

Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации
Ордена Трудового Красного Знамени
федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра МКиИТ
Проектирование клиент-серверных приложений

Лабораторная работа №2
“Создание web-страницы с текстом”

Выполнил:
студент 3 курса,
группы БФИ2001
Лушин Е. А.

Москва 2023

Цель работы: научиться создавать web-страницы с простым текстом и html-шаблоны, настраивать обработку статичных файлов для Django.

Задание:

Создание web-страницы с текстом

- Сделать так, чтобы по адресу <http://127.0.0.1:8000/hello/> возвращался тот же самый текст;
- Убрать указание типа возвращаемого ответа (если классу HttpResponse напрямую не указать тип ответа, то будет выставлено значение по умолчанию). Сравнить полученные результаты.

Работа с шаблонами в Django

- Добавить к созданной таблице две строки и один столбец;
- Убрать у созданной таблицы все границы;
- Сделать заголовки списков (нумерованного и маркированного) подзаголовками четвертого уровня;
- Создать абсолютно такой же шаблон, только поменяв название на «static_handler.html». В следующих заданиях при выполнении этой лабораторной работы изменяйте именно новосозданный шаблон.

Настройка обработки статичных файлов для Django

- Установить для заголовка первого уровня шрифт с засечками;
- Добавить картинку высотой 30px;
- Изменить размер шрифта для подзаголовков четвертого уровня с 22px до 14px;
- Сделайте ширину таблицы на 100% экрана;

Краткая теория

Работа с шаблонами в Django

Как уже упоминалось ранее, шаблон в Django представляет собой строку текста, предназначенную для отделения представления документа от его данных. В шаблоне могут встречаться маркеры и простые логические конструкции (шаблонные теги), управляющие отображением документа.

Обычно шаблоны применяются для создания HTML-разметки, но в Django они позволяют генерировать документы в любом текстовом формате.

В каком-то смысле шаблоны в Django — это некий аналог бланков из реального мира. Приходя в любое государственное учреждение, вы можете получить полупустую заготовку какого-либо документа, к примеру заявления, которое вы заполняете лишь отчасти, потому что общие части уже пропечатаны за вас. То есть шаблон — это основа будущей HTML-разметки, которая должна быть заполнена теми данными, что будут переданы в шаблон.

Эти данные, которые необходимо внести в шаблон, в сумме называются контекстом, а процесс, когда данные вносятся в шаблон, называется рендерингом.

Пример шаблона для Django:

```
from django.template import Template

template = Template("<h1>Привет, моё имя {{name}}!</h1>")
```

и если вы пожелаете на место переменной name поставить, например, имя «Василий», то контекстом будет обычный словарь из языка Python, оформленный в специальный класс Context примерно такого содержания:

```
from django.template import Context

context = Context({"name": u"Василий"})
```

Теперь осталось только совместить шаблон и контекст с помощью рендеринга. Рендеринг шаблонов в Django:

```
html page = template.render(context)

# html page == "<h1>Привет, моё имя Василий!</h1>"
```

Помимо прямого вывода переменной в будущий HTML-документ в шаблонах можно пользоваться некими аналогами конструкций из языка Python. Например, в шаблонах можно использовать циклы for и условия if.

Пример использования алгоритмических конструкций в шаблонах:

```
from django.template import Context, Template

template = Template("""

<h1>Привет, моё имя {{ my name }}!</h1>


```

```

        {% for friend in friend_list %}
            {% if friend.is_groupmate %}
                <p>Мой одноклассник: {{ friend.name }}</p>
            {% endif %}
        {% endfor %}
    """
context = Context({"my name": u"Василий",
                  "friend_list": [
                      {"name": u"Георгий", "is_groupmate": False},
                      {"name": u"Леонид", "is_groupmate": True},
                      {"name": u"Константин", "is_groupmate": True},
                      {"name": u"Гавриил", "is_groupmate": False}, ] })
html_page = template.render(context)
# в итоге получится следующий html-код:
# <h1>Привет, моё имя Василий!</h1>
# <p>Мой одноклассник: Леонид</p>
# <p>Мой одноклассник: Константин</p>

```

В данном примере прямо внутри шаблона был произведен обход по всем элементам массива `friend_list` и для каждого из друзей была произведена проверка, является ли друг ещё и одноклассником, и если вся проверка успешно прошла, имя друга выводится в списке ниже.

Естественно, зашивать HTML-код прямо в представления на языке Python не очень хорошая штука (как и любое другое смешивание языков), поэтому хорошей, точнее сказать, обязательной, практикой является вынос шаблона в отдельный файл с форматом `.html`, и файл этот должен храниться в папке `templates` любого вашего приложения. Так что файловая структура ваших приложений в Django-проектах станет ещё чуточку сложнее. Теперь пример базового приложения стал шире.

