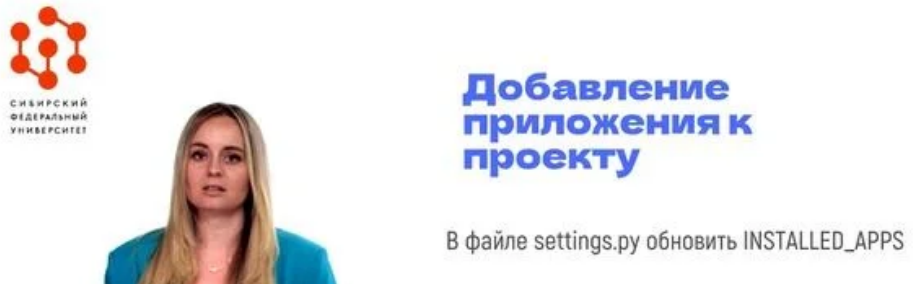
Разработка программного обеспечения на языке Python

Лекция 2. Библиотека Django

Посмотрите видеоуроки и ответьте на контрольные вопросы после лекции

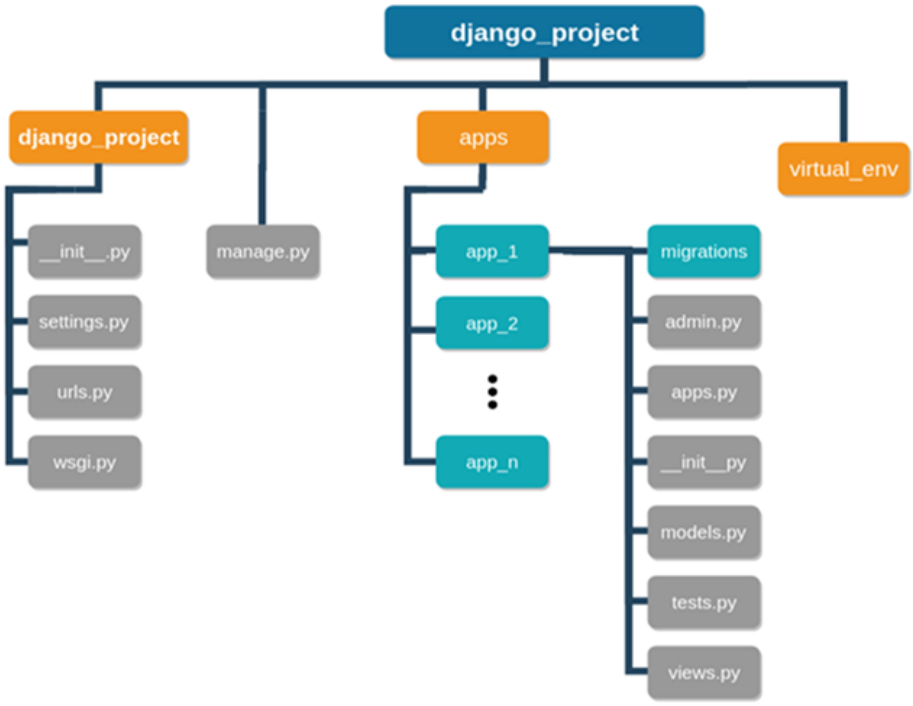
Создание приложения в django



00:00 / 04:51

в данной теме рассмотрим добавление приложений в проект django.

Проект Django имеет общую файловую структуру.

