**Создание HTML-шаблонов и шаблонизатор Jinja**

Давайте создадим несколько html-шаблонов. Так как целью нашей работы является именно создание своего сайта, то необходимы именно html-файлы вместо простого вывода текста командой HttpResponse. Функция render предназначена для вывода готовых шаблонов через запросы от сервера. Напишем её в файле views.py.

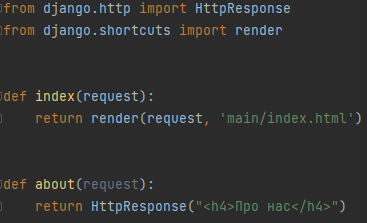


Рисунок 1 - Команда render для вывода

начальной HTML-страницы

Создадим директорию с названием templates, в которой будут храниться все созданные нами шаблоны. В этой директории создадим ещё дополнительно папку main, которая понадобится, чтобы разграничить шаблоны между приложениями. Далее создадим файл index.html, в котором будем описывать главную страницу нашего проекта.

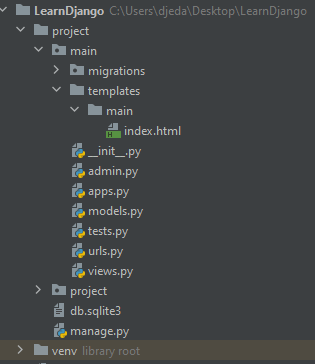


Рисунок 2 - Создание в папке main файла index.html

В созданном html-файле напишем код. Для более быстрого заполнения в пустой файл напишем ! и нажмём клавишу табуляции. Файл заполнится следующим кодом:

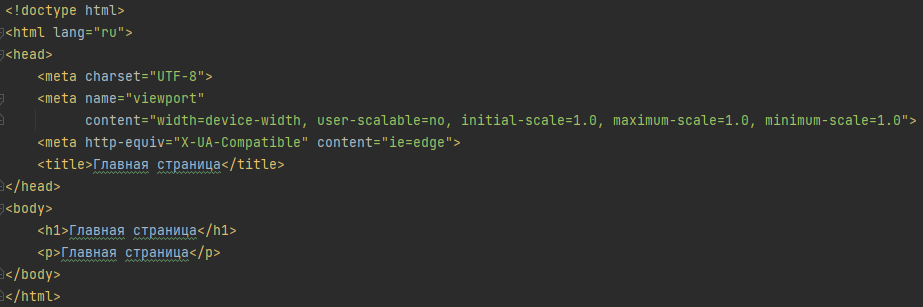


Рисунок 3 - Файл index.html

Запишем в теге <title> название страницы “Главная страница”, тоже самое напишем в теге <body>. Далее запустим наш проект через терминал, введя ранее использованную в предыдущей лабораторной команду, не забыв перейти в директорию project:

cd project

python manage.py runserver

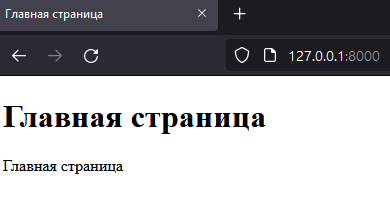


Рисунок 4 - Результат вывода страницы index.html

Проделаем тоже самое для страницы about, поменяв только заголовок и тело в html-файле на “Про нас”.

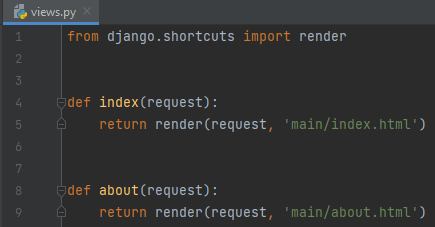


Рисунок 5 - Команда render для вывода

страницы about

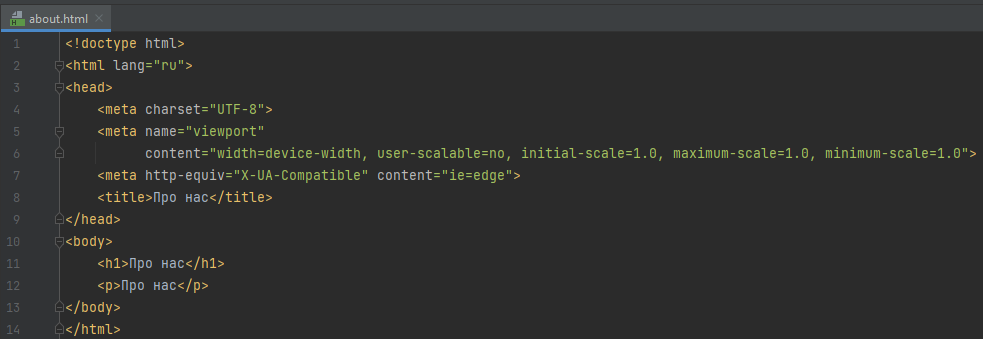


Рисунок 6 - Файл about.html

Перейдём на страницу about для проверки, добавив в адресную строку /about.