Файловая структура приложения с шаблонами:

```
your app/
```

```
templates/  
    # здесь будут храниться ваши шаблоны  
    __init__.py  
models.py  
tests.py  
views.py
```

Также вам обязательно нужно знать, что создавать вручную объекты Template, затем объекты Context, а потом вызывать метод render первого, передавая туда контекст, необязательно. Естественно, как и все уважающие себя программисты, создатели Django создали автоматизированный вариант этого кода, который умещается всего. . . в одну строку кода, куда уж короче! Для этого была создана функция render, которая находится в модуле django.shortcuts. Пользоваться этой функцией очень просто.

Пример использования функции render():

```
from django.shortcuts import render  
  
def some_view(request):  
    return render(request,  
                   'some_template.html',  
                   {  
                       'context key1': 'context value1',  
                       'context key2': 'context value2',  
                   })
```

Здесь функции render в качестве первого аргумента передается объект запроса, который должен быть в каждом представлении, вторым аргументом идет имя шаблона, который сохранен в одной из папок templates ваших приложений, а третий аргумент — не что иное, как словарь, содержащий контекст вашего шаблона.

Работа со статичными файлами в Django-приложениях

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 для всех будет общим). Но веб-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 файлу веб-серверу достаточно просто вернуть их прямо такими, какими их сохранили в последний раз.

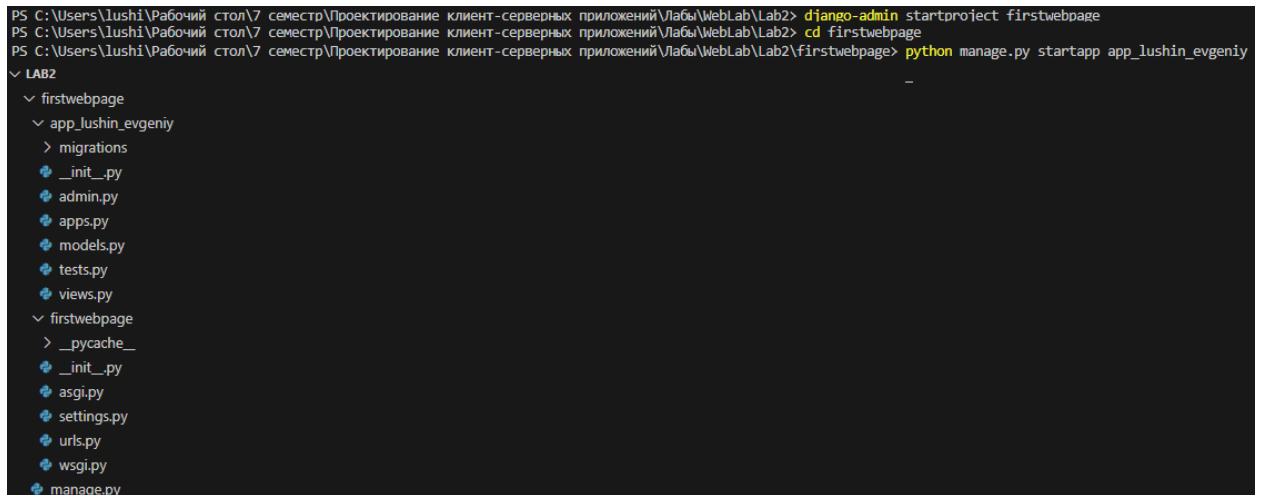
В больших проектах — особенно состоящих из десятков, а то и сотен приложений — работа с большим количеством файлов становится нелегким делом, потому что статические файлы расположены в разных папках. При базовой настройке ваши статичные файлы должны храниться в папке `static` каждого вашего приложения, и получается, что сколько приложений в вашем проекте, столько папок со статичными файлами нужно обрабатывать.

Для этого было создано приложение `django.contrib.staticfiles`: оно собирает статические файлы из всех ваших приложений (и остальных мест, которые вы укажете) в одном месте, что позволяет легко настроить выдачу статических файлов на реальном сервере.

Как уже было указано, в число статичных файлов обычно входят картинки всевозможных форматов (только если при каждом запросе вам нет нужды в дополнительной обработке изображения, однако такая потребность из разряда экзотических), сценарии на языке JavaScript и каскадные таблицы стилей (CSS, Cascading Style Sheets). О последних ниже будет краткое описание.

Выполнение

Для выполнения данной лабораторной работы создадим директорию Lab2, а в ней, создадим новый проект firstwebpage. Далее, перейдём в директорию firstwebpage и создадим в мой проект новое приложение под именем “app_lushin_evgeniy”. Все выше описанные действия продемонстрированы на рисунке 1.



```
PS C:\Users\lushi\Рабочий стол\7 семестр\Проектирование клиент-серверных приложений\Лабы\WebLab\Lab2> django-admin startproject firstwebpage
PS C:\Users\lushi\Рабочий стол\7 семестр\Проектирование клиент-серверных приложений\Лабы\WebLab\Lab2> cd firstwebpage
PS C:\Users\lushi\Рабочий стол\7 семестр\Проектирование клиент-серверных приложений\Лабы\WebLab\Lab2\firstwebpage> python manage.py startapp app_lushin_evgeniy
```