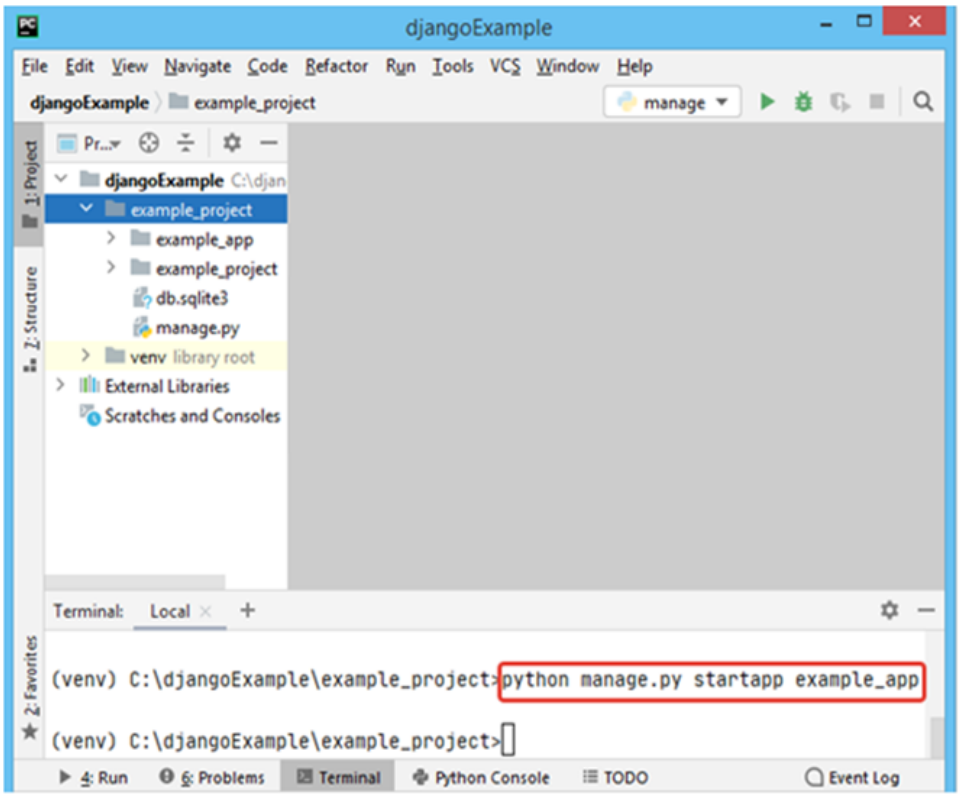


Одно из важнейшей ее составляющей – приложения. Проект django состоит из одного или нескольких приложений, каждое из которых выполняет свою задачу. Приложение Django – это отдельный пакет Python, который содержит набор связанных файлов для определенной цели. Основное преимущество приложений - возможность их повторного использования, подключения к другому проекту.

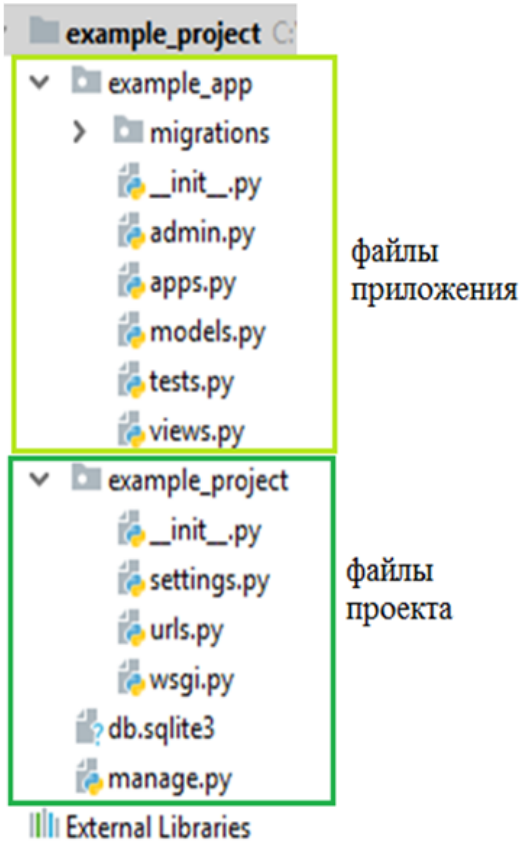
Создадим в проекте, созданном в прошлой теме или новом, приложение. Для этого выполним в командной строке/ терминале следующую команду:

python manage.py startapp [имя приложения].

Имя приложение указывается после команды startapp.

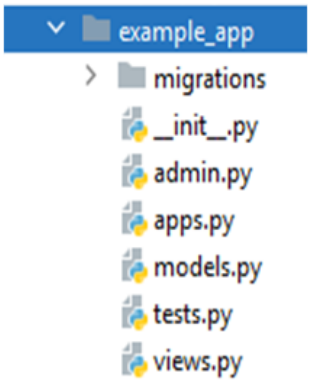


В результате в проект Django будет добавлена новая папка, в которой содержатся все файлы созданного приложения, созданные автоматически. В нашем примере, она называется Example_app.



