**Django. Создание первого проекта**

Крупные компании и амбициозные стартапы часто выбирают фреймворк Django для своих продуктов и сервисов из-за его широких возможностей и высокой скорости работы, а разработчики на Django востребованы на рынке труда. Вместе с fullstack-разработчицей Анной Агабекян и программистом из Amazon Андреем Берендой разберемся, в чем секрет популярности Django и напишем серверную часть простого блога.

**Что такое Django**

Django — бесплатный [фреймворк](https://blog.skillfactory.ru/glossary/framework/) для разработки веб-приложений на языке Python. С его помощью можно быстро создавать безопасные сайты, которые легко поддерживать в дальнейшем.

Практически каждый сайт состоит из фронтенда, который пользователь видит каждый раз при переходе по ссылке, и бэкенда — скрытой части, которая работает на стороне сервера. Эта часть хранит в себе данные пользователей, обрабатывает контент и связывается со сторонними службами. Фреймворк позволяет упростить, ускорить и стандартизировать разработку бэкенда.

В Django есть множество шаблонов веб-разработки, а значит, в процессе не придется заново писать код часто используемых модулей — к примеру, компонентов панели управления сайтом, инструментов аутентификации пользователей, хранения данных и отправки форм.

[**Fullstack-разработчик на Python**](https://skillfactory.ru/python-fullstack-web-developer?utm_source=blog&utm_medium=knowledge-base&utm_campaign=none_coding_fpw_blog_knowledge-base_course_none_none_all_sf_django-s-chego-nachat-obychenie_sidebanner_picture_fix&utm_content=django-s-chego-nachat-obychenie&utm_term=sidebanner_picture_fix)

Первая версия Django вышла в 2005 году, и до сих пор фреймворк остается популярным. Среди плюсов Django — частый выход обновлений и подробная документация. Также у него большое и активное сообщество, которое всегда готово помочь новичкам. Кроме того, у фреймворка есть много обучающих материалов, курсов и книг, часть из которых полностью бесплатна.

**Где используют Django**

Django используют не только стартапы и любители, но и крупные компании. Благодаря скорости работы и удобству его применяют в Spotify, YouTube, Reddit, поиске Google, DropBox и Pinterest.

Разработчики на Django востребованы на рынке труда и могут найти предложения о работе в мировых IT-компаниях. В зависимости от направления бизнеса Django-разработчик может быть:  
— [Fullstack-разработчиком](https://blog.skillfactory.ru/kak-stat-fullstack-razrabotchikom/" \t "_blank);  
— [Backend-разработчиком](https://blog.skillfactory.ru/chto-nuzhno-znat-chtoby-nachat-rabotat-backend-razrabotchikom/" \t "_blank);  
— Разработчиком [API](https://blog.skillfactory.ru/glossary/api/).

В США Django-разработчик с опытом работы от 1 до 3 лет [может претендовать](https://www.glassdoor.com/Salaries/django-developer-salary-SRCH_KO0,16.htm) на 79 000 долларов в год, а зарплаты высококвалифицированных программистов с богатым опытом доходят до 160 000 долларов в год. В России Django-разработчикам [предлагают](https://hh.ru/search/vacancy?area=113&search_field=name&enable_snippets=true&text=Django&ored_clusters=true&order_by=salary_desc) от 60 000 рублей на позиции Junior до 450 000 для Senior-специалистов.

**Является ли Django гибким?**

Веб-фреймворки часто можно поделить на "гибкие" и "негибкие".

Негибкие - это те, у которых есть "правильный путь" для решения какой-либо конкретной задачи. Они часто поддерживают быстрое развёртывание в определённой области (решение проблем определённого типа), потому что правильный способ сделать что-либо обычно хорошо понимается и хорошо документируется. Однако они могут быть менее гибкими при решении проблем за пределами их основной сферы и, как правило, предлагают меньше вариантов того, какие компоненты и подходы они могут использовать.

Напротив, у гибких фреймворков гораздо меньше ограничений на лучший способ склеивания компонентов для достижения цели или даже того, какие компоненты следует использовать. Они облегчают разработчикам использование наиболее подходящих инструментов для выполнения конкретной задачи, хотя и за счёт того, что вам нужно самим найти эти компоненты.