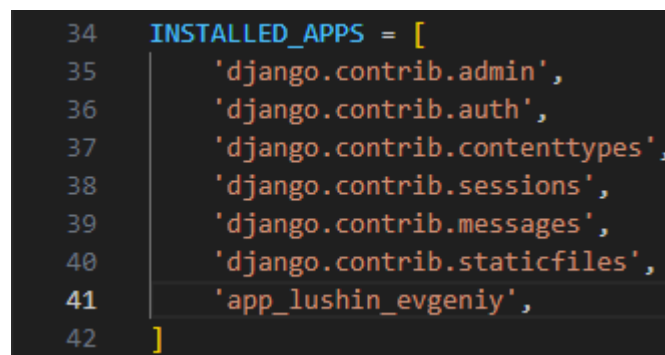
File Explorer view:

- LAB2
 - firstwebpage
 - app_lushin_evgeniy
 - migrations
 - __init__.py
 - admin.py
 - apps.py
 - models.py
 - tests.py
 - views.py
 - firstwebpage
 - __pycache__
 - __init__.py
 - asgi.py
 - settings.py
 - urls.py
 - wsgi.py
 - manage.py

Рисунок 1 – Создание проекта и приложения

В начале работы над проектом необходимо задать базовые настройки: задать имя базы данных и создать ее таблицы, как это было сделано в лабораторной работе №1, а также добавить нужные приложения в проект.

Далее, чтобы добавить приложение app_lushin_evgeniy в проект, откроем файл settings.py, найдём кортеж INSTALLED_APPS и добавим в конец элемента строку 'app_lushin_evgeniy', как показано на рисунке 2.



```
34 INSTALLED_APPS = [
35     'django.contrib.admin',
36     'django.contrib.auth',
37     'django.contrib.contenttypes',
38     'django.contrib.sessions',
39     'django.contrib.messages',
40     'django.contrib.staticfiles',
41     'app_lushin_evgeniy',
42 ]
```

Рисунок 2 – Файл “settings.py”

Перед началом работы с файлом `urls.py` необходимо импортировать `views` нашего приложения. А также для будущей страницы создадим новый адрес в этом же файле.

Листинг 1. Содержимое файла `urls.py`

```
firstwebpage > firstwebpage > 📄 urls.py > ...
1  from app_lushin_evgeniy import views
2
3  """
4  URL configuration for firstwebpage project.
5
6  The `urlpatterns` list routes URLs to views. For more information please see:
7  |   https://docs.djangoproject.com/en/4.2/topics/http/urls/
8  |   Examples:
9  |   Function views
10 |       1. Add an import:  from my_app import views
11 |       2. Add a URL to urlpatterns:  path('', views.home, name='home')
12 |   Class-based views
13 |       1. Add an import:  from other_app.views import Home
14 |       2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
15 |   Including another URLconf
16 |       1. Import the include() function: from django.urls import include, path
17 |       2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
18 |   """
19  from django.contrib import admin
20  from django.urls import path
21
22  urlpatterns = [
23      path('admin/', admin.site.urls),
24      path('', views.home, name='home'),
25  ]
```

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

Листинг 2. Содержимое файла `views.py`

```
firstwebpage > app_lushin_evgeniy > 📄 views.py > ...
1  from django.shortcuts import render
2  from django.http import HttpResponse
3
4  def home(request):
5      return HttpResponse(u'Привет, меня зовут Евгений и это мой вебсайт!', content_type="text/plain")
6
7  # Create your views here.
```

Для запуска сервера выполним команду, которая запускает локальный сервер на порту 8000. Результат выполнения команды показан на рисунке 3.


```

PS C:\Users\lushi\Рабочий стол\7 семестр\Проектирование клиент-серверных приложений\Лабы\WebLab\Lab2\firstwebpage> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
September 12, 2023 - 16:11:03
Django version 4.2.5, using settings 'firstwebpage.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

```

Рисунок 3 – Запуск сервера

После запуска сервера перейдём по адресу (<http://127.0.0.1:8000/>). Начальная страница продемонстрирована на рисунке 4.

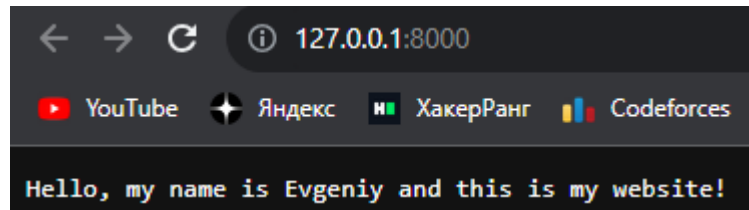


Рисунок 4 – Начальная страница вебсайта

Чтобы по адресу (<http://127.0.0.1:8000/hello/>) возвращался текст “Hello world!” необходимо в файл `urls.py` добавить следующую строку: `path('hello/', views.home, name='home')`. На рисунке 5 продемонстрирован вывод текста на сайте.