Таким образом, проект будет состоять из следующих элементов: папки проекта (структура которого была рассмотрена в предыдущей теме), файла manage.py для управления проектом, а также добавляется папка с файлами приложения.

Рассмотрим структуру папок и файлов приложения подробнее:



Файл **__init__.py**: указывает интерпретатору python, что текущий каталог будет рассматриваться в качестве библиотеки.

Файл **models.py**: хранит определение моделей, которые описывают используемые в приложении данные.

Папка **migrations** предназначена для хранения миграций - скриптов, которые позволяют синхронизировать структуру базы данных с описанием моделей.

Файл **admin.py:** предназначен для административных функций, в частности, здесь производится регистрация моделей, которые используются в интерфейсе администратора.

Файл **views.py**: определяет функции, которые получают запросы пользователей, обрабатывают их и возвращают ответ.

Файл **apps.py**: определяет конфигурацию приложения.

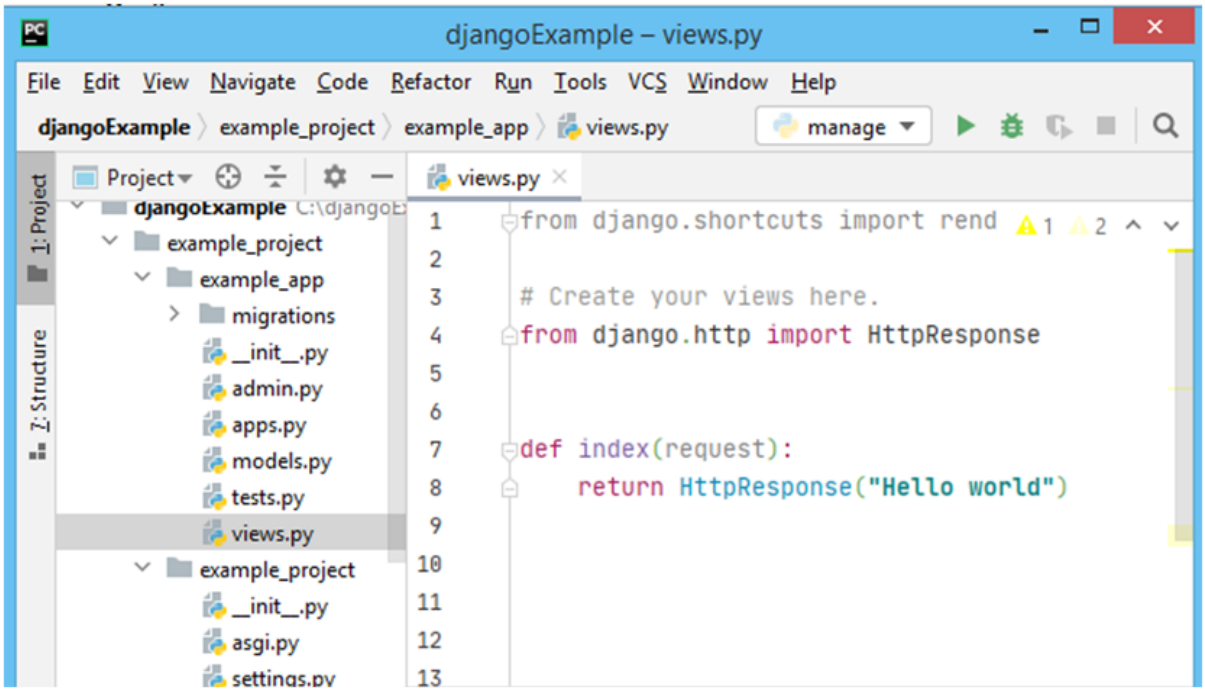
Файл **tests.py**: хранит тесты приложения.

При создании проекта он уже содержит несколько приложений по умолчанию. Список всех приложений можно найти в проекте в файле settings.py в переменной INSTALLED_APPS. Созданное приложение требуется зарегистрировать в проекте Django. Для этого добавим его в список INSTALLED_APPS. (на слайде добавляемый код выделен красным цветом и полужирным шрифтом).

```
INSTALLED_APPS = [

'example_app',

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

]
```