Django «умеренно гибкий» и, следовательно, обеспечивает «лучшее из обоих миров». Он предоставляет набор компонентов для обработки большинства задач веб-разработки и один (или два) предпочтительных способа их использования. Однако такая архитектура Django означает, что вы обычно можете выбирать из нескольких различных опций или при необходимости добавлять поддержку для совершенно новых.

[**Как выглядит код Django?**](https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/Introduction#%D0%BA%D0%B0%D0%BA_%D0%B2%D1%8B%D0%B3%D0%BB%D1%8F%D0%B4%D0%B8%D1%82_%D0%BA%D0%BE%D0%B4_django)

На традиционном информационном веб-сайте веб-приложение ожидает HTTP-запросы от веб-браузера (или другого клиента). Когда запрос получен, приложение разрабатывает то, что необходимо на основе URL-адреса и, возможно, данных в POST или GET запросах. В зависимости от того, что требуется, далее он может читать или записывать информацию из базы данных или выполнять другие задачи, необходимые для удовлетворения запроса. Затем приложение вернёт ответ веб-браузеру, часто динамически создавая HTML-страницу для отображения в браузере, вставляя полученные данные в HTML-шаблон.

Веб-приложения, написанные на Django, обычно группируют код, который обрабатывает каждый из этих шагов, в отдельные файлы:



* URLs: Хотя можно обрабатывать запросы с каждого URL-адреса с помощью одной функции, гораздо удобнее писать отдельную функцию для обработки каждого ресурса. URL-маршрутизатор используется для перенаправления HTTP-запросов в соответствующее представление на основе URL-адреса запроса. Кроме того, URL-маршрутизатор может извлекать данные из URL-адреса в соответствии с заданным шаблоном и передавать их в соответствующую функцию отображения (view) в виде аргументов.
* View: View (англ. «отображение») — это функция обработчика запросов, которая получает HTTP-запросы и возвращает ответы. Функция view имеет доступ к данным, необходимым для удовлетворения запросов, и делегирует ответы в шаблоны через модели.
* Models: Модели представляют собой объекты Python, которые определяют структуру данных приложения и предоставляют механизмы для управления (добавления, изменения, удаления) и выполнения запросов в базу данных.
* Templates: Template (англ. «шаблон») — это текстовый файл, определяющий структуру или разметку страницы (например HTML-страницы), с полями для подстановки, которые используются для вывода актуального содержимого. View может динамически создавать HTML-страницы, используя HTML-шаблоны и заполняя их данными из модели (model). Шаблон может быть использован для определения структуры файлов любых типов, не обязательно HTML.