Листинг 3. Содержимое файла `urls.py`

```

firstwebpage > firstwebpage > urls.py > ...
1  from app_lushin_evgeniy import views
2  """
3  URL configuration for firstwebpage project.
4
5  The `urlpatterns` list routes URLs to views. For more information please see:
6  |   https://docs.djangoproject.com/en/4.2/topics/http/urls/
7  |   Examples:
8  |   Function views
9  |       1. Add an import: from my_app import views
10 |       2. Add a URL to urlpatterns: path('', views.home, name='home')
11 |   Class-based views
12 |       1. Add an import: from other_app.views import Home
13 |       2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
14 |   Including another URLconf
15 |       1. Import the include() function: from django.urls import include, path
16 |       2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
17 |   """
18  from django.contrib import admin
19  from django.urls import path
20
21  urlpatterns = [
22      path('admin/', admin.site.urls),
23      path('', views.home, name='home'),
24      path('hello/', views.hello, name='hello'),
25  ]

```

Листинг 4. Содержимое файла views.py

```
firstwebpage > app_lushin_evgeniy > views.py > ...
1  from django.shortcuts import render
2  from django.http import HttpResponseRedirect
3
4  def home(request):
5      return HttpResponseRedirect(u'Hello, my name is Evgeniy and this is my website!', content_type="text/plain")
6
7  def hello(request):
8      return HttpResponseRedirect(u'Hello, World!', content_type="text/plain")
9
10 # Create your views here.
```

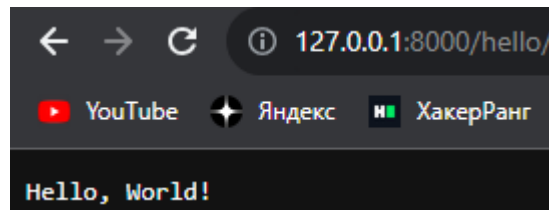


Рисунок 5 – Страница hello

Уберём тип возвращаемого ответа в функции hello.

Листинг 5. Содержимое файла views.py

```
firstwebpage > app_lushin_evgeniy > views.py > ...
1  from django.shortcuts import render
2  from django.http import HttpResponseRedirect
3
4  def home(request):
5      return HttpResponseRedirect(u'Hello, my name is Evgeniy and this is my website!', content_type="text/plain")
6
7  def hello(request):
8      return HttpResponseRedirect(u'Hello, World!')
9
10 # Create your views here.
```

Откроем страницу сайта (<http://127.0.0.1:8000/hello/>), после изменений можно заметить, что изменился шрифт нашего текста и фон сайта. Это продемонстрировано на рисунке 6.

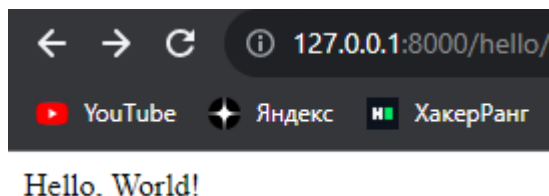


Рисунок 6 – Страница hello после изменения функции

Создадим папку templates в директории app_lushin_evgeniy, в данной папке создадим файл index.html. Структура проекта продемонстрирована на рисунке 7 на странице 11.

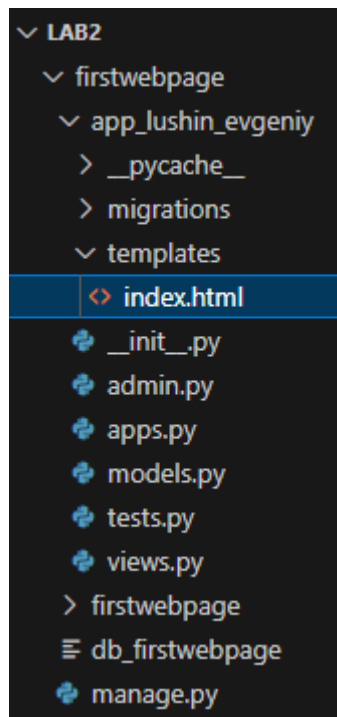


Рисунок 7 – Структура проекта Lab2

Код index.html представлен в листинге ниже.

