Шаблоны (Templates)

Лёзин Арсений, arseny.lezin@gmail.com

Использование шаблонов

- Формирование содержимого страницы
- Использование принципа DRY (Don't Repeat Yourself)
- Формирование любой текстовой информации на основе входных данных

Отображение страницы

- Код страницы обычно пишется на языке разметки HTML
- Использование динамических данных с бекенда
- Использование тегов для избавления от копипасты

HTML

Язык разметки документов, предназначенных для обмена ими в Интернете

Сферический HTML в вакууме

```
<!DOCTYPE html>
<html lang="ru">
  <head>
    <title>Google</title>
    <link rel="stylesheet"
          href="/static/css/style.css" />
  </head>
  <body>
    <div>google.com</div>
    <div>
      <input type="text" name="search"
                placeholder="Запрос"/>
    </div>
    <div>
      <button>Мне повезет</button>
      <button>Поиск</button>
    </div>
  </body>
</html>
```

Полезные ссылки

- HTML - https://developer.mozilla.org/ru/docs/Learn/HTML/Введение_в_HTML
- JS, CSS - http://learn.javascript.ru/

Работа с шаблонами в Django

Минимальный код

```
from django.shortcuts import render


def index(request):
    return render(request, 'index.html', {
        'title': 'Первая страница',
        'content': 'Какой то контент'
    })
```

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>{{ title }}</title>
  </head>
  <body>
    <main>
      <div>Django Templates работает, {{ content }}</div>
    </main>
  </body>
</html>
```

Рендеринг

- Термин в компьютерной графике, обозначающий процесс получения изображения по модели с помощью компьютерной программы.

В нашем случае из шаблона получаем готовый текст на базе входных данных

Функция render

```
render(request, template_name, context, status=status)
```

- request - Объект запрос
- template_name - Имя шаблона
- context - Словарь с данными которые попадают в шаблон
- status - возвращаемы HTTP-код (не обязательный аргумент, по-умолчанию 200)

Шаблонизаторы

Для формирования страницы django использует движки шаблонизации

- DjangoTemplates - стандартный шаблонизатор
- Jinja

Полезные ссылки

- Django Templates - https://docs.djangoproject.com/en/2.0/ref/templates/language/
- Jinja - http://jinja.pocoo.org/docs/2.10/templates/

Виды шаблонных литералов

- `{{ ... }}` - переменные, пример: `{{ title }}`
- `{% ... %}` - теги, пример: `{% if is_enabled %}<div>Выключить</div>{% endif %}`
- `| внутри {{ ... }}` - фильтры, пример: `{{ name|lower|truncatewords }}`
- `{# ... #}` - комментарии, пример: `{# Тут что то происходит и не попадет в html #}`

Часто встречаемые группы тегов

- Условия - `{% if ... %}{% else %}{% endif %}`
- Циклы - `{% for ... in ... %}{% endfor %}`
- Наследование - `{% include ... %}`, `{% block block_name %}{% endblock %}`, `{% extends ... %}`
- Статические файлы (js, css, img, ...) - `{% static ... %}`
- Загрузка расширений - `{% load ... %}`

Условия

```
...
<div>
{% if is_enabled %}
  <div>Выключить</div>
{% else %}
  <div>Включить</div>
{% endif %}
</div>
...
```

Циклы

```
{% for cat in cats %}
<div class="cat cat--{{ cat.type }}">
  <div>{{ cat.name }}</div>
  <div>{{ cat.description }}</div>
</div>
{% endfor %}
```

Наследование

base.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>{% block title %}{% endblock %}</title>
  </head>
  <body>
    <main>{% block content %}{% endblock %}</main>
  </body>
</html>
```

cat_list.html

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

{% block title %}Маркет котиков{% endblock %}
{% block content %}
        <div>...</div>
{% endblock %}
```

Все что нужно знать для использования базовых шаблонов (layout)

- Тэги
  - `{% extends 'template_name' %}`
  - `{% block content %}{% endblock %}`

Если мы используем базовый шаблон, тег `extends` должен идти в самой первой строчке

Подробнее про наследование шаблонов

- https://docs.djangoproject.com/en/2.0/ref/templates/language/#template-inheritance

Статические файлы

- Тег `{% static ... %}`

```
{% load static %}
...
<script src="{% static 'js/app.js' %}"></script>
<link rel="stylesheet" href="{% static 'css/styles.css' %}
...
<a class="logo" href="/">
  <img class="logo__img" src="{% static 'img/logo.svg' %}"
</a>
...
```

Отрендеренная страница

```
...
<link rel="stylesheet" href="/static/css/styles.css">
...
<script src="/static/js/app.js"></script>
...
<a class="logo" href="/">
  <img class="logo__img" src="/static/img/logo.svg" >
</a>
...
```

Поиск шаблонов и статики

Стандартно Django настроена на поиск:

- Шаблонов в директории templates например:
  `./blog/articles/templates/...`
- Статики в директориии static например:
  `./blog/articles/static/...`

Настройка для поиска шаблонов

settings.py

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.Django
--->    'DIRS': [
            os.path.join(BASE_DIR, 'templates')
        ],
--->    'APP_DIRS': True,
        ...
    },
]
```

Настройка для поиска статики

Относительно директорий из `STATICFILES_DIRS` будет происходить поиск статики

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static'),
    ...
]
```

Полезные функции о которых нужно знать

- `render_to_string(template_name, context)` - рендерить шаблон в строку
- `select_template(template_name_list)` - принимает на себя список шаблонов и выдает первый который нашли

На домашнее чтение

- context processors
  https://djangosimple.blogspot.com/2013/04/template-context-processor-django.html
- Как написать свой тег и фильтр
  https://docs.djangoproject.com/en/2.0/howto/custom-template-tags/
- Как работает django templates
  https://docs.djangoproject.com/en/2.0/ref/templates/api/#using-requestcontext