# **Лабораторная работа №2**

*Создание web-страницы с текстом*

Для выполнения данной лабораторной работы создайте директорию lab2, а в ней, с помощью командной строки и команды: django-admin.py startproject firstwebpage создайте новый проект **firstwebpage**.

Каждый проект в Django – это набор нескольких приложений, каждое из которых выполняет определенную задачу. Чтобы добавить в наш проект новое приложение, необходимо перейдите в созданную директорию **firstwebpage** и выполнить команду: python manage.py startapp app\_name, где app\_name – это имя подключаемого приложения.

Вам необходимо подключить приложение **flatpages**. Это приложение позволяет вам управлять статическими страницами через интерфейс администратора Django и указывать шаблоны для таких страниц с помощью шаблонной системы Django.

**python manage.py startapp flatpages**

В начале работы над новым проектом необходимо задать базовые настройки, например, задать имя базы данных и создать ее таблицы (см. Лабораторную работу №1), а также добавить нужные приложения в проект.

#############################

В файле settings.py измените параметр базы данных. Переменной DATABASES есть внутренний словарь с ключами ENGINE и NAME. Для параметра NAME установите следующее значение:

'NAME': os.path.join(BASE\_DIR, 'db\_project\_name')

**В нём же указать import os перед патчем**

– данный параметр является названием базы данных, так что он может быть любым.

Чтобы создать таблицы базы данных, необходимо перейти в директорию project\_name и через командную строку выполнить команду:

**python3 manage.py migrate**

**(3-нужна, когда несколько версий питона стоит)**

Затем создаем суперпользователя (при создании необходимо ввести логин, email и пароль пользователя):

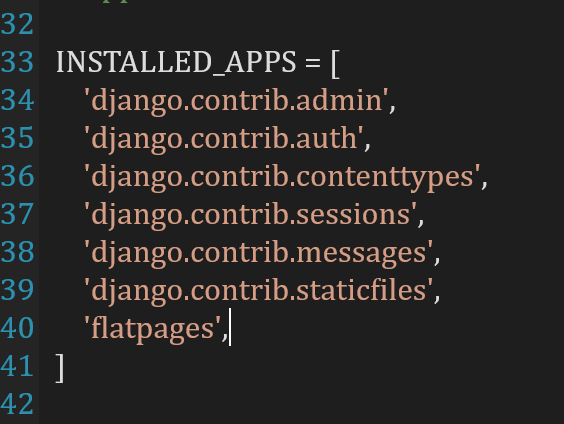
**python3 manage.py createsuperuser**

Теперь необходимо запустить сервер, если он не был запущен до этого, командой:

**python3 manage.py runserver**

################################

Чтобы добавить приложение **flatpages** в ваш проект, откройте файл settings.py, найдите кортеж INSTALLED\_APPS и добавьте в конец элемента строку 'flatpages'.(просто запихнуть, без дополнительных файлов)



Перед началом работы с файлом **urls.py** необходимо импортировать views нашего приложения:

from flatpages import views

Для будущей страницы создайте новый адрес в файле **urls.py**:

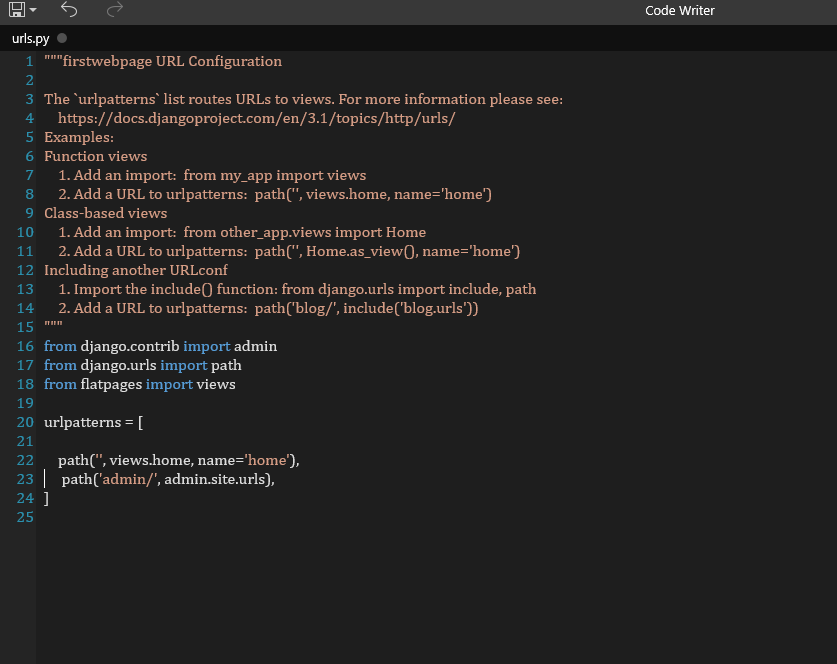
urlpatterns = [

path('admin/', admin.site.urls),

path('', views.home, name='home'),

]

Второй аргумент, передаваемы в функцию, указывает на то, что функция представления, которая будет обрабатывать запросы по этому адресу, находится в файле **views** и называется **home**.