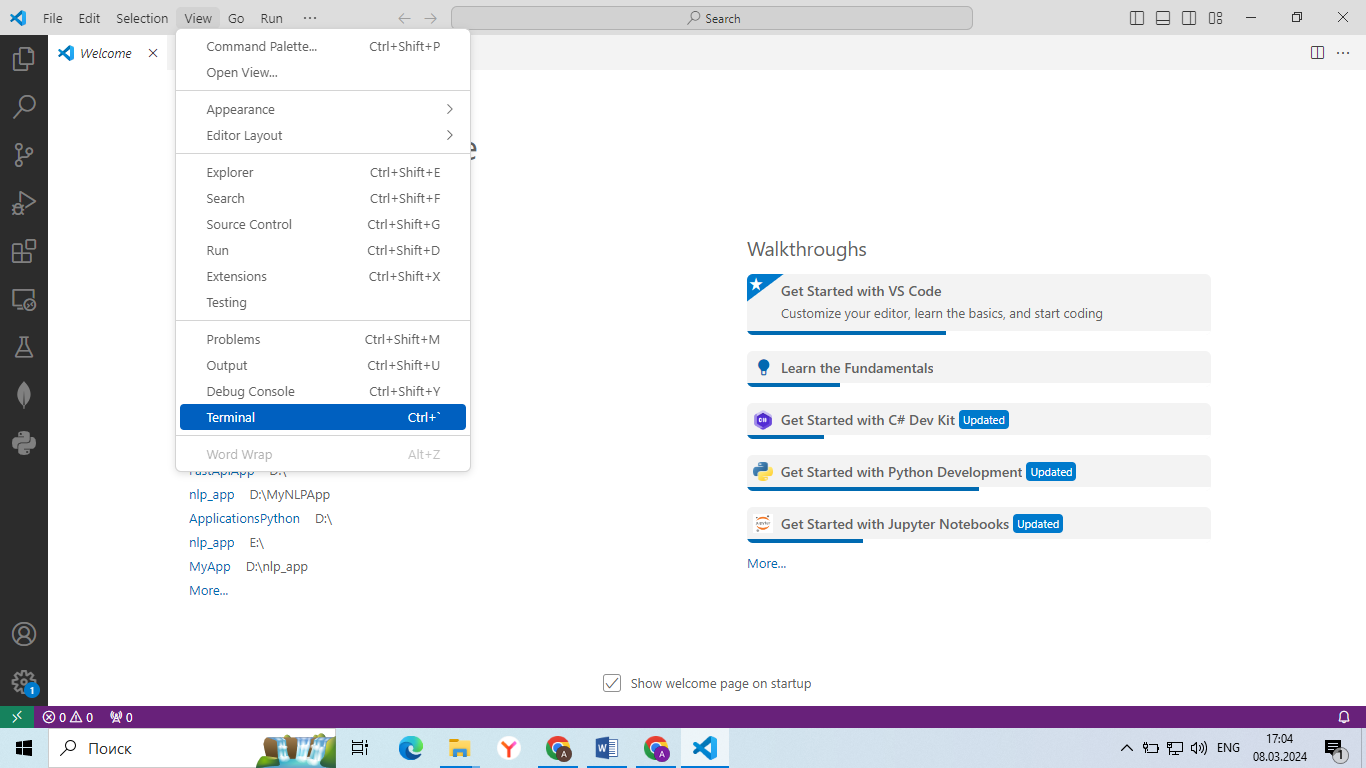
*Примечание: Django реализует уровневую архитектуру "Model View Template (MVT)". Она имеет много общего с более известной архитектурой*[*Model View Controller*](https://developer.mozilla.org/ru/docs/Web/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture)*.*

Следующие разделы дадут вам понимание того, как выглядят основные части Django (мы их изучим более детально чуть позже на курсе, когда будет настраивать окружение разработчика).

**Для создания и запуска Django приложения**

Для создания и запуска Django приложения с использованием виртуальной среды в Visual Studio Code:

Шаг 1. Откройте терминал в VS Code.



Перейдите на диск D:

cd D:

Создайте новую папку с названием myFirstDjangoApp:

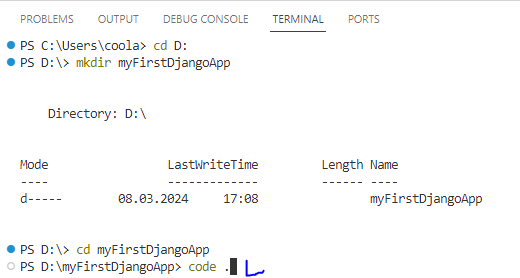
mkdir myFirstDjangoApp

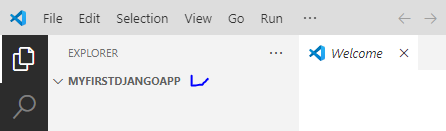
Перейти в созданную папку:

cd myFirstDjangoApp

Откройте эту папку в VS Code:

code .





Шаг 2: Установка Django и создание виртуальной среды

Убедитесь, что у вас установлен Python и инструмент virtualenv. Если нет, установите их с помощью следующих команд:

pip install django

pip install virtualenv





Шаг 3: Создание виртуальной среды и активация

Откройте терминал в VS Code, перейдите в папку, где будет расположен ваш проект, и создайте виртуальную среду:

virtualenv venv

Активируйте виртуальную среду:

На Windows:

venv\Scripts\activate

На macOS и Linux:

source venv/bin/activate

Шаг 4: Установка Django в виртуальной среде

Установите Django в вашу виртуальную среду:

pip install django

Шаг 5: Создание проекта Django

Создайте новый проект Django в той же папке:

django-admin startproject myproject



Созданный каталог (проекта Django) будет состоять из следующих элементов:

* **manage.py**: выполняет различные команды проекта, например, создает и запускает приложение
* myproject - собственно папка проекта myproject, которая содержит следующие файлы:
* **\_\_init\_\_.py**: данный файл указывает, что папка, в которой он находится, будет рассматриваться как модуль. Это стандартный файл для программы на языке Python.
* **settings.py**: содержит настройки конфигурации проекта
* **urls.py**: содержит шаблоны URL-адресов, по сути определяет систему маршрутизации проекта
* **wsgi.py**: содержит свойства конфигурации WSGI (Web Server Gateway Inerface). Он используется при развертывании проекта.
* **asgi.py**: название файла представляет сокращение от Asynchronous Server Gateway Interface и расширяет возможности WSGI, добавляя поддержку для взаимодействия между асинхронными веб-серверами и приложениями.