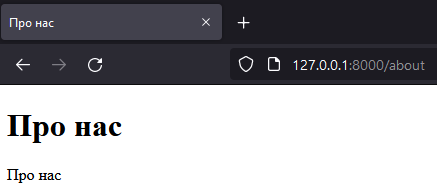


Рисунок 7 - Результат вывода страницы about.html

Как видим, у нас имеются одинаковые части кода в двух html-файлах. В дальнейшем мы можем сократить количество кода, создав отдельный html-файл, в котором отдельно выделим изменяющиеся части - блоки/секции. Эти блоки мы и будем описывать в html-файлах.

Создадим файл base.html, который будет служить базовым шаблоном для всех других файлов. Запишем в него почти такой же код как в файле index.html или about.html, но с некоторыми изменениями. В теге <title> при помощи Jinja-шаблонов укажем изменяющийся блок. Обозначим начало блока {% block <название блока> %} и конец блока {% endblock %}. Тоже пропишем для тега <body>, поменяв название блока.

****

Рисунок 8 - Файл base.html

Очистим файлы index.html и about.html. Так как мы создали базовый шаблон, то в этих файлах мы будем писать только то, что хотим отобразить на определённой странице. Ссылаться на базовый шаблон base.html будем при помощи Jinja-шаблона {% extends ‘<путь до файла>’ %}.

Теперь опишем, что мы хотим видеть на главной странице, внутри блоков заголовка и тела страницы. Для этого снова откроем блок заголовка {% block title %}, напишем какой-то текст и закроем его {% endblock %}. Тоже самое проделаем с блоком содержимого тела страницы.

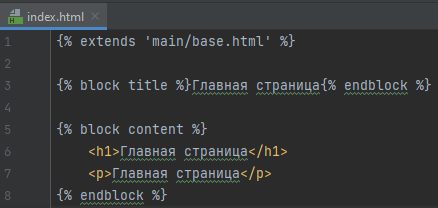
****

Рисунок 9 - Файл index.html

Те же действия проделываем и для файла about.html

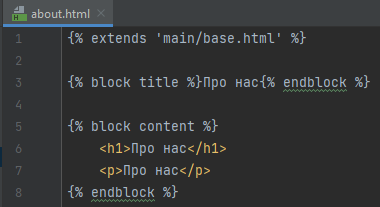
****

Рисунок 10 - Файл about.html

При проверке страниц в браузере, результат не изменится, при этом кода в файлах станет меньше ввиду отсутствия повторяющихся его частей.

Также существует в Jinja-шаблоне тег *include*, который позволяет импортировать непосредственно внутрь одного файла содержимое другого файла. Импортировать можно неограниченное количество раз.

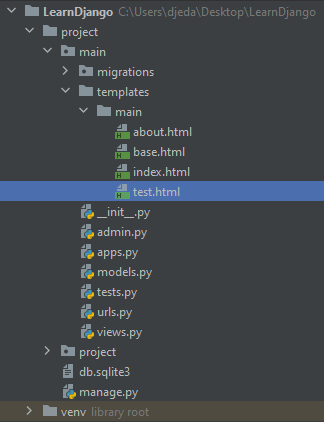
****

Рисунок 11 - Создание файла test.html

****

Рисунок 12 - Подключаемый файл

с каким-то содержимым

Импортируем содержимое файла test.html внутрь страницы index

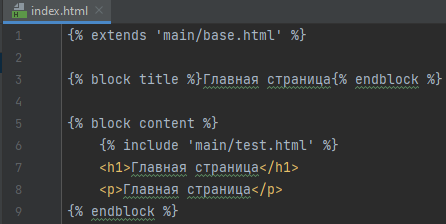
****

Рисунок 13 - Тег include в коде файла index.html

Проверим данный тег на уже запущенном нами сервере, просто обновив главную страницу.

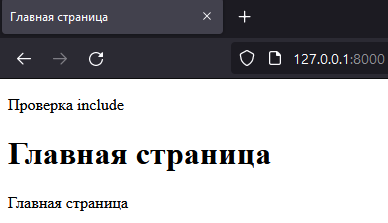
****

Рисунок 14 - Проверка include