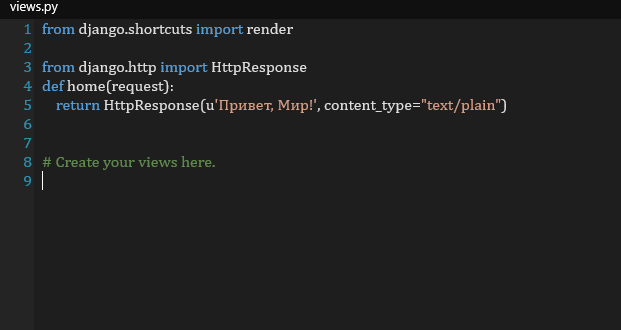


Для того, чтобы в будущем при обращении генерировался ответ, необходимо создать функцию home в файле **views.py** в директории **flatpages**.

from django.http import HttpResponse

def home(request):

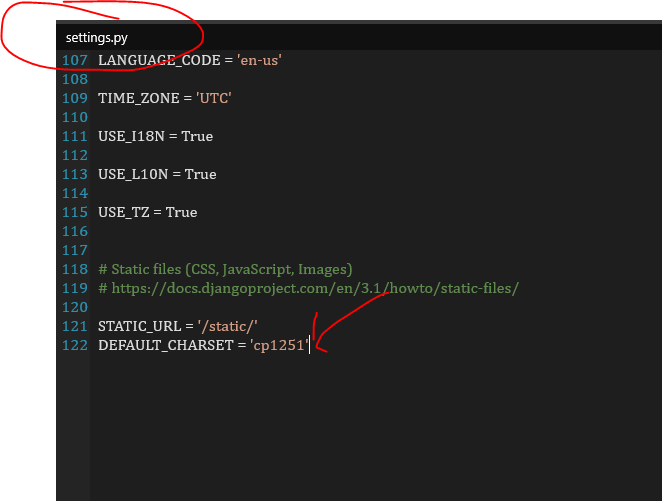
return HttpResponse(u'Привет, Мир!', content\_type="text/plain")



Сначала происходит импортирование специального класса HttpResponse, любая функция представления должна возвращать переменные, где хранятся представители именно этого класса или класса, наследующего HttpResponse. В данном случае функция представления **home** является типичными примером представлений в Django – она принимает на входе объект запроса и возвращает на выходе объект ответа. В нашем случае в теле объекта ответа будет находится простая строка 'Привет, Мир!', а в качестве типа ответа указан простой текст. Теперь можно зайти на страницу по адресу http://127.0.0.1:8000/ и посмотреть результат.

Запуск сервера:

**python manage.py runserver**

Отоброжение русского и могучего 

в setting.py добавь строку в любое место

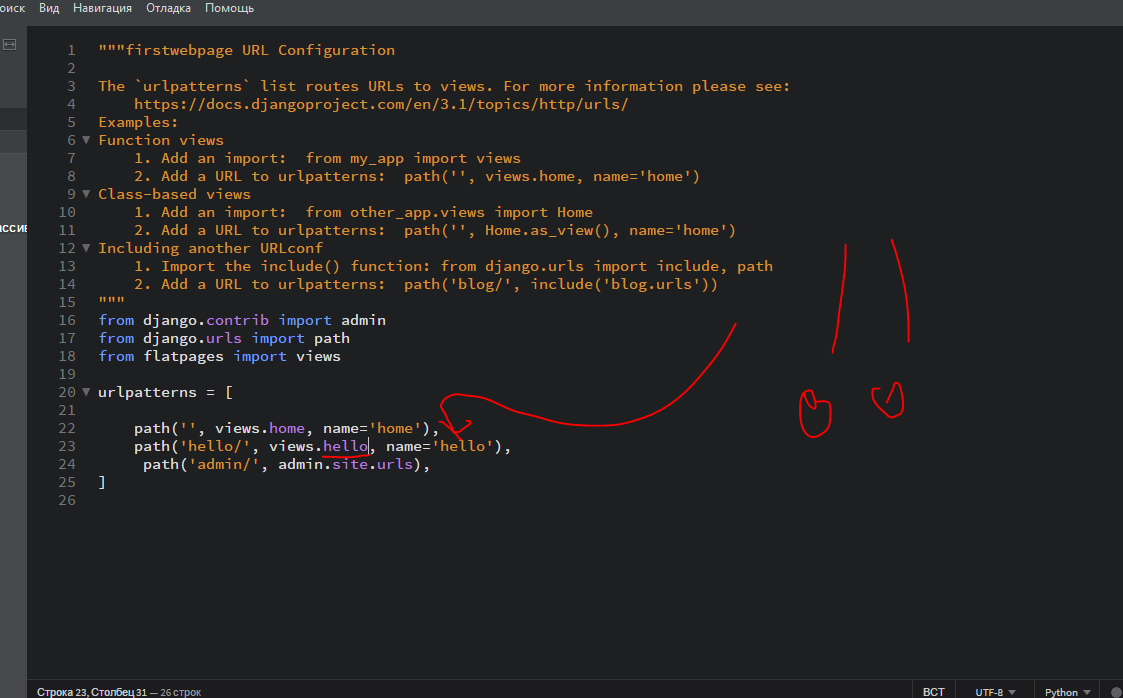
DEFAULT\_CHARSET = 'cp1251'