Листинг 6. Содержимое файла index.html

```
firstwebpage > app_lushin_evgeniy > templates > <> index.html > ...
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5  |   <title>Привет, Мир!</title>
6  </head>
7
8  <body>
9  |   <h1>Привет, Мир!</h1>
10 |   <h2>Это учебный сайт, с его помощью будут изучены технологии
11 |   python/django, html/css.</h2>
12 |   <h3>Как видите, здесь используются заголовки различных
13 |   уровней.</h3>
14 |   <p>Здесь есть маркированный список:</p>
15 |   <ul>
16 |   |   <li>Элемент 1;</li>
17 |   |   <li>элемент 2;</li>
18 |   |   <li>элемент 3;</li>
19 |   |   <li>последний элемент.</li>
20 |   </ul>
21 |   <p>И нумерованный список:</p>
22 |   <ol>
23 |   |   <li>Элемент 1;</li>
24 |   |   <li>элемент 2;</li>
25 |   |   <li>элемент 3;</li>
26 |   |   <li>последний элемент.</li>
27 |   </ol>
```

```

28     <p>И даже таблица:</p>
29     <table border="1">
30         <thead>
31             <tr>
32                 <th>Столбик 1</th>
33                 <th>Столбик 2</th>
34                 <th>Столбик 3</th>
35             </tr>
36         </thead>
37         <tr>
38             <td>Строка 1 Столбец 1</td>
39             <td>Строка 1 Столбец 2</td>
40             <td>Строка 1 Столбец 3</td>
41         </tr>
42         <tr>
43             <td>Строка 2 Столбец 1</td>
44             <td>Строка 2 Столбец 2</td>
45             <td>Строка 2 Столбец 3</td>
46         </tr>
47         <tr>
48             <td>Строка 3 Столбец 1</td>
49             <td>Строка 3 Столбец 2</td>
50             <td>Строка 3 Столбец 3</td>
51         </tr>
52     </table>
53 </body>
54
55 </html>

```

Созданный html-файл необходимо подключить к функции-представлению home. Для этого в файле views.py, который находится в директории app_lushin_evgeniy, добавим операции импортирования и функцию-представление home.

Листинг 7. Содержимое файла views.py

```

firstwebpage > app_lushin_evgeniy > views.py > ...
1  from django.shortcuts import render
2  from django.http import HttpResponse
3
4  def home(request):
5      return render(request, 'templates/index.html')
6
7  def hello(request):
8      return HttpResponse(u'Hello, World!')
9
10 # Create your views here.

```

Также, для того чтобы файл index.html был найден в директории templates, в файле settings.py изменим поле DIRS в кортеже TEMPLATES.

Поле DIRS должно содержать адрес директории, в которой располагается файл index.html. На рисунке 8 продемонстрирован изменённый кусок кода.

```
56 TEMPLATES = [
57     {
58         'BACKEND': 'django.template.backends.django.DjangoTemplates',
59         'DIRS': ["C:\\Users\\lushi\\Рабочий стол\\7 семестр\\Проектирование клиент-серверных приложений\\Лабы\\WebLab\\Lab2\\firstwebpage\\app_lushin_evgeniy"],
60         'APP_DIRS': True,
61         'OPTIONS': {
62             'context_processors': [
63                 'django.template.context_processors.debug',
64                 'django.template.context_processors.request',
65                 'django.contrib.auth.context_processors.auth',
66                 'django.contrib.messages.context_processors.messages',
67             ],
68         },
69     ],
70 ]
```

Рисунок 8 – Изменённый кортеж TEMPLATES

Проверим теперь, произошли ли изменения на сайте. На рисунке 9 продемонстрирована новая начальная страница.