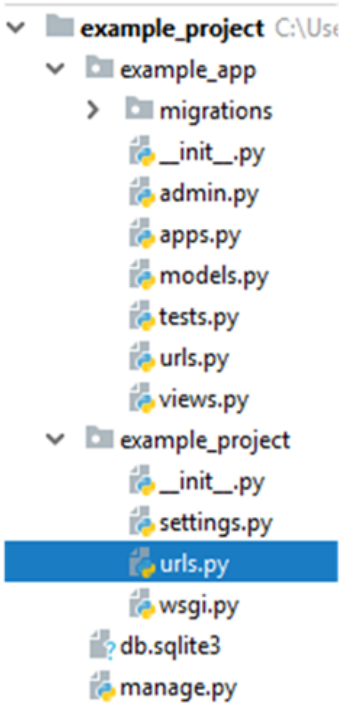
Теперь определим какие-нибудь действия, которые будет выполнять данное приложение, например, отправлять в ответ пользователю строку "Hello World".

Для этого перейдем в приложении ExampleApp к файлу views.py.



Изменим код следующим образом: импортируем класс HttpResponse из стандартной библиотеки django.http. Затем создадим функцию 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 указанную строку.

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
from django.conf.urls import include

urlpatterns = [
    path('admin/', admin.site.urls),


]
```

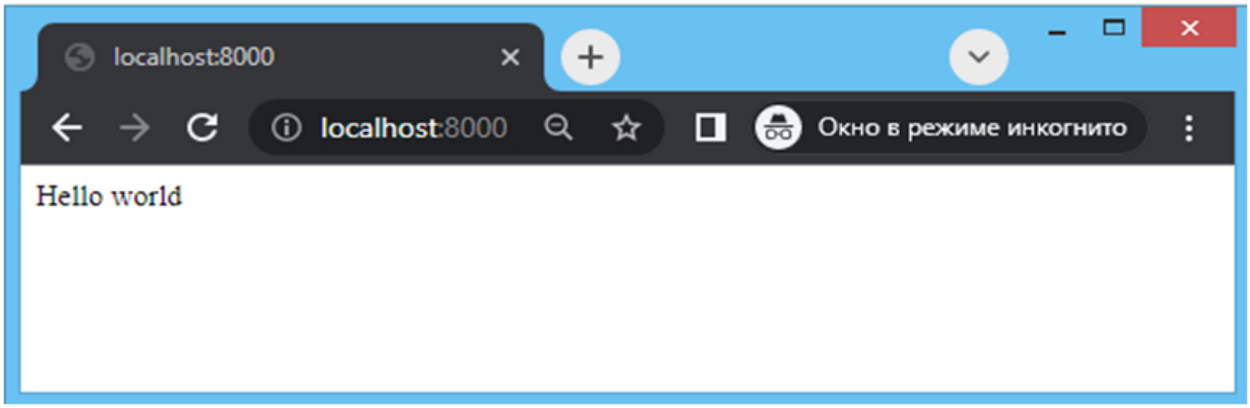
Первой строкой из модуля django.contrib импортируется класс AdminSite, который предоставляет возможности работы с интерфейсом администратора. Второй строкой из модуля django.urls импортируется функция path. Эта функция задает сопоставление определенного маршрута с функцией обработки. Так, в данном случае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

Ранее мы создали функцию в файле views.py index, которая возвращает пользователю строку "Hello world". Поэтому изменим файл urls.py следующим образом:

```
from django.contrib import admin
from django.urls import path
from example_app import views

urlpatterns = [
  path('admin/', admin.site.urls),
  path('', views.index)
]
```

Чтобы использовать функцию index вначале импортируем модуль views. Затем определяем сопоставление маршрута в виде пустой строки (' ') и функции views.index. По сути данный маршрут будет сопоставляться с запросом к корню приложения.

Теперь снова запустим приложение командой python manage.py runserver. И перейдем в браузере по адресу http://127.0.0.1:8000/, и браузер нам отобразит строку ("Hello world").



Подведем итоги, в данной теме мы добавили в проект django приложение, в котором в дальнейшем будет содержаться основной функционал: в частности, будут добавлены маршруты сайта и обработчики данных маршрутов. Кроме того, в приложении будут описываться таблицы базы данных в виде моделей.

Маршрутизация

ПРЕДЫДУЩИЙ ЭЛЕМЕНТ КУРСА

◄ Задание 2. Создание flask приложения

Перейти на...

СЛЕДУЮЩИЙ ЭЛЕМЕНТ КУРСА

Задание 3. Создание своего первого сайта на Django ►