

## Разработка программного обеспечения на языке Python

[Обзорная панель](#) ▶ [Мои курсы](#) ▶ [Разработка ПО на языке Python](#) ▶ [Веб-программирование на Python](#) ▶

[Лекция 3. Шаблонизация в django](#)

### Лекция 3. Шаблонизация в django

Посмотрите видеоуроки и ответьте на контрольные вопросы после лекции

#### Шаблоны в django



#### Передача данных в шаблоны

Передача значений - параметр `context`

00:00 / 04:06



В данной теме рассмотрим добавление в веб-приложение django пользовательского интерфейса и его настройку.

**Шаблоны (template)** отвечают за формирование внешнего вида приложения. Они предоставляют специальный синтаксис, который позволяет внедрять данные в код HTML.

Итак, в прошлых темах был создан пример проекта. Теперь добавим шаблоны. Для этого определим в папке приложения новый каталог **templates**. Теперь нам надо указать, что этот каталог будет использоваться в качестве хранилища шаблонов.

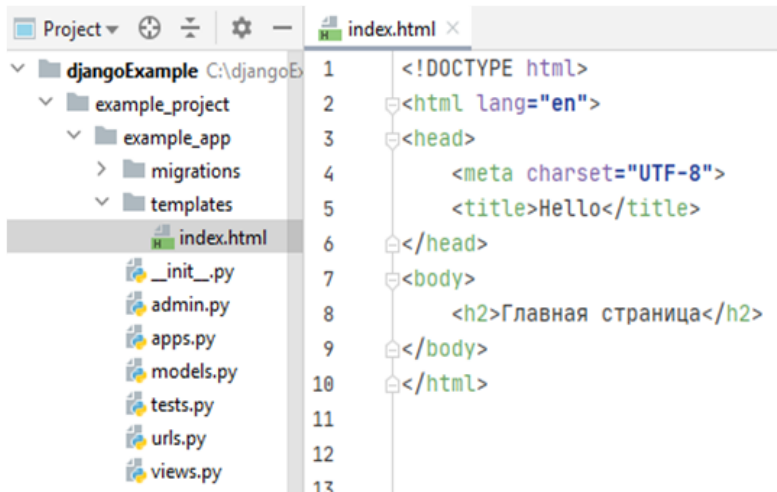
Для этого откроем файл **settings.py**. В этом файле настройка шаблонов производится с помощью переменной **TEMPLATES**. Параметр **DIRS** задает набор каталогов, которые хранят шаблоны. Но по умолчанию он пуст. Если создавать папку шаблонов в корневой папке проекта, то требуется в настройках задать расположения данной папки.

```

54
55 TEMPLATES = [
56     {
57         'BACKEND': 'django.template.backends.django.DjangoTemplates',
58         'DIRS': [],
59         'APP_DIRS': True,
60         'OPTIONS': {
61             'context_processors': [
62                 'django.template.context_processors.debug',
63                 'django.template.context_processors.request',
64                 'django.contrib.auth.context_processors.auth',
65                 'django.contrib.messages.context_processors.messages',
66             ],
67         },
68     },
69 ]

```

Затем в папке `templates` определим новый файл `index.html` со следующим html кодом.



```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Hello</title>
6 </head>
7 <body>
8     <h2>Главная страница</h2>
9 </body>
10 </html>
11
12
13

```

Теперь используем эту страницу для отправки ответа пользователю.

И для этого перейдем в приложении к файлу `views.py`, который определяет функции для обработки запроса. Изменим этот файл следующим образом:

```

1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4
5 def index(request):
6     return render(request, 'index.html')
7

```

Функция `index` вызывает функцию `render`, которой передаются объект запроса `request` и путь к файлу шаблона в рамках папки `templates` - `"index.html"`.

Одним из преимуществ шаблонов является то, что мы можем передать в них динамически из представлений различные данные. Для вывода данных в шаблоне могут использоваться двойная пара фигурных скобок.