Шаг 6: Создание приложения Django

Перейдите в папку проекта и создайте новое приложение Django:

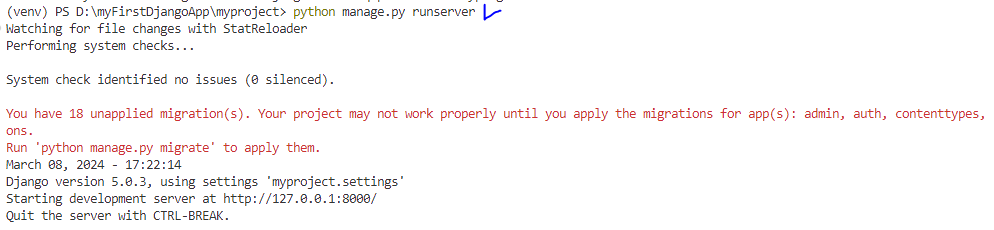
cd myproject

python manage.py startapp myapp

Шаг 7: Запуск сервера разработки

Запустите сервер разработки Django с помощью следующей команды:

python manage.py runserver



Шаг 8: Миграция

Из шага видно, что наш сервер разработки Django успешно запущен и работает на адресе http://127.0.0.1:8000/. Однако, вам рекомендуется применить все накопленные миграции для приложений Django, чтобы ваше приложение работало правильно.

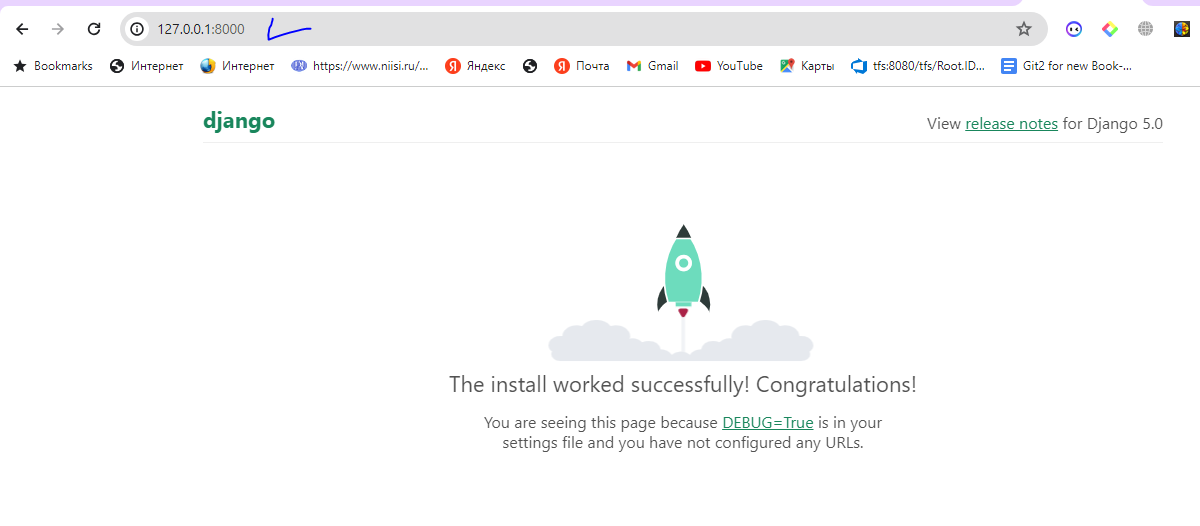
Для применения накопленных миграций выполните следующую команду в терминале, находясь в директории вашего проекта:

python manage.py migrate

Эта команда применит все накопленные миграции для приложений Django, таких как admin, auth, contenttypes, sessions, и ваше приложение будет готово к использованию без проблем.