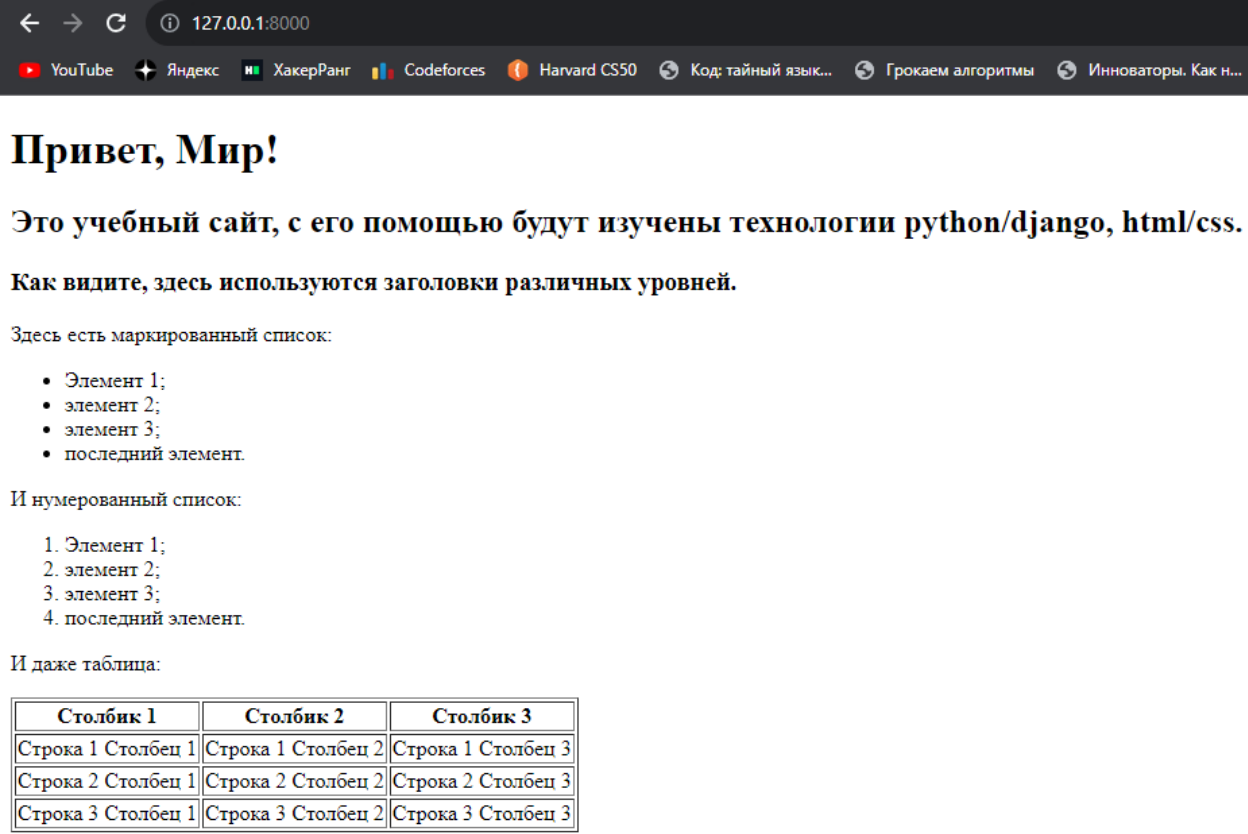


Рисунок 9 – Новая начальная страница

Создадим ещё 1 столбец и 2 строки к заданной таблице. Результат продемонстрирован на рисунке 10 на странице 15.

Листинг 8. Содержимое файла index.html

```
<!DOCTYPE html>
<html>

<head>
    <title>Привет, Мир!</title>
```

```
</head>

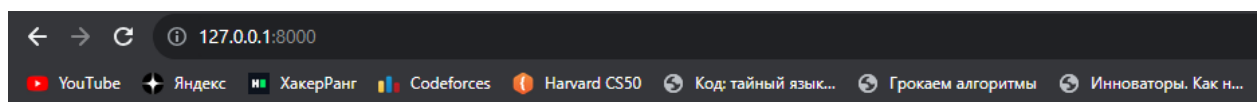
<body>
  <h1>Привет, Мир!</h1>
  <h2>Это учебный сайт, с его помощью будут изучены технологии
    python/django, html/css.</h2>
  <h3>Как видите, здесь используются заголовки различных
    уровней.</h3>
  <p>Здесь есть маркированный список:</p>
  <ul>
    <li>Элемент 1;</li>
    <li>элемент 2;</li>
    <li>элемент 3;</li>
    <li>последний элемент.</li>
  </ul>
  <p>И нумерованный список:</p>
  <ol>
    <li>Элемент 1;</li>
    <li>элемент 2;</li>
    <li>элемент 3;</li>
    <li>последний элемент.</li>
  </ol>
  <p>И даже таблица:</p>
  <table border="1">
    <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>
  </table>
```

```

        <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>

```



Привет, Мир!

Это учебный сайт, с его помощью будут изучены технологии python/django, html/css.

Как видите, здесь используются заголовки различных уровней.

Здесь есть маркированный список:

- Элемент 1;
- элемент 2;
- элемент 3;
- последний элемент.

И нумерованный список:

1. Элемент 1;
2. элемент 2;
3. элемент 3;
4. последний элемент.

И даже таблица:

Столбик 1	Столбик 2	Столбик 3	Столбик 4
Строка 1 Столбец 1	Строка 1 Столбец 2	Строка 1 Столбец 3	Строка 1 Столбец 4
Строка 2 Столбец 1	Строка 2 Столбец 2	Строка 2 Столбец 3	Строка 2 Столбец 4
Строка 3 Столбец 1	Строка 3 Столбец 2	Строка 3 Столбец 3	Строка 3 Столбец 4
Строка 4 Столбец 1	Строка 4 Столбец 2	Строка 4 Столбец 3	Строка 4 Столбец 4
Строка 5 Столбец 1	Строка 5 Столбец 2	Строка 5 Столбец 3	Строка 5 Столбец 4

Рисунок 10 – Страница с увеличенной таблицей

Уберём все границы у нашей таблицы. Результат продемонстрирован на рисунке 11 на странице 17.

Листинг 9. Содержимое файла index.html

```

<!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>
  <ul>
    <li>Элемент 1;</li>
    <li>элемент 2;</li>
    <li>элемент 3;</li>
    <li>последний элемент.</li>
  </ul>
  <p>И нумерованный список:</p>
  <ol>
    <li>Элемент 1;</li>
    <li>элемент 2;</li>
    <li>элемент 3;</li>
    <li>последний элемент.</li>
  </ol>
  <p>И даже таблица:</p>
  <table>
    <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>
  </table>
```

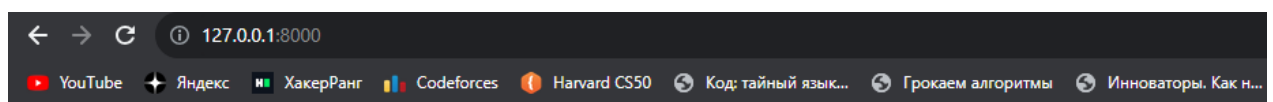


```

        </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>

```



Привет, Мир!