**Задание:**

* Сделайте так, чтобы по адресу <http://127.0.0.1:8000/hello/> возвращался тот же самый текст;

Home пустой из-за того, что это изначальный документ,

А hello имеет путь



(Тут накосячил urls.py. Имя страницы, оно разное)

* Уберите указание типа возвращаемого ответа (если классу HttpResponse напрямую не указать тип ответа, то будет выставлено значение по умолчанию). Сравните полученные результаты.

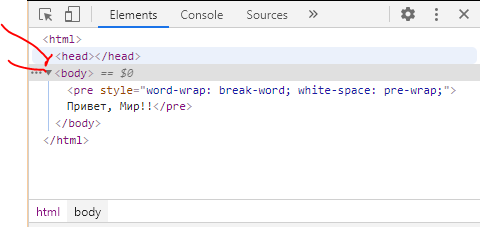
В файле views.py:

return HttpResponse(u'Привет, Мир!!')#без указания ответа

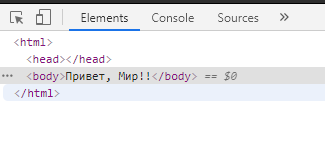
return HttpResponse(u'Привет, Мир!!', content\_type="text/plain")# указания ответа

*Работа с шаблонами в Django*

*Изменится код генерации элемента*



*У*казания типа ответа



*Без* указания типа ответа

Т.е. изменится стиль элемента в одном случае он есть, в другом его нет.

Шаблон в Django представляет собой строку текста, предназначенную для отделения представления документа от его данных. В шаблоне могут встречаться маркеры и простые логические конструкции (теги), управляющие отображением документа. Обычно шаблоны применяются для создания HTML-разметки, но в Django они позволяют генерировать документы в любом текстовом формате. Шаблон – это основа будущей HTML-разметки, которая должна быть заполнена теми данными, которые будут переданы в шаблон. Данные, которые необходимо внести в шаблон, называются *контекстом*, а процесс, когда данные вносятся в шаблон, называется *рендерингом*.

Создайте папку **templates** в директории **flatpages**. Затем, в папке **templates** создайте файл **index.html** со следующим кодом:

