа с моделями

Модели в Django представляют собой описание структуры базы данных с помощью кода на Python. Каждая модель соответствует одной таблице в базе данных, а её атрибуты — столбцам таблицы.

Модели хранятся в файле models.py внутри каждого приложения, например, для приложения myapp путь будет следующим:

myapp/models.py

Когда вы создаёте модель, Django автоматически создаёт SQL-запросы для создания, изменения и удаления таблиц в базе данных.

### Как работают модели

1. **Определение модели**
   * Каждая модель является классом, унаследованным от django.db.models.Model.
   * Атрибуты класса — это поля таблицы, а их типы определяют тип данных в базе.

Пример простой модели для блога:

from django.db import models

class Post(models.Model):

title = models.CharField(max\_length=200)

content = models.TextField()

created\_at = models.DateTimeField(auto\_now\_add=True)

def \_\_str\_\_(self):

return self.title

1. **Создание миграций**
   * После добавления или изменения моделей необходимо создать и применить миграции.
   * Для создания миграций используйте команду:

python manage.py makemigrations

* + Для применения миграций:

python manage.py migrate

1. **Работа с моделями в Python**
   * Django предоставляет интерфейс ORM для работы с базой данных через Python-код.

Примеры операций с моделью Post:

# Импорт модели

from myapp.models import Post

# Создание записи

post = Post.objects.create(title='Первый пост', content='Привет, Django!')

# Получение всех записей

posts = Post.objects.all()

# Поиск записи

post = Post.objects.get(id=1)

# Обновление записи

post.title = 'Обновлённый заголовок'

post.save()

# Удаление записи

post.delete()

Теперь вы знаете, как создавать модели и работать с ними. В следующем разделе рассмотрим более детально типы полей и их возможности.

### 3.2Изменение встроенной модели пользователя (Custom User)

Django предоставляет встроенную модель пользователя, но в реальных проектах часто требуется добавить дополнительные поля или изменить логику аутентификации. Для этого создадим кастомную модель пользователя.

#### 1. Создание кастомной модели пользователя

* Откройте файл models.py вашего приложения myapp и добавьте следующий код:

from django.contrib.auth.models import AbstractUser

from django.db import models

class CustomUser(AbstractUser):

first\_name = models.CharField(max\_length=30, blank=True, null=True)

last\_name = models.CharField(max\_length=30, blank=True, null=True)

username = models.CharField(max\_length=150, unique=True)

gender = models.CharField(max\_length=10, choices=[('M', 'Мужчина'), ('F', 'Женщина')], blank=True, null=True)

phone\_number = models.CharField(max\_length=15, blank=True, null=True)

birth\_date = models.DateField(blank=True, null=True)

def \_\_str\_\_(self):

return self.username

#### 2. Настройка модели пользователя в настройках

* Откройте файл settings.py и добавьте следующую строку:

AUTH\_USER\_MODEL = 'myapp.CustomUser'

#### 3. Создание миграций

* Создайте миграции для обновления базы данных:

python manage.py makemigrations

python manage.py migrate

#### 4. Обновление администратора Django

* Чтобы кастомный пользователь отображался в административной панели, измените файл admin.py приложения myapp:

from django.contrib import admin

from django.contrib.auth.admin import UserAdmin

from .models import CustomUser

class CustomUserAdmin(UserAdmin):

fieldsets = UserAdmin.fieldsets + (

("Дополнительная информация", {"fields": ("first\_name", "last\_name", "username", "gender", "phone\_number", "birth\_date")}),

)

admin.site.register(CustomUser, CustomUserAdmin)

#### 5. Создание суперпользователя

python manage.py createsuperuser

При создании суперпользователя будут запрошены стандартные поля и дополнительные поля, если они обязательны.

Теперь ваша кастомная модель пользователя готова к использованию!