ВВЕДЕНИЕ ВО FLASK

Первая страница — это просто

Flask — это микрофреймворк на Python для создания веб-приложений. Он прост в освоении и отлично подходит для обучения.

Установка Flask

```
pip install flask
```

Простой пример

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return 'Привет, Flask!'

if __name__ == '__main__':
    app.run(debug=True)
```

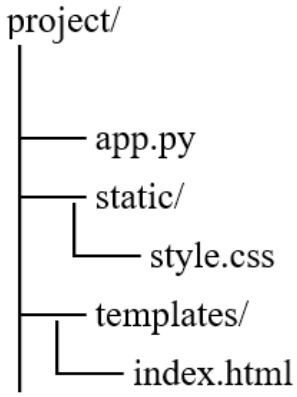
Объяснение

- @app.route('/') — указывает, по какому пути будет доступна функция.
- app.run(debug=True) — запускает сервер в режиме отладки.

Статический контент

Flask автоматически обрабатывает папку static, откуда можно подключать CSS, JS и изображения.

Структура проекта



Пример:

index.html

```
<!doctype html>
<html>
<head>
    <link rel="stylesheet" href="{{ url_for('static',
filename='style.css') }}">
    <title>Моя первая страница</title>
</head>
<body>
    <h1>Добро пожаловать!</h1>
</body>
</html>
```

app.py

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')
```

В Flask **можно переназначить папку static** на любую другую, указав её при создании экземпляра приложения.

```
from flask import Flask
# 'assets' — новая папка вместо 'static'
app = Flask(__name__, static_folder='assets')

.....
```

Теперь Flask будет искать статические файлы (CSS, JS, изображения) в папке assets.

Пример использования в шаблоне

```
<link rel="stylesheet" href="{{ url_for('static',
filename='style.css') }}">
```

Этот код по-прежнему работает, но теперь Flask ищет style.css внутри папки assets.

Дополнительно: можно изменить и путь в URL

По умолчанию статические файлы доступны по /static/<имя_файла>. Если необходимо изменить и это, указываем аргумент static_url_path.

```
app = Flask(
    __name__,
    static_folder='assets',        # физическая папка
    static_url_path='/files'       # URL-путь в браузере
)
```

Теперь файл assets/style.css будет доступен по адресу:

http://localhost:5000/files/style.css

Подключение стилей Bootstrap

Bootstrap — это популярный CSS-фреймворк для быстрой стилизации страниц. Подключение Bootstrap с помощью CDN:

Пример:

```
<head>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>
<body class="bg-light text-center">
  <div class="container">
    <h1 class="mt-5">Привет, мир!</h1>
    <p class="lead">Это страница с Bootstrap-стилями.</p>
  </div>
</body>
```

Переход к динамическому содержимому: параметры адресной строки

Можно передавать значения через URL и обрабатывать их во view-функциях.

Пример

```
@app.route('/user/<name>')
def user(name):
    return f'Привет, {name}!'
```

Открываем в браузере:

http://localhost:5000/user/Иван → Привет, Иван!

Обработка форм (GET и POST)

Теория

Flask может обрабатывать формы, используя методы GET и POST.

HTML-форма

```
<form method="POST">
    <input type="text" name="username"
placeholder="Введите имя">
    <input type="submit" value="Отправить">
</form>
```

View-функция

```
from flask import request

@app.route('/form', methods=['GET', 'POST'])
def form():
    if request.method == 'POST':
        username = request.form['username']
        return f'Привет, {username}!'
    return render_template('form.html')
```

Упражнения

Упражнение 1:

Создайте страницу /hello/<имя>, которая приветствует пользователя.

Упражнение 2:

Создайте HTML-страницу с формой, которая отправляет имя и выводит его на новой странице.

Упражнение 3:

Добавьте выбор цвета фона через выпадающий список и примените его.

РЕШЕНИЯ

Упражнение 1:

```
@app.route('/hello/<name>')
def hello(name):
    return f'Привет, {name}!'
```

Упражнение 2:

form.html:

```
<form method="POST">
  <input type="text" name="name">
  <button type="submit">OK</button>
</form>
```

app.py:

```
@app.route('/greet', methods=['GET', 'POST'])
def greet():
    if request.method == 'POST':
        name = request.form['name']
        return f'Здравствуйте, {name}!'
    return render_template('form.html')
```

Упражнение 3:

form.html:

```
<form method="POST">
  <input type="text" name="name" placeholder="Имя">
  <select name="color">
    <option value="white">Белый</option>
    <option value="lightblue">Голубой</option>
    <option value="lightgreen">Зеленый</option>
  </select>
  <button type="submit">OK</button>
</form>
```

app.py:

```
@app.route('/custom', methods=['GET', 'POST'])
def custom():
    if request.method == 'POST':
        name = request.form['name']
        color = request.form['color']
        return f'<body style="background-
color:{color}">Привет, {name}!</body>'
    return render_template('form.html')
```