<!DOCTYPE html>

<html>

<head>

<title>Привет, Мир!</title>

</head>

<body>

<h1>Привет, Мир!</h1>

<h2>Это учебный сайт, с его помощью будут изучены технологии

python/django, html/css.</h2>

<h3>Как видите, здесь используются заголовки различных

уровней.</h3>

<p>Здесь есть маркированный список:</p>

<h4>

<ul>

<li>Элемент 1;</li>

<li>элемент 2;</li>

<li>элемент 3;</li>

<li>последний элемент.</li>

</ul>

</h4>

<p>И нумерованный список:</p>

<h4>

<ol>

<li>Элемент 1;</li>

<li>элемент 2;</li>

<li>элемент 3;</li>

<li>последний элемент.</li>

</ol>

</h4>

<p>И даже таблица:</p>

<table style="border: none">

<thead>

<tr>

<th>Столбик 1</th>

<th>Столбик 2</th>

<th>Столбик 3</th>

<th>Столбик 4</th>

</tr>

</thead>

<tr>

<td>Строка 1 Столбец 1</td>

<td>Строка 1 Столбец 2</td>

<td>Строка 1 Столбец 3</td>

<td>Строка 1 Столбец 4</td>

</tr>

<tr>

<td>Строка 2 Столбец 1</td>

<td>Строка 2 Столбец 2</td>

<td>Строка 2 Столбец 3</td>

<td>Строка 2 Столбец 4</td>

</tr>

<tr>

<td>Строка 3 Столбец 1</td>

<td>Строка 3 Столбец 2</td>

<td>Строка 3 Столбец 3</td>

<td>Строка 3 Столбец 4</td>

</tr>

<tr>

<td>Строка 4 Столбец 1</td>

<td>Строка 4 Столбец 2</td>

<td>Строка 4 Столбец 3</td>

<td>Строка 4 Столбец 4</td>

</tr>

<tr>

<td>Строка 5 Столбец 1</td>

<td>Строка 5 Столбец 2</td>

<td>Строка 5 Столбец 3</td>

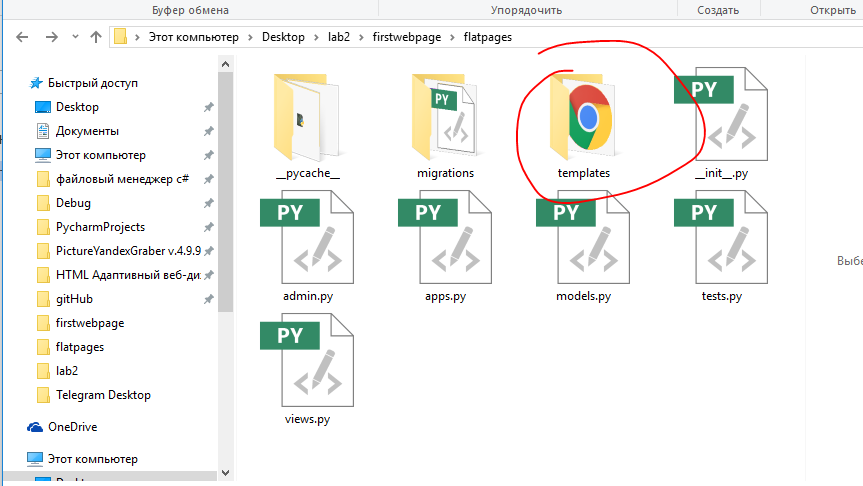
<td>Строка 5 Столбец 4</td>

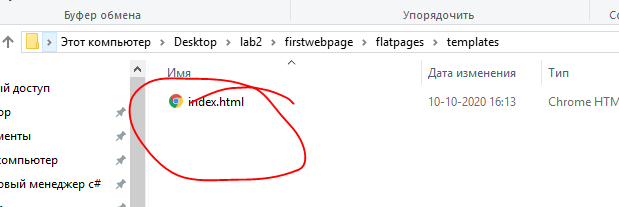
</tr>

</table>

</body>

</html>





Теперь созданный html-файл необходимо подключить к функции-представлению home, чтобы вместо простого текстового ответа приходил html-документ. Для этого в файле views.py, который находится в директории flatpages, добавьте операции импортирования:

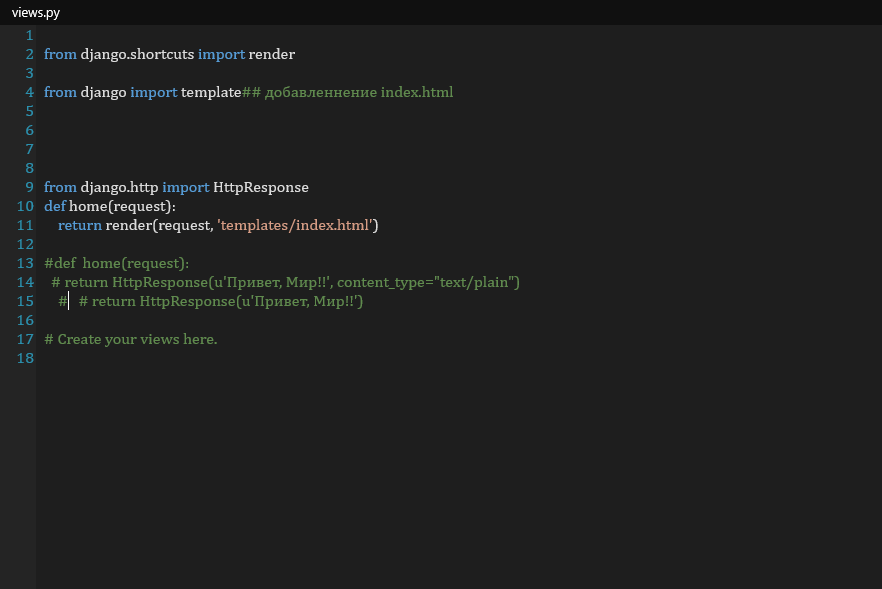
from django.shortcuts import render

from django import template

А функцию-представление home изменим следующим образом:

def home(request):