Переменные в шаблонах

```
{{ название_объекта }}
```

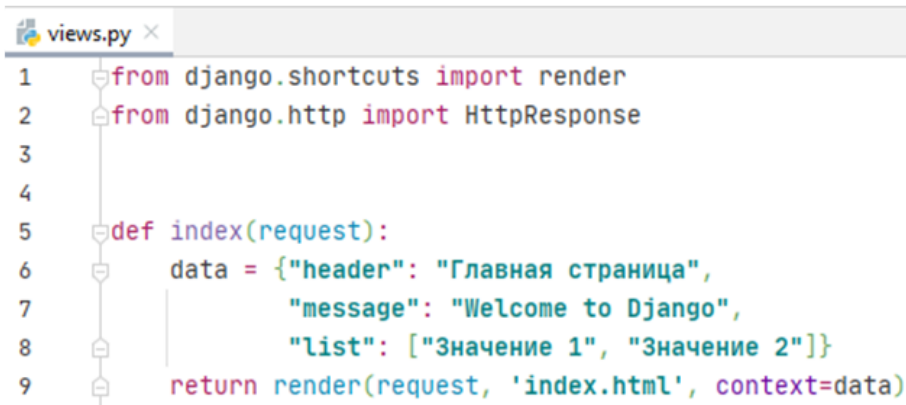
Например, пусть в проекте у нас есть папка `templates`, в которой содержится шаблон `index.html` со следующим кодом.

A screenshot of a code editor showing the content of index.html. The code is an HTML template with a DOCTYPE declaration, a head section with a meta charset and a title 'Hello', and a body section. In the body, there is an h2 tag with a Django template variable {{header}} and a message tag with a Django template variable {{message}}. These two variables are enclosed in a red rectangular box.

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Hello</title>
6 </head>
7 <body>
8     <h2>{{header}}</h2>
9     {{message}}
10 </body>
11 </html>
```

Здесь используется две переменных: `message` и `header`. Они будут передаваться из представления.

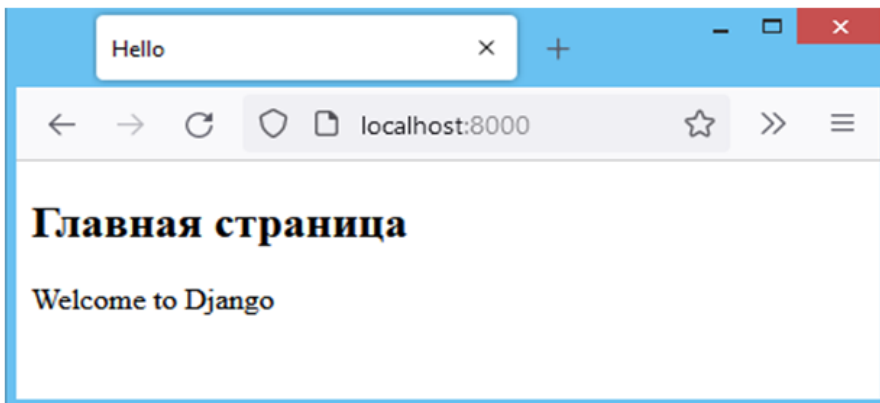
Чтобы из функции-представления передать данные в шаблон применяется третий параметр функции **render**, который называется контекст. Например, изменим файл `views.py` следующим образом:

A screenshot of a code editor showing the content of views.py. The code imports 'render' from 'django.shortcuts' and 'HttpResponse' from 'django.http'. It defines an 'index' function that takes a 'request' argument. Inside the function, a dictionary 'data' is created with keys 'header', 'message', and 'list'. The 'header' value is 'Главная страница', 'message' is 'Welcome to Django', and 'list' is a list containing 'Значение 1' and 'Значение 2'. The function returns the result of 'render(request, 'index.html', context=data)'.