Шаг 9: Просмотр приложения в браузере

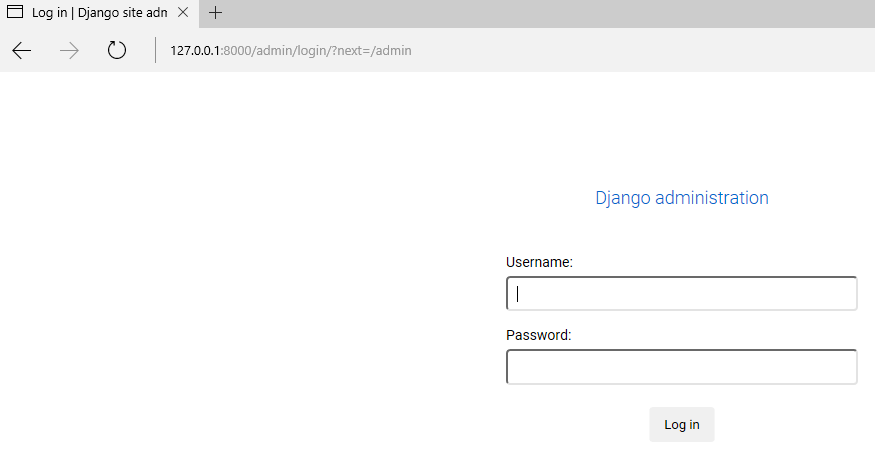
Откройте браузер и перейдите по адресу http://127.0.0.1:8000/, чтобы увидеть ваше новое Django приложение.



Теперь у вас есть базовое Django приложение, созданное и запущенное с использованием виртуальной среды в Visual Studio Code.

## Настройка входа администратора

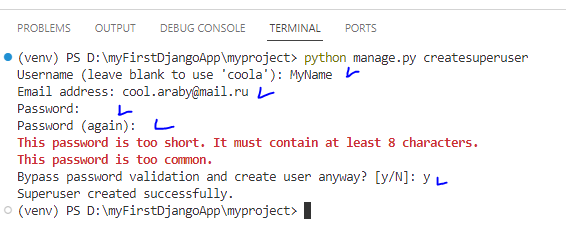
По умолчанию Django поставляется с готовой панелью администратора, которую вы можете использовать. Он находится по адресу http://127.0.0.1:8000/admin и выглядит примерно так:



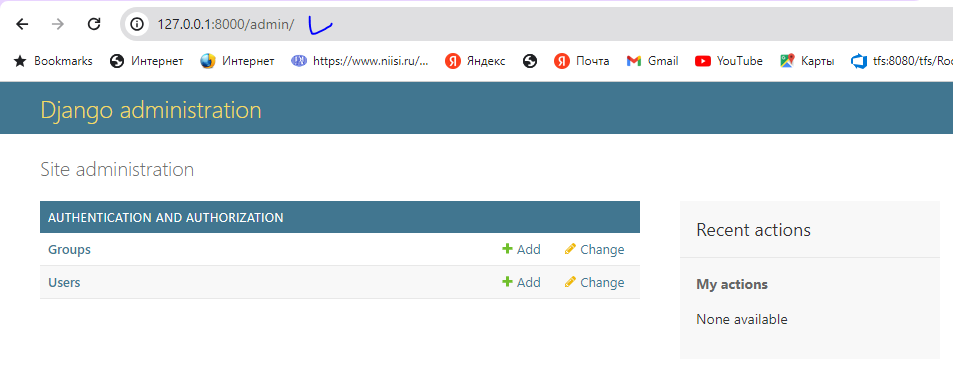
Запустите команду createsuperuser, чтобы создать нового пользователя для вашего приложения Django. Вот команда для него:

|  |
| --- |
| python manage.py createsuperuser |

В командной строке вы можете ввести свое имя пользователя, адрес электронной почты, а затем пароль. Так как это на вашем локальном компьютере, вы можете использовать слабый пароль, который выдаст слишком распространенное сообщение о пароле. Вы можете обойти это, набрав y при появлении запроса. Как только это будет завершено, вы успешно создадите суперпользователя. Сообщение.



