WEB DEVELOPMENT WITH PYTHON

2. web python без django

В Python есть встроенный CGI сервер, который запускается следующим образом (из рабочей директории):

```
1 python3 -m http.server --cgi
```

Веб-приложение доступно по адресу `localhost:8000`.

В поддиректории `cgi-bin` размещаются исполняемые скрипты (`chmod +x`).

3. Django vs Flask

- *Django* — всё-в-одном фреймворк для стандартного набора задач: регистрация, база данных, e-mail. Комплектуется встроенной Django ORM (Object-Relational Mapping).
- *Flask* — легковесный и гибкий фреймворк. Позволяет выбирать модули под конкретные задачи и устанавливать их по мере необходимости. Например, для ORM подключается библиотека SQLAlchemy.

4.1. **Простейшее приложение.**

Файл helloworld.py:

```
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route('/')
5 def hello_world():
6     return 'Hello World!'
7
8 if __name__ == '__main__':
9     app.run()
```

Запуск веб-приложения: `python3 helloworld.py`

Веб-страница доступна по адресу `http://127.0.0.1:5000/`.

Созданный объект `app` класса `Flask` является WSGI-приложением (Web Server Gateway Interface).

4.2. **Структура проекта.**

Рекомендуемая структура:

```
config.py
requirements.txt
run.py
instance/
    config.py
yourapp/
    __init__.py
    views.py
    models.py
    forms.py
    static/
    templates/
```

4.3. **HTML-шаблоны.**

HTML-файлы (шаблоны) располагаются в папке `templates`.

Их вывод осуществляется, например, так:

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

4.4. **Передача данных в шаблон.**

```
   date = 'some data'
   return render_template('template.html', data=data)
```

```
<!DOCTYPE html>
<html>
 <body>
   Some text
   {{ data }}
 </body>
</html>
```

4.5. **css и статические файлы.**

Для подключения ccs и img добавить в html-шаблон:

```
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
<link rel=stylesheet type=text/css href="{{ url_for('static', filename='style.css') }}">
<img src="{{ url_for('static', filename='img/1.jpg') }}" />
```

Сами файлы находятся в каталоге `static`:

static/css/style.css

static/img/1.jpg

Jinja2 — язык шаблонов для Flask (и не только).

```
{% if not session.logged_in %}
   {{ data.html | safe }}
{% else %}
   {{ somedata }}
{% endif %}

{% for entry in entries %}
   <li><h2>{{ entry.title }}</h2>{{ entry.text|safe }}
{% else %}
   <li><em>Unbelievable.  No entries here so far</em>
{% endfor %}
```

Разделители (delimiter):

```
{% ... %} for Statements
{{ ... }} for Expressions to print to the template output
{# ... #} for Comments not included in the template output
#  ... ## for Line Statements
```

Переменные:

```
{{ foo.bar }}
{{ foo['bar'] }}
```

Литераты (Literals):

- `''Hello World''`
- `42 / 42.23`
- `['l','i','s','t']`
- `('tup','le')`
- `{'di' : 'ct'}`
- `true, false.`

Операции:

- Сравнение: `==, !=, >, >=, <, <=`

6.1. Шаблон адреса URL.

```
@app.route('/', defaults={'path': ''})
@app.route('/<path:path>')
def catch_all(path):
    return 'You want path: %s' % path
```

6.2. Класс request.

```
from flask import request
>>> type(request)
<class 'werkzeug.local.LocalProxy'>
```

```
request.url:                http://127.0.0.1:5000/alert/dingding/test?x=y
request.base_url:           http://127.0.0.1:5000/alert/dingding/test
request.url_charset:        utf-8
request.url_root:           http://127.0.0.1:5000/
str(request.url_rule):      /alert/dingding/test
request.host_url:           http://127.0.0.1:5000/
request.host:               127.0.0.1:5000
request.script_root:
```

```
request.path:                /alert/dingding/test
request.full_path:           /alert/dingding/test?x=y
```

7.1. **Menue.** `pip3 install Flask-Menu`

WSGI (англ. Web Server Gateway Interface) — стандарт взаимодействия между Python-программой, выполняющейся на стороне сервера, и самим веб-сервером, например Apache.

WSGI предоставляет простой и универсальный интерфейс между большинством веб-серверов и веб-приложениями или фреймворками.

По стандарту, WSGI-приложение должно удовлетворять следующим требованиям:

- должно быть вызываемым (callable) объектом (обычно это функция или метод)
- принимать два параметра:
- словарь переменных окружения (environ)
- обработчик запроса (`start_response`)
- вызывать обработчик запроса с кодом HTTP-ответа и HTTP-заголовками
- возвращать итерируемый объект с телом ответа

8.1. Установка mod-wsgi.

```
apt-get install libapache2-mod-wsgi
```

8.2. Настройка Apache.

```
<VirtualHost *>
    ServerName example.com

    WSGIDaemonProcess yourapplication user=user1 group=group1 threads=5
    WSGIScriptAlias / /var/www/yourapplication/yourapplication.wsgi

    <Directory /var/www/yourapplication>
        WSGIProcessGroup yourapplication
        WSGIApplicationGroup %{GLOBAL}
        Order deny,allow
        Allow from all
    </Directory>
</VirtualHost>
```

8.3. **Создание файла .wsgi.** Этот файл (в корне проекта) содержит код, выполняемый `mod_wsgi` для получения объекта приложения:

```
from yourapplication import app as application
```

Объект с именем `application` будет использоваться в качестве приложения.

На диске должен быть также файл `yourapplication.py`, в котором создается объект app класса flask.

9. Flask + MongoDB

`https://flask-pymongo.readthedocs.io/en/latest/`

10. Bokeh

```
sudo pip3 install bokeh
```

11. Dash, React

Dash: `https://tproger.ru/translations/reactive-web-apps-with-python/`

React: `https://tproger.ru/translations/react-basic-weather-app/`

Приложения на `Dash` — веб-серверы, которые запускают Flask и связывают пакеты `JSON` через HTTP-запросы. Интерфейс Dash формирует компоненты, используя `React.js`.

Install

```
sudo pip3 install dash dash-renderer dash-html-components dash-core-components plotly
```

Чтобы Dash работал, нужно в wsgi-файле строку

```
1 from myapp import app as application
```

заменить на

```
from myapp import app
application = app.server
```