# Проектирование Web сервиса.

| **№ урока:** | 8 | **Курс:** | Python Practice |
|---|---|---|---|

| **Средства обучения:** | Интерпретатор Python, virtualenv, текстовый редактор |
|---|---|

## Обзор, цель и назначение урока

Цель урока: познакомить студентов с понятием web разработки, а также с web фреймворками в Python. На примере web фреймворка Flask показать преимущества использования фреймворков при разработке web приложений. Создать простейшее web приложение с использованием Flask. Спроектировать web проект Golden-Eye - определить постановку задачи для сервиса.

## Изучив материал данного занятия, учащийся:

- Узнает, что такое web фреймворки, зачем они нужны.
- Сможет использовать Flask фреймворк для создания простейших web приложений.
- Изучит основные элементы Flask фреймворка - маршрутизация, шаблонизатор, конфигурирование сервиса, запуск приложения

## Содержание урока

1. Краткое резюме, что мы уже имеем к данному момент и обзор дальнейших действий
2. Что такое web разработка?
3. Что такое web фреймворки, зачем они нужны.
4. Web фреймворки в Python - какие существуют. "batteries-included" философия VS "extensibility" философия.
5. Построение Hello Word приложения на Flask, запуск, изучение основных элементов приложения.

## Резюме

- Мы начинаем вторую часть нашего курса — разработка и реализация web сервиса на основе кода, с реализацией классов api. Необходимо вспомнить постановку задачи — что же будет делать web сервис Golden-Eye.
- Web разработка, как мы уже говорили в самом начале нашего курса — это создание, разработка web приложений, web сервисов. Web-сервис — специальная служба, доступная через интернет. Web-сайт — как частный случай web-сервисов.
- Каждое Web приложение должно "уметь" принимать и обрабатывать HTTP запросы, формировать ответы, а также делать очень много другого функционала, который в целом не зависит от конкретного проекта - например, роутинг урлов, работа с сессиями, аутентификация пользователей и тп.
- Для упрощения разработки, чтоб для каждого web приложения не приходилось реализовывать весь этот функционал на таком техническом уровне — используются web фреймворки - набор библиотек, которые предоставляют общие шаблоны для построения надежных, масштабируемых и поддерживаемых веб-приложений. И разработчику остается только реализовать бизнес логику web приложения.
- Несмотря на то, что использование web фреймворка не является обязательным условием для построения web приложения, очень редко разработчик не использует их для ускорения разработки.



CyberBionic systematics

CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

- Существует большое количество web фреймворков на python. Наиболее популярные из них - Django и Flask. Каждый из них реализован в концепции, противоположной друг другу.
- Django реализован в стиле так называемом “batteries-included”, фреймворк включает в себя весь необходимый функционал для построения приложения - от роутинга урлов до orm библиотеки доступа к БД, работы с сессиями. При таком подходе, достаточно установить фреймворк, и все, больше никаких библиотек устанавливать не нужно. Но если мы захотим использовать другую orm библиотеку, например, то это может стать большой проблемой, если вообще возможно.
- В отличии от Django и batteries-included подхода, Flask построен с применением философии extensibility, расширяемости. То есть он включает только небольшое core с минимальным функционалом, но зато очень легко расширяем. То есть можно устанавливать нужные модули и использовать их. Flask считается более Pythonic, чем веб-фреймворк Django, потому что в обычных ситуациях эквивалентное веб-приложение Flask более явное и простое. Flask также более удобен для начинающего web разработчика.
- Прежде чем использовать Flask фреймворк, необходимо установить его в окружение для проекта — с помощью команды pip install flask==1.0.2 (с указанием версии).
- Для создания простейшего web приложения на основе Flask фреймворка достаточно создать один(!) python модуль, в котором будет несколько строчек кода — импорт класса Flask, создание экземпляра этого класса — это и будет наше Flask приложение. Также необходимо описать функцию и в декораторе описать урл, вызов которого приведет к вызову описанной функции. И вызвать метод run у экземпляра класса Flask, нашего приложения. При этом приложение будет запущено на локальном сервере, и указанный урл будет доступен в браузере!

## Закрепление материала

- Что такое web разработка?
- Для чего нужны web фреймворки?
- В чем основное отличие Flask и Django фреймворков?
- Как создать простейший сайт с помощью фреймворка Flask? Какой минимальный код должна содержать его реализация?

## Дополнительное задание

Задание

Добавить новый url и новую функцию в модуль приложения - функция с параметром name, результат которой — строка вида "Hello, <name>!" Проверить работу url в браузере.

## Самостоятельная деятельность учащегося

Задание 1

Написать скрипт для проверки работы Flask приложения, созданного на уроке. Скрипт должен отправлять GET запрос на url приложения и проверять, что в ответ приходит строка Hello word.

Задание 2

Добавить логирование в модуль приложения - создать объект-логгер, добавить логирование при запуске приложения, а также при вызове функций.

## Рекомендуемые ресурсы

http://flask.pocoo.org/docs/1.0/
http://jinja.pocoo.org/docs/2.10/



CyberBionic Systematics ® 2018
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com