Это учебный сайт, с его помощью будут изучены технологии python/django, html/css.

Как видите, здесь используются заголовки различных уровней.

Здесь есть маркированный список:

- Элемент 1;
- элемент 2;
- элемент 3;
- последний элемент.

И нумерованный список:

1. Элемент 1;
2. элемент 2;
3. элемент 3;
4. последний элемент.

И даже таблица:

Столбик 1	Столбик 2	Столбик 3	Столбик 4
Строка 1 Столбец 1	Строка 1 Столбец 2	Строка 1 Столбец 3	Строка 1 Столбец 4
Строка 2 Столбец 1	Строка 2 Столбец 2	Строка 2 Столбец 3	Строка 2 Столбец 4
Строка 3 Столбец 1	Строка 3 Столбец 2	Строка 3 Столбец 3	Строка 3 Столбец 4
Строка 4 Столбец 1	Строка 4 Столбец 2	Строка 4 Столбец 3	Строка 4 Столбец 4
Строка 5 Столбец 1	Строка 5 Столбец 2	Строка 5 Столбец 3	Строка 5 Столбец 4

Рисунок 11 – Страница с таблицей без границ

Сделаем заголовки списков (нумерованного и маркированного)

подзаголовками четвертого уровня. Результат продемонстрирован на рисунке 12 на странице 19.

Листинг 10. Содержимое файла index.html

```
<!DOCTYPE html>
<html>

<head>
  <title>Привет, Мир!</title>
</head>

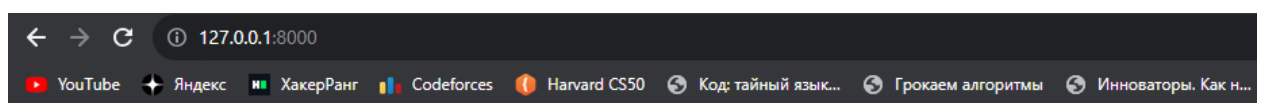
<body>
  <h1>Привет, Мир!</h1>
  <h2>Это учебный сайт, с его помощью будут изучены технологии
    python/django, html/css.</h2>
  <h3>Как видите, здесь используются заголовки различных
    уровней.</h3>
  <h4>Здесь есть маркированный список:</h4>
  <ul>
    <li>Элемент 1;</li>
    <li>элемент 2;</li>
    <li>элемент 3;</li>
    <li>последний элемент.</li>
  </ul>
  <h4>И нумерованный список:</h4>
  <ol>
    <li>Элемент 1;</li>
    <li>элемент 2;</li>
    <li>элемент 3;</li>
    <li>последний элемент.</li>
  </ol>
  <p>И даже таблица:</p>
  <table>
    <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>
  </table>
```

```

        <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>

```



Привет, Мир!

Это учебный сайт, с его помощью будут изучены технологии python/django, html/css.

Как видите, здесь используются заголовки различных уровней.

Здесь есть маркированный список:

- Элемент 1;
- элемент 2;
- элемент 3;
- последний элемент.

И нумерованный список:

1. Элемент 1;
2. элемент 2;
3. элемент 3;
4. последний элемент.

И даже таблица:

Столбик 1	Столбик 2	Столбик 3	Столбик 4
Строка 1 Столбец 1	Строка 1 Столбец 2	Строка 1 Столбец 3	Строка 1 Столбец 4
Строка 2 Столбец 1	Строка 2 Столбец 2	Строка 2 Столбец 3	Строка 2 Столбец 4
Строка 3 Столбец 1	Строка 3 Столбец 2	Строка 3 Столбец 3	Строка 3 Столбец 4
Строка 4 Столбец 1	Строка 4 Столбец 2	Строка 4 Столбец 3	Строка 4 Столбец 4
Строка 5 Столбец 1	Строка 5 Столбец 2	Строка 5 Столбец 3	Строка 5 Столбец 4

Рисунок 12 – Страница с новыми заголовками списков

Создадим идентичный шаблон с названием: **static_handler.html**. Для того, чтобы придать документу стиль в соответствии с макетом, будем использовать CSS. Создадим папку static в директории app_lushin_evgeniy, а в ней файл index.css. Структура проекта продемонстрирована на рисунке 13.