```
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4
5 def index(request):
6     data = {"header": "Главная страница",
7            "message": "Welcome to Django",
8            "list": ["Значение 1", "Значение 2"]}
9     return render(request, 'index.html', context=data)
```

В шаблоне используются две переменных, соответственно словарь, который передается в функцию `render` через параметр `context`, содержит два значения с ключами `header` и `message`.

Вид страницы приложения таким образом будет следующий:



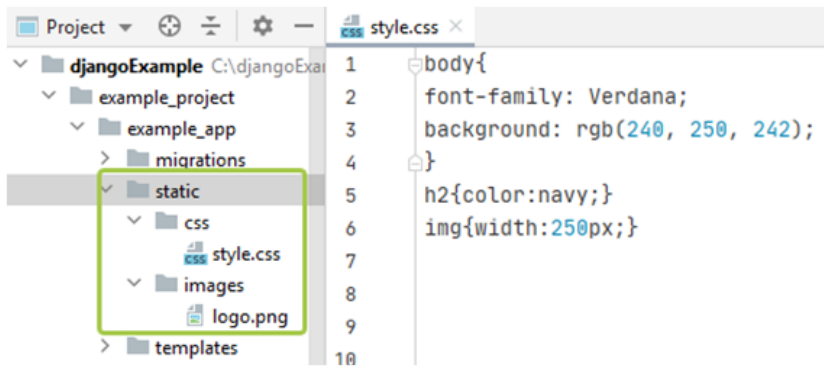
Веб-приложение, как правило, использует различные статические файлы - изображения, файлы стилей `css`, скриптов `javascript` и так далее.

Рассмотрим, как мы можем использовать подобные файлы. Добавим в папку приложения новый каталог **static**.

Определим для каждого типа файлов отдельные папки. В частности, создадим в папке `static` для изображений каталог **images**, а для стилей - каталог **css**. Подобным образом можно создавать папки и для других типов файлов.

В папку `static/images` добавим какое-нибудь изображение, а в папке `static/css` определим новый файл **styles.css**.

Определим в файле **styles.css** следующий код.



Теперь используем эти файлы в шаблоне. Для этого в начале файла шаблона необходимо определить инструкцию `load static`.

Для определения пути к статическим файлам используются выражения, в котором указывается путь расположения файла стилей.

Подключение стилей:

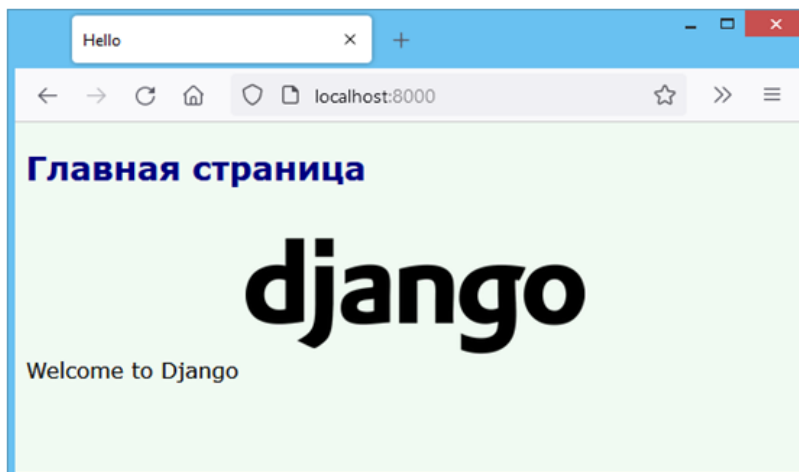
`{% load static %}`

`{% static "путь к файлу в папке static" %}`

Так, возьмем шаблон `index.html` из папки `templates` и изменим его следующим образом: добавим рисунок и подключим стили, описанные в статических файлах.



В файле `urls.py` в главном проекте пропишем сопоставление функции `index` с запросом к корню веб-приложения. И запустим проект на выполнение и перейдем к приложению в браузере.



Таким образом, мы можем передавать в шаблоны данные и возвращать пользователю динамически генерируемые веб-страницы.

Подведем итоги, были приведены сведения о шаблонах Django и показано, как можно создать шаблоны и передать в них различные данные из программы на Python.

Теги шаблонизатора в Django

ПРЕДЫДУЩИЙ ЭЛЕМЕНТ КУРСА

◀ [Задание 3. Создание своего первого сайта на Django](#)

Перейти на...

СЛЕДУЮЩИЙ ЭЛЕМЕНТ КУРСА

[Задание 4. Создание страниц веб-приложения](#) ▶

© 2010-2023 Центр обучающих систем  
Сибирского федерального университета, sfu-kras.ru

Разработано на платформе moodle  
Beta-version (3.9.1.5.w3)

[Политика конфиденциальности](#)

[Соглашение о Персональных данных](#)

[Политика допустимого использования](#)

**Контакты** +7(391) 206-27-05  
[info-ms@sfu-kras.ru](mailto:info-ms@sfu-kras.ru)

[Скачать мобильное приложение](#)

[Инструкции по работе в системе](#)