Разработка программного обеспечения на языке Python

Лекция 3. Шаблонизация в django

Посмотрите видеоуроки и ответьте на контрольные вопросы после лекции

Теги шаблонизатора в Django

00:00 / 03:49

В данной теме рассмотрим добавление кода в пользовательский интерфейс веб-приложения джанго элементов за счет встроенных инструментов шаблонизатора.

Django предоставляет возможность использовать в шаблонах ряд специальных тегов, которые упрощают вывод некоторых данных. Рассмотрим некоторые наиболее используемые теги. Переменные уже рассмотрели, они используются при передаче данных в шаблоны.

Нередко шаблоны должны иметь одинаковую базовую структуру, одни и те же блоки, при этом определять для отдельных блоков различное содержимое. Это позволяет сформировать единообразный стиль сайта, когда веб-страницы имеют одни и те же структурные элементы - меню, хедер, футер и так далее.

В этом случае оптимально повторно использовать один базовый шаблон для одинаковых элементов приложения, который определяет все основные блоки.

base.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Example project</title>
    <link rel="stylesheet" href="{% static 'css/style.css' %}" />
</head>
<body>
    <h2>{% block header %}{% endblock header %}</h2>
    <div>Панель навигации</div>
     <div>
        <a href="/">Главная</a>
        <a href="about">О сайте</a>
     </div>
    <div>{% block content%}{% endblock content %}</div>
</body>
</html>
```

Например, определим шаблон, который назовем **base.html**.

С помощью элементов {% block название_блока %}{% endblock название_блока %} определяются отдельные блоки шаблонов. При этом для каждого блока определяется открывающий и закрывающий элемент.

Например, блок content: Когда другие шаблоны будут применять данный шаблон, то они могут определить для блока content какое-то свое содержимое. Подобным образом здесь могут быть определены еще блоки. Самих блоков при необходимости можно определить сколько угодно.

Теперь применим этот базовый шаблон. Например, создадим новый шаблон **index.html**:

index.html

```
{% extends "base.html" %}
{% block header %}
Главная страница
{% endblock header %}

{% block content %}
<p>Информация о приложении</p>
{% endblock content %}
```

и файл about.html

about.html

```
{% extends "base.html" %}
{% block header %}
О сайте
{% endblock header %}

{% block content %}
<p>Тестовая информация</p>
{% endblock content %}
```

С помощью выражения {% extends "base.html" %} определяем, какой базовый шаблон будет расширяться. Затем определяется содержимое для блоков header, content. Стоит отметить, что необязательно указывать содержимое для всех блоков базового шаблона.

В итоге если нам потребуется изменить структуру всех веб-страниц сайта, добавить новые элементы или убрать старые, то достаточно будет изменить один базовый шаблон.

В итоге по шаблону index.html будет создана следующая веб-страница.

Далее рассмотрим теги, позволяющие добавить условия. Тег **{% if %} {% endif %}** позволяет выводить в шаблоне определенное содержимое в зависимости от некоторого условия. В качестве параметра тегу if передается выражение, которое должно возвращать True или False.

```
•{% if "Условие" %}
...
{% endif %}
```

```
•{% if "Условие" %}
...
{% elif "Условие 2" %}
...
{% else %}
...
{% endif %}
```

Тег **for** позволяет создавать циклы. Этот тег принимает в качестве параметра некоторую коллекцию и пробегается по этой коллекции, обрабатывая каждый ее элемент. Тег имеет следующую структуру: {% for "Индекс элемента" in "Коллекция элементов" %) {% endfor %}

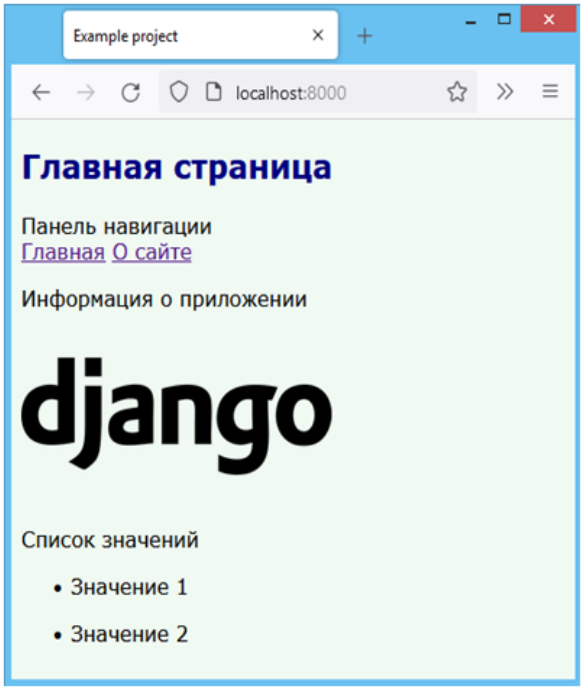
```
{% for "Индекс элемента" in "Коллекция
элементов" %}
...
{% endfor %}
```

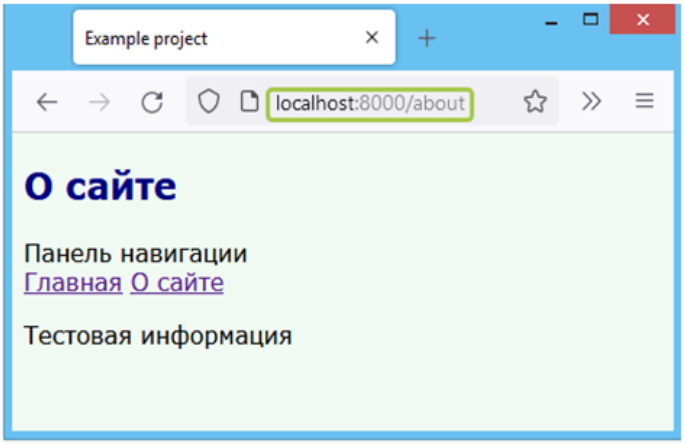
Например, пусть из представления view в шаблон передается некоторый массив. Выведем элементы массива в шаблоне с помощью тега for. Также зададим условие, что если переданное сообщение в переменную message соответствует определенному тексту, то на страницу будет загружен рисунок из папки статических файлов.

index.html

```
1  {% extends "base.html" %}
2  {% load static %}
3  {% block header %}
4  Главная страница
5  {% endblock header %}
6
7  {% block content %}
8  <p>Информация о приложении</p>                    Условие
9  {% if message == 'Welcome to Django' %}
10     <img src="{% static 'images/logo.png' %}" alt="django logo" >
11 {% endif %}
12
13 {% if list|length > 0 %}
14     <p>Список значений</p>
15     {% for element in list %}
16     <ul>
17         <li>{{element}}</li>
18     </ul>
19     {% endfor %}                                  Цикл
20 {% endif %}
21 {% endblock content %}
22
```

В результате мы получим следующую веб-страницу.





В данном разделе были приведены теоретические сведения о шаблонах Django и показано, как можно создать шаблоны и передать в шаблоны различные данные из программы на Python. Мы узнали, что через шаблоны можно сформировать и выдать пользователю ответ на его запрос. Однако динамичные веб-сайты должны выполнять и обратную функцию, - принимать от пользователя некоторые данные и сохранять их в БД. Данную тему будем рассматривать на следующей лекции.

Формы в Django

ПРЕДЫДУЩИЙ ЭЛЕМЕНТ КУРСА

◄ Задание 3. Создание своего первого сайта на Django

Перейти на...

СЛЕДУЮЩИЙ ЭЛЕМЕНТ КУРСА

Задание 4. Создание страниц веб-приложения ►