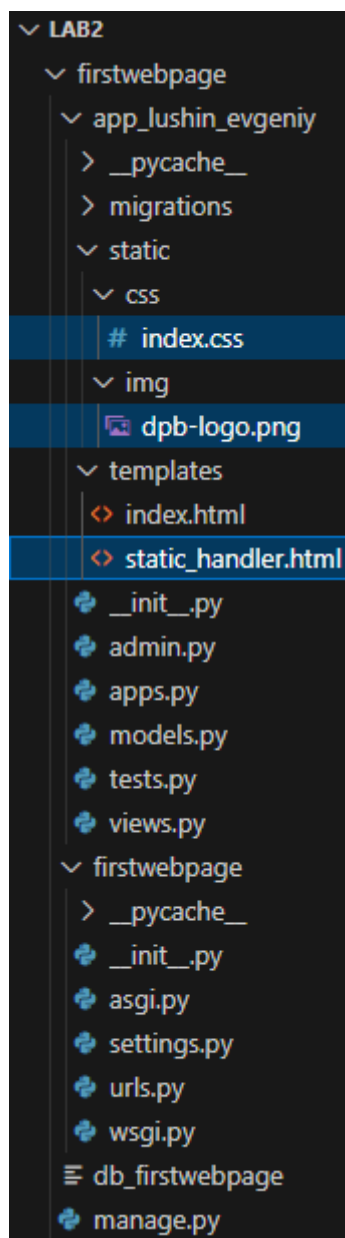


Рисунок 13 – Структура проекта Lab2

В файл страницы static_handler.html во внутрь тега <head> вставим тег подключения css-скрипта. Установим для заголовка первого уровня шрифт с засечками, добавим картинку и сделаем ее высотой 30px, изменим размер шрифта для подзаголовков четвертого уровня, сделаем ширину таблицы на 100% экрана. Результат продемонстрирован на рисунке 14 на странице 23.

Листинг 11. Содержимое файла static_handler.html

```
{% load static %}

<!DOCTYPE html>
<html>

<head>
    <link rel="stylesheet" type="text/css" href="{% static 'css/index.css' %}">
    <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>
    

    <h4>Здесь есть маркированный список:</h4>
    <ul>
        <li>Элемент 1;</li>
        <li>элемент 2;</li>
        <li>элемент 3;</li>
        <li>последний элемент.</li>
    </ul>
    <h4>И нумерованный список:</h4>
    <ol>
        <li>Элемент 1;</li>
        <li>элемент 2;</li>
        <li>элемент 3;</li>
        <li>последний элемент.</li>
    </ol>
    <p>И даже таблица:</p>
    <table>
        <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>
    </table>

```


```

        <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>

```

Листинг 12. Содержимое файла index.css

```

firstwebpage > app_lushin_evgeniy > static > css > # index.css >  img
1  body {
2      background: #1abc9c;
3      font-family: Tahoma, Arial, sans-serif;
4      color: #333;
5  }
6
7  table {
8      border-collapse: collapse;
9      width: 100%;
10 }
11
12 h1 {
13     font-family: "Times New Roman", Times, serif;
14     font-style: italic;
15 }
16
17 h4 {
18     font-size: 14px;
19 }
20
21 ul, ol {
22     margin: 0;
23 }
24
25 table tr td {
26     padding: 5px;
27 }
28
29 img {
30     height: 30px;
31 }

```

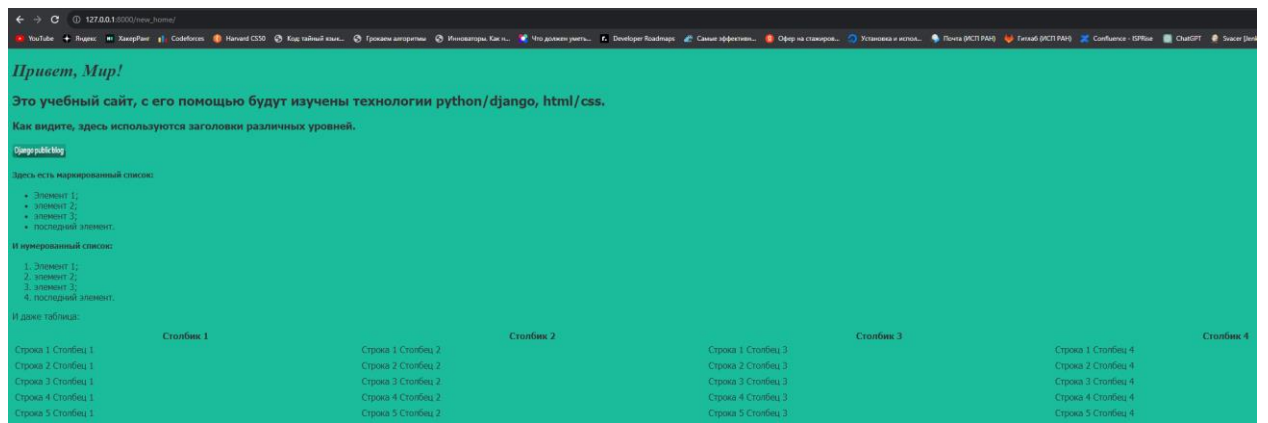


Рисунок 14 – Страница со стилем из шаблона

Вывод: В данной лабораторной работе я научился создавать web-страницы с простым текстом и html-шаблоны, а также настраивать обработку статичных файлов для Django.