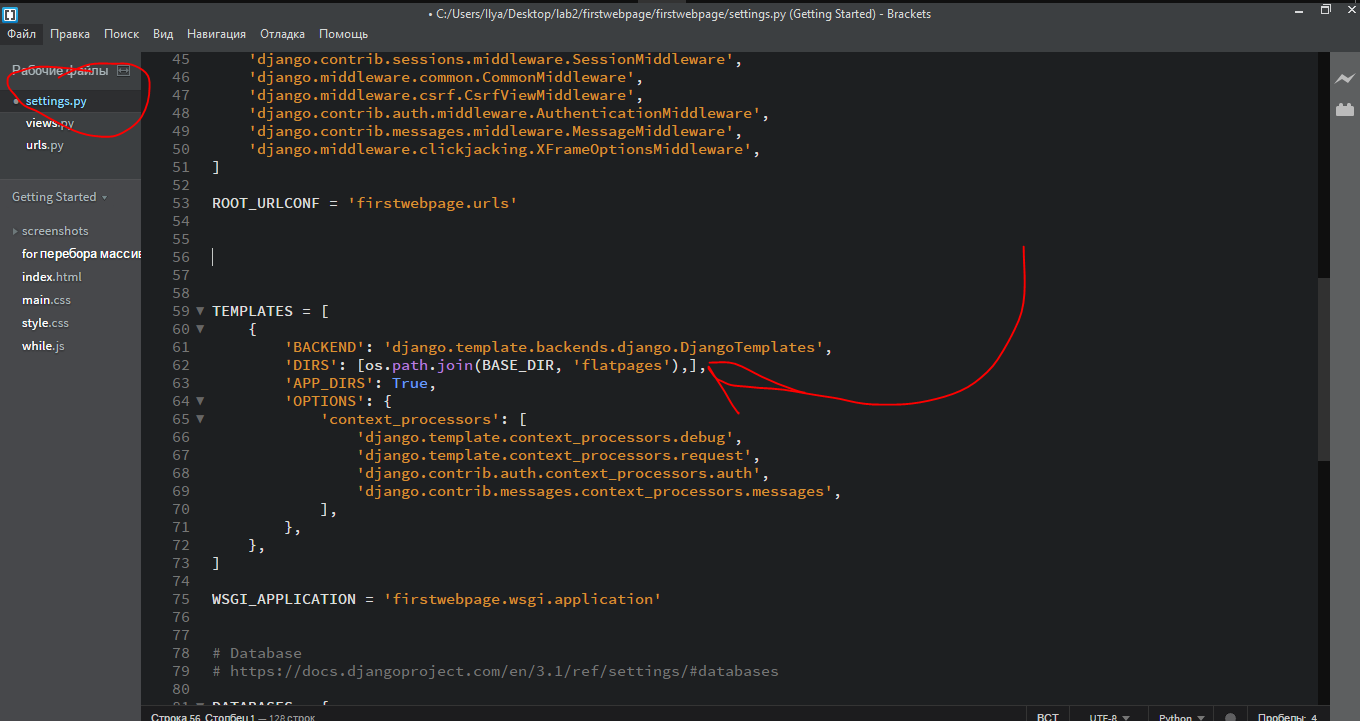
return render(request, 'templates/index.html')

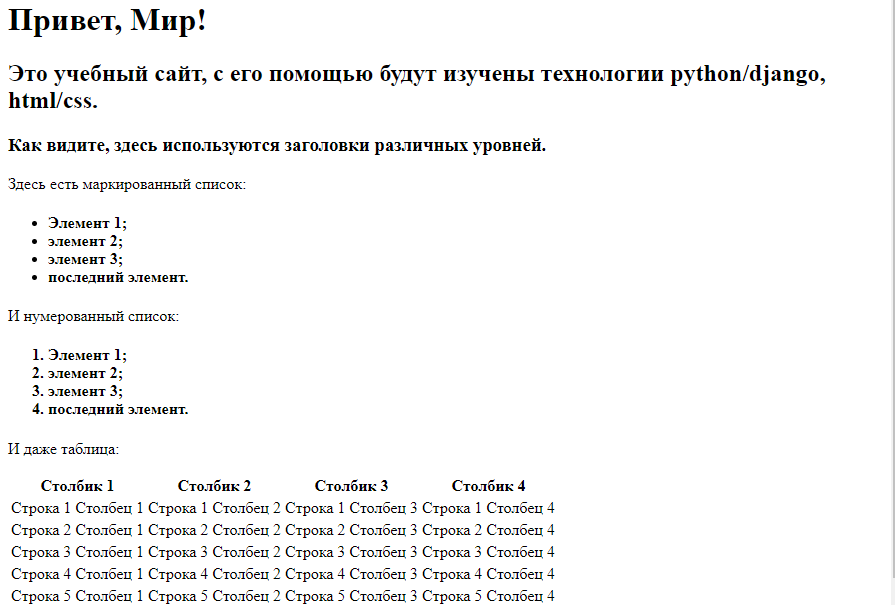


Также, для того, чтобы файл **index.html** был найден в директории **templates**, необходимо в файле **settings.py** изменить поле **DIRS** кортеже **TEMPLATES**. Поле **DIRS** должно содержать адрес директории, в которой располагается файл **index.html**, например:

'DIRS': [os.path.join(BASE\_DIR, 'flatpages'),],

Представление и шаблон готовы. Для того, чтобы посмотреть изменения, перезагрузите страницу с адресом <http://127.0.0.1:8000/>, если сервер запущен.