Теперь вы можете снова запустить сервер, используя python manage.py runserver. Обновите браузер, и вы сможете войти в систему, используя свои учетные данные. Вот как выглядит экран зарегистрированного администратора:



**Создание первого приложения**

Веб-приложение или проект Django состоит из отдельных приложений. Вместе они образуют полноценное веб-приложение. Каждое приложение представляет какую-то определенную функциональность или группу функциональностей. Один проект может включать множество приложений. Это позволяет выделить группу задач в отдельный модуль и разрабатывать их относительно независимо от других. Кроме того, мы можем переносить приложение из одного проекта в другой независимо от другой функциональности проекта.

При создании проекта он уже содержит несколько приложений по умолчанию.

* **django.contrib.admin**
* **django.contrib.auth**
* **django.contrib.contenttypes**
* **django.contrib.sessions**
* **django.contrib.messages**
* **django.contrib.staticfiles**

Список всех приложений можно найти в проекте в файле **settings.py** в переменной **INSTALLED\_APPS**:

|  |
| --- |
| INSTALLED\_APPS = [      'django.contrib.admin',      'django.contrib.auth',      'django.contrib.contenttypes',      'django.contrib.sessions',      'django.contrib.messages',      'django.contrib.staticfiles',  ] |

И, конечно, мы можем создавать свои приложения, которые реализуют определенный функционал. Для создания приложения в проекте применяется команда

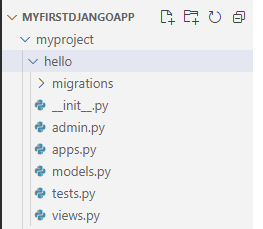
python manage.py startapp название\_приложения

Так, создадим в проекте, созданном в прошлой теме или новом, первое приложение, которое будет называться hello. Для этого выполним в командной строке/терминале следующую команду:

python manage.py startapp hello



В результате в проекте Django будет создано приложение hello. И после выполнения этой команды мы увидим в проекте новую папку, которое будет хранить все файлы созданного приложения:

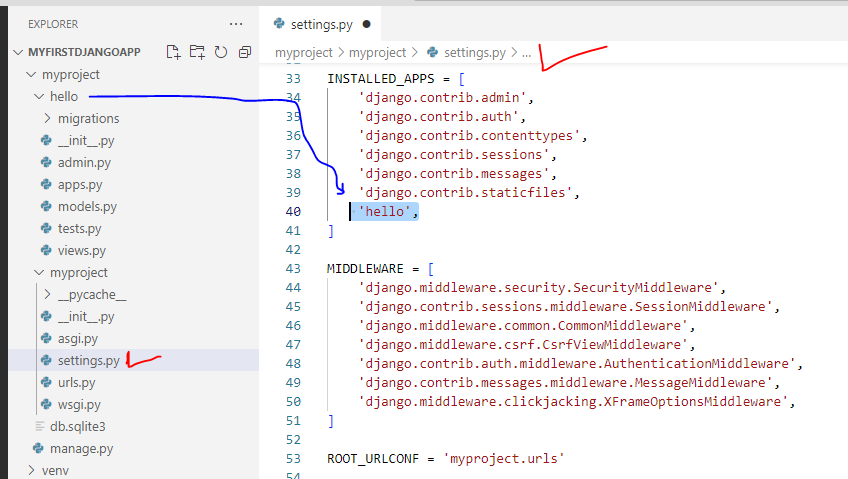


Рассмотрим вкратце его структуру:

* папка **migrations**: предназначена для хранения миграций - скриптов, которые позволяют синхронизировать структуру базы данных с определением моделей
* **\_\_init\_\_.py**: указывает интерпретатору python, что текущий каталог будет рассматриваться в качестве пакета
* **admin.py**: предназначен для административных функций, в частности, здесь призводится регистрация моделей, которые используются в интерфейсе администратора
* **apps.py**: определяет конфигурацию приложения
* **models.py**: хранит определение моделей, которые описывают используемые в приложении данные
* **tests.py**: хранит тесты приложения
* **views.py**: определяет функции, которые получают запросы пользователей, обрабатывают их и возвращают ответ

Но пока приложение никак не задействуется. Его надо зарегистрировать в проекте Django. Для этого откроем файл **settings.py** и добавим в конец массива INSTALLED\_APPS наше приложение:

|  |
| --- |
| INSTALLED\_APPS = [      'django.contrib.admin',      'django.contrib.auth',      'django.contrib.contenttypes',      'django.contrib.sessions',      'django.contrib.messages',      'django.contrib.staticfiles',      'hello',  ] |



В проекте может быть несколько приложений, и каждое из них надо добавлять таким вот образом.

