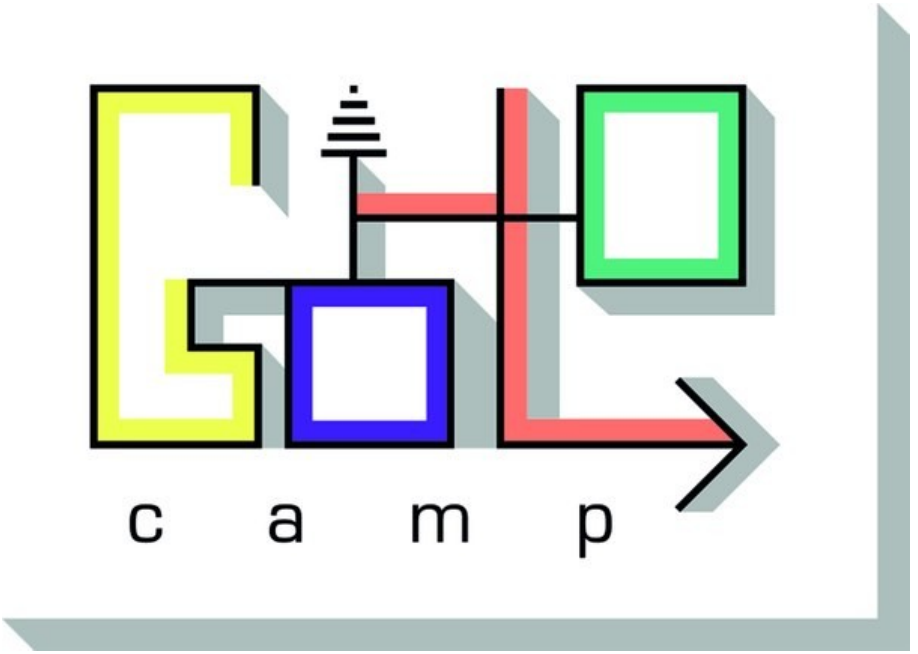
Основы веб-разработки

GET / POST / FLASK / SQL etc.

Павел Куликов
kulikovpavel@gmail.com



Что есть WEB

- Интерне́т (англ. Internet, МФА: [ˈɪn.tə.net]) — всемирная система объединённых компьютерных сетей для хранения и передачи информации. Часто упоминается как Всемирная сеть и Глобальная сеть, а также просто Сеть. Построена на базе стека протоколов TCP/IP. На основе Интернета работает Всемирная паутина (World Wide Web, WWW) и множество других систем передачи данных.
- К 30 июня 2012 года число пользователей, регулярно использующих Интернет, составило более чем 2,5 млрд. человек, более трети населения Земли пользовалось услугами Интернета. К середине 2015 года число пользователей достигло 3,3 млрд. человек

Под капотом

- TCP/IP 4, адреса вида 88.168.44.22, заканчиваются

  127.0.0.1 — ваш компьютер

  192.168.0.1 — 0.255 — локальная сеть

  192.168.1.1 — 1.255
- TCP/IP 6, будущее, кончатся нескоро, вид:

  2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d

  Самое то для интернета вещей, которых ожидается много

DNS

- DNS (англ. Domain Name System — система доменных имён) — компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения
- DNS-серверы сами определяются по IP-адресам
- Ответы на запросы от DNS-сервера:

  Yandex.ru - > 5.255.255.5

  Google.ru - > 173.194.73.94

  ...

Браузер

- Бра́узер или веб-обозрева́тель (от англ. Web browser) — прикладное программное обеспечение для просмотра веб-страниц; содержания веб-документов, компьютерных файлов и их каталогов; управления веб-приложениями; а также для решения других задач. В глобальной сети браузеры используют для запроса, обработки, манипулирования и отображения содержания веб-сайтов

HTTP протокол

- Сервера получают запросы определенного вида и возвращают ответ.

telnet auto.ru 80

GET /

HTTP/1.1 200 OK

Server: nginx

Date: Tue, 26 Jul 2016 21:45:57 GMT

Content-Type: text/html; charset=UTF-8

…

<!doctype html>

<html xmlns:og="http://ogp.me/ns#">

<head>

<title>Покупка и продажа автомобилей в России: купить авто новый или с пробегом. Купить, продать и обменять машину в России на АВТО.РУ</title>

...

GET / POST

- GET в URL, для запроса данных:

  /test/demo_form.asp?name1=value1&name2=value2
- POST в теле запроса, для изменения данных
- Символы экранируются, строки запроса собирать долго, нужно преобразовывать ответ, прилагать файлы и т.п.

  В python библиотека requests

  pip install requests

Варианты ответа сервера

- Text (HTML)
- JSON
- XML
- Файлы

Троица веба

- HTML — HyperText Markup Language — «язык гипертекстовой разметки»
- JavaScript (динамическое изменение страниц)
- CSS — Cascading Style Sheets — каскадные таблицы стилей дизайн)

Работа браузера

- Получить IP адрес сервера через службы DNS, если указано имя, а не адрес сразу
- Выполнить запрос к серверу и получить ответ
- Отобразить страницу пользователю, учесть дизайн, выполнить скрипты
- Организовать хранилище информации, cookies, etc.