*Пример:*

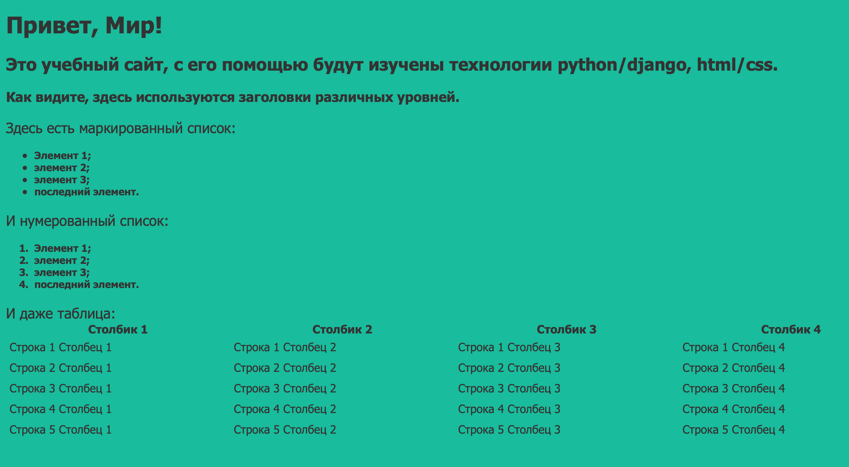
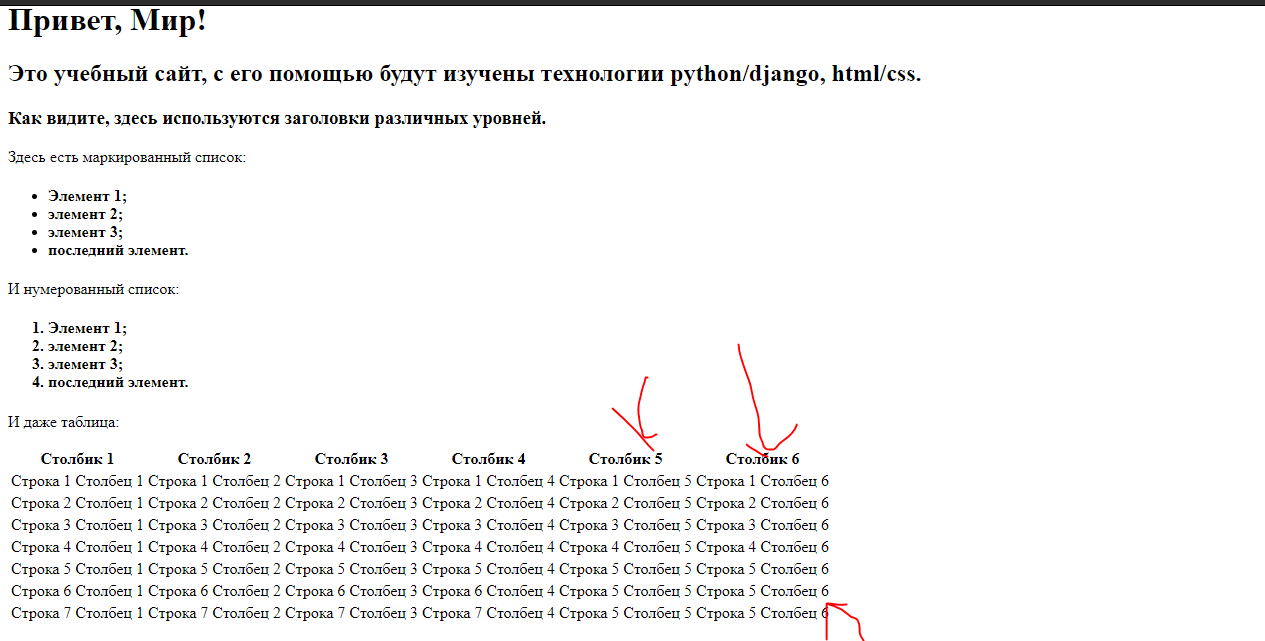
**

Рисунок 3. Пример отображения страницы

**Задание:**

Добавьте к созданной таблице 2 строки и 2 столбца;

Стиль не подрубался, багануло( нужно:

в файле urls.py в конце пропишите

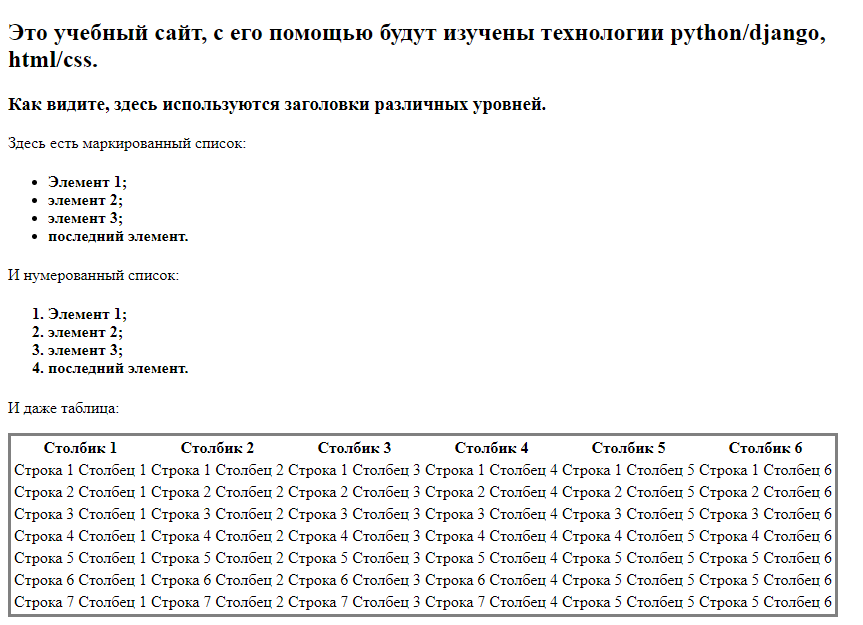
if settings.DEBUG:

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

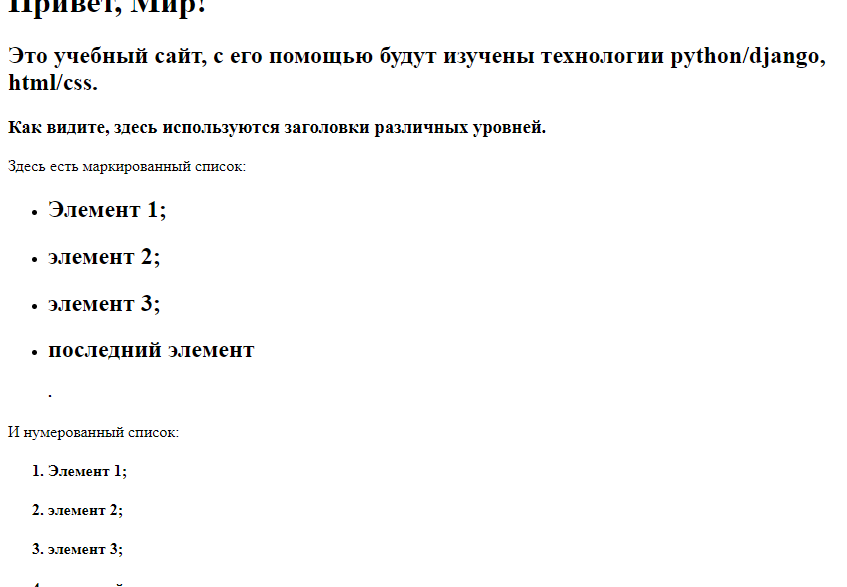
urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

потом удалил, после того как сервак написал что не может оторбразить страницу)

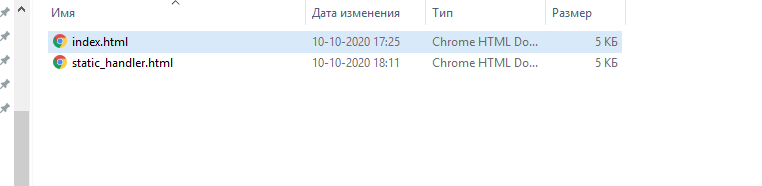
* Добавьте границы для таблицы;(сделал через id)



* Сделайте заголовки списков (нумерованного и маркированного) подзаголовками четвертого уровня;



* Создайте абсолютно идентичный шаблон, изменив только название на «static\_handler.html». В следующих заданиях при выполнении этой лабораторной работы изменяйте именно этот файл.



*Настройка обработки статичных файлов для Django*

Django-разработчики в основном работают с динамической частью приложения – представлениями и шаблонами, которые чаще всего изменяют свое содержимое при каждом запросе (например, страница профиля /profile/ будет у каждого пользователя разная, хотя каркас у всех будет одинаковым). Но web-приложения содержат и другую часть: статические файлы (изображения, CSS, Javascript и др.), которые не требуют никакой программной обработки. Для них нет потребности в рендеринге, они не зависят от содержимого базы данных. При каждом запросе к такому веб-серверу достаточно просто вернуть их прямо такими, какими их сохранили в последний раз.