Теперь определим какие-нибудь простейшие действия, которые будет выполнять данное приложение, например, отправлять в ответ пользователю строку "Hello World".

Для этого перейдем в проекте приложения **hello** к файлу **views.py**, который по умолчанию должен выглядеть следующим образом:

|  |
| --- |
| from django.shortcuts import render  # Create your views here. |

Изменим код следующим образом:

|  |
| --- |
| from django.http import HttpResponse    def index(request):      return HttpResponse("Hello World!") |

В данном случае мы импортируем класс **HttpResponse** из стандартного пакета *django.http*. Затем определяется функция index(), которая в качестве параметра получает объект запроса request. Класс HttpResponse предназначен для создания ответа, который отправляется пользователю. И с помощью выражения return HttpResponse("Hello World!") мы отправляем пользователю строку " Hello World!"

Теперь также в основном проекте Django откроем файл **urls.py**, который позволяет сопоставить маршруты с представлениями, которые будут обрабатывать запрос по этим маршрутам. По умолчанию этот файл выглядит следующим образом:

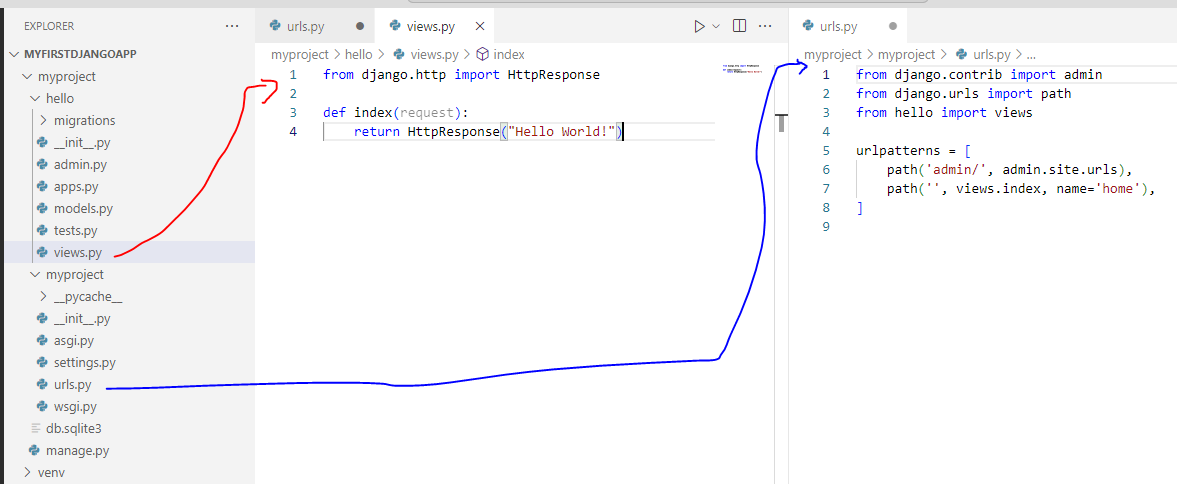
|  |
| --- |
| from django.contrib import admin  from django.urls import path    urlpatterns = [      path('admin/', admin.site.urls),  ] |

Первой строкой из модуля *django.contrib* импортируется класс AdminSite, который предоставляет возможности работы с интерфейсом администратора. Второй строкой из модуля *django.urls* импортируется функция **path**. Эта функция задает сопоставление определенного маршрута с функцией обработки. Так, в данном случае маршрут "admin/" будет обрабатываться методом admin.site.urls.

Но выше мы определили функцию index в файле **views.py**, который возвращает пользователю строку "Hello World!". Поэтому изменим файл **urls.py** следующим образом:

|  |
| --- |
| from django.urls import path  from hello import views    urlpatterns = [      path('', views.index, name='home'),  ] |

Чтобы использовать функцию views.index вначале импортируем модуль views. Затем определяем сопоставление маршрута ' ' и функции views.index и также дополнительно имя для маршрута: name='home'. По сути маршрут ' ' будет сопоставляться с запросом к корню приложения.



Теперь снова запустим приложение командой

python manage.py runserver

И вновь перейдем в браузере по адресу *http://127.0.0.1:8000/*, и браузер нам отобразит строку "Hello World!":