Работа сервера

- Принять запрос
- Обработать его, получить информацию из БД, сохранить информацию в БД и т.п.
- Вернуть ответ

Базы данных

Нужно место для постоянного хранения информации, но:

- Нужно хранить большие объемы данных
- Нужно быстро и удобно получать к ним доступ для чтения и изменения
- Гарантии безопасности при записи данных и т. п.
- Масштабируемость

Текстовые файлики, очевидно, не выход, слишком медленно и неудобно.

SQL

- SQL (англ. structured query language — «язык структурированных запросов») — формальный непроцедурный язык программирования, применяемый для создания, модификации и управления данными в произвольной реляционной базе данных, управляемой соответствующей системой управления базами данных (СУБД).

Создание схемы БД

```
CREATE TABLE Persons
(
    id int NOT NULL PRIMARY KEY,
    lastname varchar(255) NOT NULL,
    firstname varchar(255),
    address varchar(255),
    city varchar(255)
)
```

Частые команды

- SELECT CustomerName,City FROM Customers;
- SELECT * FROM Customers WHERE Country='Mexico';
- SELECT * FROM Customers ORDER BY Country DESC;
- INSERT INTO Customers (CustomerName, City, Country) VALUES ('Cardinal', 'Stavanger', 'Norway');
- DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste' AND ContactName='Maria Anders';

СУБД

- Система управления базами данных

  поддерживающие SQL:
- SQLite
- MySQL
- PostgreSQL
- MSSQL
- IBM DB2
- Etc…

  Небольшие отличия внешне, дополнительные надстройки в каждой

  Внутри может быть совершенно разное устройство.

NoSQL

- Модный тренд
- Нужно смотреть на задачи и понимать отличия
- MongoDB, Redis etc
- Документо-ориентированные, ключ-значения и другие

История веба

- От возможностей
- Статические страницы с минимумом дизайна и графики
- 1995 JavaScript, принят разработчиками браузеров, сейчас один из самых популярных языков программирования
- Динамически подгружаемые сайты (React, Angular etc.)
- Постепенное изменение стандартов

Разработка серверов

- Нужно принимать и обрабатывать запросы
- PHP (Hypertext Preprocessor — «PHP: препроцессор гипертекста»; первоначально Personal Home Page Tools — «Инструменты для создания персональных веб-страниц») — скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.
- Плохая модульность, морально устарел
- Нужно собирать HTML страницу, избегать повторений кода и т.п.

Веб-фреймворки

- Каркас веб-приложений (Web application framework, WAF) — это каркас, предназначенный для создания динамических вебсайтов, сетевых приложений, сервисов или ресурсов. Он упрощает разработку и избавляет от необходимости написания рутинного кода. Многие каркасы упрощают доступ к базам данных, разработку интерфейса, и также уменьшают дублирование кода.

- Существуют специализированные типы каркасов веб-приложений, например, каркасы для создания систем управления содержимым. (WordPress, Joomla, Drupal etc)

Примеры

- Django (python)
- Ruby on Rails (ruby)
- Symfony (php)
- Spring (java)
- CodeIgniter (php)
- Play (java)

Микрофреймворк Flask (python)

- “Микро” не означает, что ваше веб-приложение целиком помещается в один файл с кодом на Python, хотя, конечно же это может быть и так. Также, это не означает, что Flask испытывает недостаток функциональности. “Микро” в слове “микрофреймворк” означает, что Flask стремится придерживаться простого, но расширяемого ядра.

Установка

- pip install Flask

Документация:

http://flask.pocoo.org/docs/0.11/quickstart

Hello, Flask!

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

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

Сервер будет доступен локально по порту 5000

Если нужна доступность извне:

```
app.run(host='0.0.0.0')
```

Отладка

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

Роутинг

```
@app.route('/')
def index():
    return 'Index Page'


@app.route('/hello')
def hello():
    return 'Hello, World'
```

Параметры в URL

```
@app.route('/user/<username>')
def show_user_profile(username):
    # show the user profile for that user
    return 'User %s' % username

@app.route('/post/<int:post_id>')
def show_post(post_id):
    # show the post with the given id, the id is an integer
    return 'Post %d' % post_id
```

Строительство URL

- >> from flask import Flask, url_for
- >>> app = Flask(__name__)
- >>> @app.route('/')
- ... def index(): pass
- ...
- >>> @app.route('/login')
- ... def login(): pass
- ...
- >>> @app.route('/user/<username>')
- ... def profile(username): pass
- ...
- >>> with app.test_request_context():
- ... print url_for('index')
- ... print url_for('login')
- ... print url_for('login', next='/')
- ... print url_for('profile', username='John Doe')