Для того, чтобы придать документу стиль в соответствии с макетом, будем использовать CSS (формальный язык описания внешнего вида документа, написанного с использованием языка разметки). Создайте папку **static** в директории **flatpages**, а в ней файл **index.css** с кодом:

body {

background: #1abc9c;

font-family: Tahoma, Arial, sans-serif;

color: #333;

}

table {

border-collapse: collapse;

}

p, h4 {

font-size: 20px;

margin-bottom: 0;

}

h4 {

font-size: 14px;

}

ul, ol {

margin: 0;

}

table tr td {

padding: 5px;

}

table {

width: 100%;

}

img {

height: 30px;

width: auto;

}

После этого в файл страницы **static\_handler.html** во внутрь тега **<head>** вставьте тег подключения css-скрипта:

<link rel="stylesheet" href="{{ STATIC\_URL }}/static/index.css">

STATIC\_URL – переменная, объявленная в файле **settings.py**.

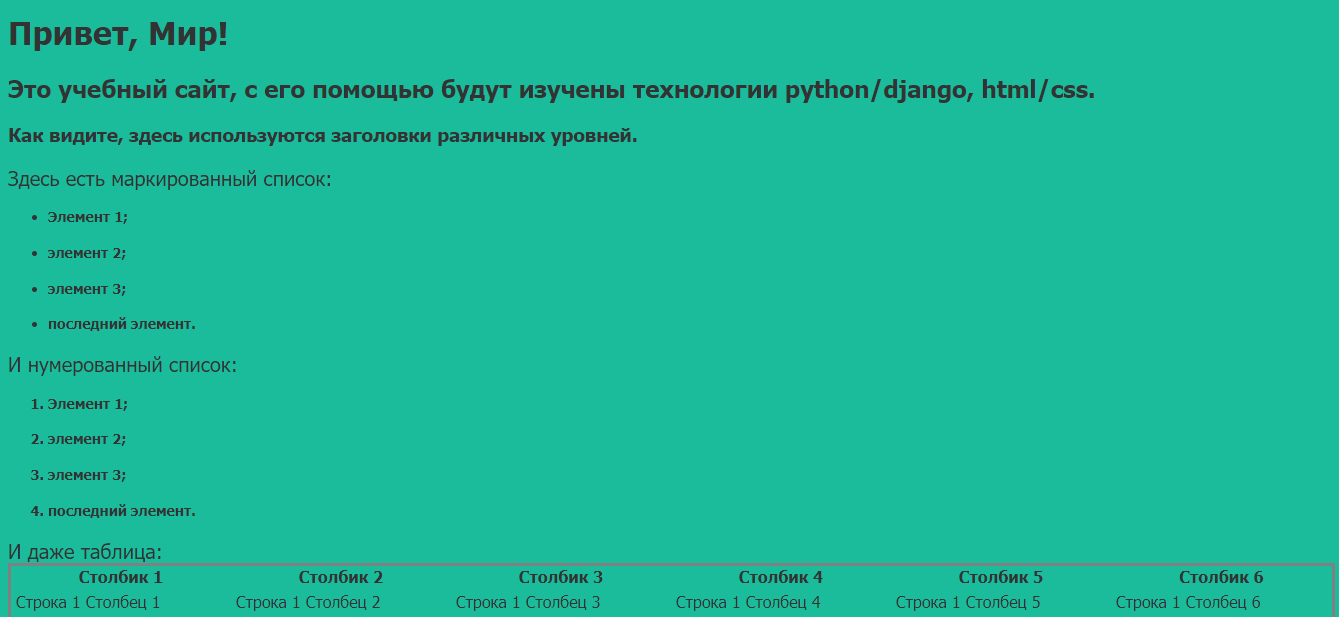
Свойство **background** задает фоновый цвет для элемента, в данном случае цвет задан в формате HEX (то есть три идущих подряд двузначных шестнадцатеричных числа).

Свойство **font-family** задает семейство шрифтов, которые будут применены к элементам.

Свойство **color** задает цвет шрифта.

Свойства border-collapse, font-size, margin и padding и дальнейшие применяемые свойства остаются на самостоятельное изучение.

Когда перед фигурными скобками указано несколько элементов через запятую, значит стиль применится ко всем перечисленным элементам.



**Задание:**

* Установите для заголовка первого уровня шрифт с засечками;

h1{font-family: "Times New Roman", Times, serif; }

* Добавьте картинку и сделайте ее высотой 30px(я сделал 300px);

<img src="{{STATIC\_URL}}/static/images/kat.png" alt="котик" id="image\_login">

* Измените размер шрифта для подзаголовков четвертого уровня;
* Сделайте ширину таблицы на 100% экрана; #image\_login{

height: 350px;}

* Загрузите ваш проект на любой гит-репозиторий (GitHub, GitLab, Google Code, Bitbucket и т.п.).