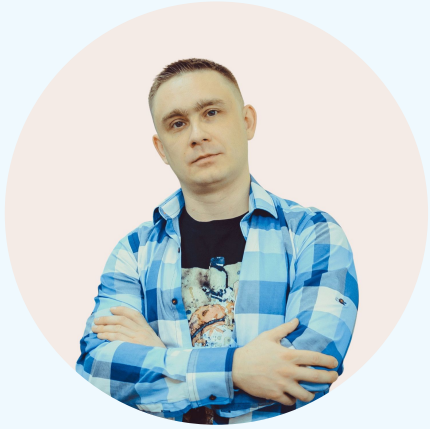
нетология

Динамическое формирование страниц на основе шаблонов

Александр
Бардин



Александр Бардин

Python-разработчик в Open Solutions

Александр Бардин

План занятия

1. Шаблоны в Django
2. Фильтры
3. Теги и фильтры
4. Наследование и композиция

ВСПОМИНАЕМ ПРОШЛЫЕ ЗАНЯТИЯ

— Что такое view-функция?

— Что она возвращает?

ВСПОМИНАЕМ ПРОШЛЫЕ ЗАНЯТИЯ

— Что такое HTML?

— Что нужно браузеру для того, чтобы отрендерить страницу?

VIEW ВОЗВРАЩАЮТ HTML

На примере сайта habr.com:

— страница профиля пользователя

— страница поста

— страница со списком постов

— страница создания поста

— страница компании

— и т.д.

КАК СОЗДАТЬ HTML

HTML генерится в результате рендеринга специальных шаблонов.

Конфигурация шаблонов задается через опцию TEMPLATE в конфиге:

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                ...
            ],
        },
    },
]
```

КАК СОЗДАТЬ HTML

Важно: шаблоны внутри одного приложения создаются в отдельном пространстве имен.

Если приложение называется `base` , то:

```
templates
|-- base
     |-- home.html
```

ШАБЛОНЫ В DJANGO

ШАБЛОНЫ

В Django поддерживаются две системы шаблонов:

стандартные шаблоны Django и Jinja2.

В рамках курса мы будем рассматривать первый вариант.

Документация:

https://docs.djangoproject.com/en/2.1/topics/templates/

ШАБЛОНЫ

Шаблоны — это расширенный HTML.

Подстановка значения переменной: {{ username }}

Через точку можно получить атрибут, ключ или вызвать метод:

```
{{ user.first_name }}
{{ name.strip }}
{{ some_dict.key }}
```

Методы вызываются без параметров.

Пример текста:

```
My first name is {{ first_name }}. My last name is {{ last_name }}
```

ФУНКЦИЯ RENDER

`render` — функция, позволяющая отрендерить шаблон с определенным «контекстом».

Контекст шаблона — это параметры, которые будут подставлены в HTML динамически.

Даже если вы не передаете явно никакие параметры, определенные глобальные объекты будут заданы через `context_processors` .
https://docs.djangoproject.com/en/2.1/topics/http/shortcuts/#render

ИСПОЛЬЗОВАНИЕ ФУНКЦИИ RENDER

```
def get(request):
    context = {
        'first_name': 'Jake',
        'last_name': 'The Dog'
    }
    return render(request, 'home.html', context)
```

Результат:

```
My first name is Jake. My last name is The Dog
```

ШАБЛОНЫ. ТЕГИ

Теги в шаблонах — это расширение для динамической генерации HTML-страниц.

Теги похожи на Python-код внутри HTML-файла:

```
{% for obj in some_list %}
   {% cycle 'row1' 'row2' as rowcolors silent %}
   <tr class="{{ rowcolors }}">{% include "subtemplate.html" %}</tr>
{% endfor %}
```

Теги указываются через `{% ... %}`

ШАБЛОНЫ. ТЕГИ

Полный список тегов можно найти в документации:
https://docs.djangoproject.com/en/2.1/ref/templates/builtins/#built-intemplate-tags-and-filters

Рассмотрим часть из них:

— `for`

— `if`

— `cycle`

Важно! Как и HTML-теги, теги шаблонов необходимо закрывать.

for позволяет итерироваться по объекту:

```
<ul>
{% for student in student_list %}
    <li>{{ student.rating }}</li>
{% endfor %}
</ul>
```

if позволяет проверять условие:

```
{% if user.is_authenticated %}
    <p>Здравствуйте, {{ user.username }}!</p>
{% endif %}
```

Внутри тегов можно использовать сложные условия:

```
{% if a == 'yes' or b == 'maybe' and e %}
...
{% endif %}
```

ФИЛЬТРЫ

ШАБЛОНЫ. ФИЛЬТРЫ

Фильтр — это специальное преобразование, которое применяется к переменной. Оно может быть достаточно сложным.

Фильтрам можно передавать параметры.

Примеры:

```
{{ name|lower }}
{{ created_at|date:"D d M Y" }}
{{ users|slice:":10" }}
```

Фильтры можно комбинировать:

```
{{ title|slugify|truncatechars:10 }}
```

ШАБЛОНЫ. ФИЛЬТРЫ

Фильтры можно использовать в условиях:

```
{% if users|length > 1 %}
...
{% endif %}
```

ШАБЛОНЫ. ФИЛЬТРЫ

Значения в шаблонах экранируются по умолчанию. Это сделано из соображений безопасности. Если вы хотите подставить HTML без экранирования, необходимо использовать фильтр `safe`:

```
{{ metrica_script|safe }}
```

ШАБЛОНЫ

Что делать, если стандартных тегов и фильтров недостаточно?

ШАБЛОНЫ

Шаблонизатор можно (и нужно!) расширять собственными фильтрами и тегами. Сделать это можно с помощью специального декоратора `register`

```
from datetime import datetime
from django import template


register = template.Library()


@register.filter
def clear_word(word):
    return word.replace('_', ' ')

@register.simple_tag
def current_time(format):
    return datetime.now().strftime(format)
```

https://docs.djangoproject.com/en/2.1/howto/custom-template-tags/

ПОЛЬЗОВАТЕЛЬСКИЕ ШАБЛОНЫ И ФИЛЬТРЫ

Пользовательские шаблоны и фильтры создаются внутри директории `templatetags`:

```
├── app
│   ├── templates
│   │   └── ...
│   ├── templatetags
│   │   ├── __init__.py
│   │   └── news_filters.py
│   ├── urls.py
│   └── views.py
```

Важно! Чтобы Django подхватил пользовательские фильтры и шаблоны, приложение должно быть подключено в настройках. После добавления модуля `templatetags` требуется перезапустить сервер.

ТЕГИ И ФИЛЬТРЫ

ТЕГИ И ФИЛЬТРЫ

Для чего стоит использовать теги, а для чего фильтры?

ТЕГИ И ФИЛЬТРЫ

Фильтры — там, где нужно преобразовывать значение к нужному виду.

Теги — там, где требуется более сложная логика с несколькими переменными или нужно получить еще неизвестное значение.

Вернемся к примеру с habr.com .

Что общего есть на страницах?

Что рекомендуется делать с повторяющимся кодом?

НАСЛЕДОВАНИЕ И КОМПОЗИЦИЯ

ШАБЛОНЫ. НАСЛЕДОВАНИЕ

Шаблонизатор Django позволяет создавать блоки, которые затем можно переопределять.

В этом примере создано два блока — `title` и `content` :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>{% block title %}Уроки{% endblock %}</title>
</head>

<body>
    <div id="content">
        {% block content %}Значение по умолчанию{% endblock %}
    </div>
</body>
</html>
```

ШАБЛОНЫ. НАСЛЕДОВАНИЕ

Блоки переопределяются для каждого отдельного view:

```
{% extends "base.html" %}

{% block title %}Уроки SMM{% endblock %}

{% block content %}Программа курса: ...{% endblock %}
```

В другом view можно указать:

```
{% extends "base.html" %}

{% block title %}Уроки SMM{% endblock %}

{% block content %}Программа курса: ...{% endblock content %}
```

ШАБЛОНЫ. НАСЛЕДОВАНИЕ

Можно переопределять не все блоки, а только те, которые нужны. В остальных блоках будет подставлено значение из родительского шаблона.

`extends` сообщает, что данный шаблон расширяет указанный базовый шаблон.

Блоки необходимо закрывать.

Блоки могут быть вложенными.

В закрывающемся теге endblock можно указывать имя блока. Это улучшает читаемость, что особенно полезно в больших шаблонах.
https://docs.djangoproject.com/en/2.1/ref/templates/language/#templateinheritance

ШАБЛОНЫ. КОМПОЗИЦИЯ

Помимо наследования можно также использовать композицию шаблонов. С помощью тега include в шаблон можно подключать другие шаблоны:

```
<div>
Страница пользователя {{user.username}}:
{% include "user_comments.html" %}
{% include "user_rating.html" %}
{% include "user_friends.html" %}
</div>
```

Во включенные шаблоны будет передан текущий контекст.

ШАБЛОНЫ. КОМПОЗИЦИЯ

При включении можно передавать дополнительные переменные:

```
{% include "user_bio.html" with first_name="Jack" last_name="The Dog" %}
```

Шаблон `user_bio.html` :

```
<div>Имя: {{ first_name }}, фамилия: {{ last_name }}</div>
```

ШАБЛОНЫ. КОМПОЗИЦИЯ

Документация по использованию include:
https://docs.djangoproject.com/en/2.1/ref/templates/builtins/#include

НАСЛЕДОВАНИЕ И КОМПОЗИЦИЯ

Что лучше — наследование или композиция?

Итоги

1. Шаблонизатор Django
2. Теги и фильтры в шаблонах
3. Наследование
4. Композиция

Теперь вы сможете создавать динамические страницы, которые легко поддерживать и менять.

Пример слайда с домашним заданием

Давайте посмотрим ваше домашнее задание.

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

нетология

Задавайте вопросы и пишите отзыв о лекции!

Александр Бардин

Александр Бардин