GET / POST

```
from flask import request


@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        do_the_login()
    else:
        show_the_login_form()
```

Статические файлы

- JS, CSS, картинки
- Лучше через nginx или apache
- Подкаталог «static»
- url_for('static', filename='style.css')

Шаблоны

- Составлять HTML с помощью вставок в строку очень криво
- Решение — шаблоны

```
from flask import render_template

@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', name=name)
```

Путь

Подкаталог templates

/application.py

/templates

    /hello.html

Сам шаблон

```
<!doctype html>
<title>Hello from Flask</title>
{% if name %}
  <h1>Hello {{ name }}!</h1>
{% else %}
  <h1>Hello, World!</h1>
{% endif %}
```

Шаблонизатор

- Jinja2
- http://jinja.pocoo.org/
- Циклы, переменные, фильтры и т.п.

Python like - синтаксис

```
<title>{% block title %}{% endblock %}</title>
<ul>
{% for user in users %}
  <li><a href="{{ user.url }}">{{ user.username }}</a></li>
{% endfor %}
</ul>
```

Блоки, extends

```
{% extends "layout.html" %}
{% block body %}
  <ul>
  {% for user in users %}
    <li><a href="{{ user.url }}">{{ user.username }}</a></li>
  {% endfor %}
  </ul>
{% endblock %}
```

Еще

```
<ul>
{% for user in users %}
    <li>{{ user.username|e }}</li>
{% else %}
    <li><em>no users found</em></li>
{% endfor %}
</ul>
```

Условия

```
{% if users %}
<ul>
{% for user in users %}
    <li>{{ user.username|e }}</li>
{% endfor %}
</ul>
{% endif %}
```

Доступ к параметрам

```
@app.route('/login', methods=['POST', 'GET'])
def login():
    error = None
    if request.method == 'POST':
        if valid_login(request.form['username'],
                       request.form['password']):
            return log_the_user_in(request.form['username'])
        else:
            error = 'Invalid username/password'
    # the code below is executed if the request method
    # was GET or the credentials were invalid
    return render_template('login.html', error=error)
```

GET параметры

- request.args
- Словарь dict()
- Чтобы не ловить KeyError:

  searchword = request.args.get('key', '')

Редирект

```
from flask import abort, redirect, url_for


@app.route('/')
def index():
    return redirect(url_for('login'))


@app.route('/login')
def login():
    abort(401)
    this_is_never_executed()
```

Сесии

```
from flask import Flask, session, redirect, url_for, escape, request

app = Flask(__name__)

@app.route('/')
def index():
    if 'username' in session:
        return 'Logged in as %s' % escape(session['username'])
    return 'You are not logged in'

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        session['username'] = request.form['username']
        return redirect(url_for('index'))
    return '''
        <form action="" method="post">
            <p><input type=text name=username>
            <p><input type=submit value=Login>
        </form>
    '''
```

Сессии продолжение

```
@app.route('/logout')
def logout():
    # remove the username from the session if it's there
    session.pop('username', None)
    return redirect(url_for('index'))

# set the secret key.  keep this really secret:
app.secret_key = 'A0Zr98j/3yX R~XHH!jmN]LWX/,?RT'
```

HTML Forms

```
<form>
  ...
</form>
```

По умолчанию GET

POST

```
<form action="/register" method="post">
  First name:<br>
  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>
  <input type="text" name="lastname"
value="Mouse"><br><br>
  <input type="submit" value="Submit">
</form>
```

Api, JSON

```
from flask import Flask, jsonify, render_template, request


@app.route('/_add_numbers')
def add_numbers():
    a = request.args.get('a', 0, type=int)
    b = request.args.get('b', 0, type=int)
    return jsonify(result=a + b)
```

```
<script type=text/javascript>
  $(function() {
    $('a#calculate').bind('click', function() {
      $.getJSON($SCRIPT_ROOT + '/_add_numbers', {
        a: $('input[name="a"]').val(),
        b: $('input[name="b"]').val()
      }, function(data) {
        $("#result").text(data.result);
      });
      return false;
    });
  });
</script>
<p><input type=text size=5 name=a> +
   <input type=text size=5 name=b> =
   <span id=result>?</span>
<p>
<a href=# id=calculate>calculate server side</a>
```

Работа с БД

- SQL напрямую, сойдет для маленьких проектов

- ORM - Object-Relational Mapping, объектно-реляционное отображение, связывает БД с концепциями ООП, запись — объект со свойствами и методами

  Пример:

  http://www.sqlalchemy.org/

Альтернативы

- Mongo, ORM-like синтаксис из коробки
- ORM, встроенные во фреймворк, см. Django, под разные БД

Переход дальше

- Django
- REST
- Client-side frameworks, dynamic pages

  AngularJS, React

- Павел Куликов
- kulikovpavel@gmail.com

- https://github.com/Kulikovpavel/GotoCampWeb