



Введение в разработку веб- приложений на Python с использованием веб-фреймворков



Классификация веб-фреймворков

- Клиентские фреймворки

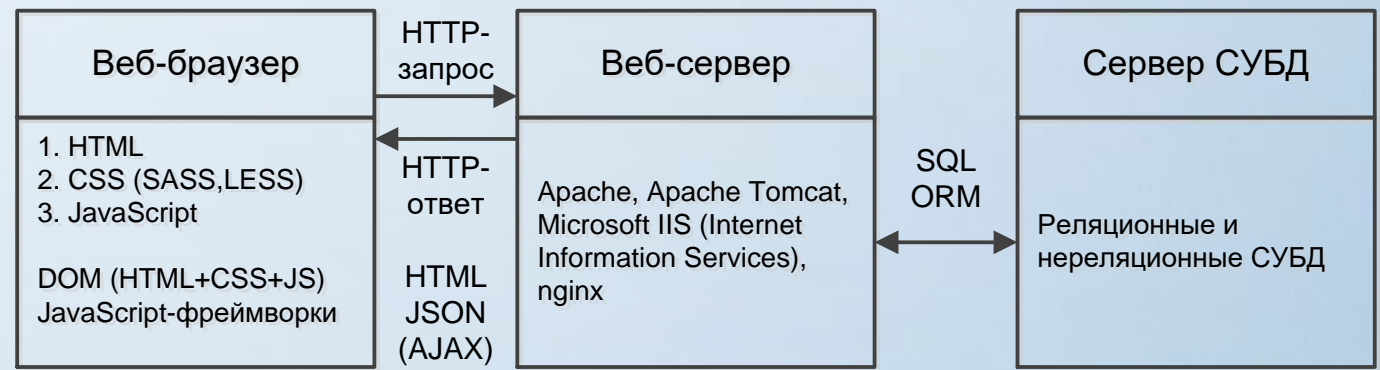
Предназначены для разработки [SPA](#). Реализуют концепцию «толстого» клиента и «тонкого» сервера. Основная функциональность реализована с использованием JavaScript (и [транспилируемых](#) в него языков). Некоторые используют паттерн MVC или его разновидности.

- [Серверные фреймворки](#)

Предназначены для разработки приложений на стороне веб-сервера. Реализуют концепцию «тонкого» клиента и «толстого» сервера. Используют традиционные языки веб-разработки: Python, PHP, Ruby, C#, ...

Подразделяются на две категории:

- [Микрофреймворки](#).
- Традиционные фреймворки с полной функциональностью (используют паттерн MVC или его разновидности).



- Универсальные фреймворки (пример [Meteor](#)).

Веб-разработка на Python

- Интерпретаторы некоторых языков, изначально ориентированных на применение в WWW (например, PHP), обладают встроенным шаблонизатором HTML и могут непосредственно использоваться для веб-разработки.
- В отличие от таких языков, Python для веб-разработки использует исключительно фреймворки.
- Для интеграции с веб-серверами в Python используются спецификация [WSGI](#), которая основана на [CGI](#).
 - В частности, для интеграции с веб-сервером Apache разработан модуль Apache [mod_wsgi](#).
 - Спецификация WSGI включает такое важное понятие как «[Middleware](#)».
- Дальнейшим развитием спецификации WSGI является спецификация [ASGI](#), которая ориентирована на разработку как синхронных, так и асинхронных веб-приложений.

Микрофреймворк Flask

- Документация.
- Создание простого приложения:
 - Установим виртуальное окружение (windows cmd):
 - `cd <каталог проекта>`
 - `python -m venv venv` #создадим виртуальное окружение
 - `venv\Scripts\activate` #активируем окружение
 - `pip install flask` # установим flask
 - `pip list`
 - Создадим в каталоге проекта Python-файл с простейшим обработчиком URL - <https://flask.palletsprojects.com/en/latest/quickstart/#a-minimal-application>
 - Запустим приложение

Традиционный серверный фреймворк. Шаблоны проектирования.

- Традиционный серверный фреймворк строится на шаблоне проектирования MVC или какой-либо из его разновидностей.
 - Шаблон проектирования [Model-View-Controller](#).
 - Шаблон проектирования [Model-View-Presenter](#).

Традиционный серверный фреймворк. Какие инструменты и возможности фреймворка использует разработчик для создания веб-приложения?

1. Статические файлы (статические HTML-документы, CSS, изображения, сценарии JavaScript и т.д.).
2. Контроллеры (обработчики событий пользовательских действий).
3. Модели (взаимодействие с БД).
4. Представления (view). Шаблоны, генерирующие HTML-страницы и другое динамическое содержимое.
5. Конфигурирование фреймворка (**конфигурирование в противоположность соглашениям о кодировании**):
 - Действия, выполняемые при запуске приложения.
 - Конфигурирование пользовательских сеансов (сессий).
 - [Переписывание URL](#) (привязка URL к контроллерам).
 - Безопасность ([аутентификация](#) и [авторизация](#)).
 - Кэширование.
 - Балансировка нагрузки.
 - Реализация [IOC](#) / [DI](#).
6. Утилиты командной строки для управления фреймворком.
 - [Скаффолдинг](#) (генерация кода одних частей приложения на основе информации о других частях приложения).

Создание структуры проекта, генерация кода контроллеров и представлений на основе моделей, генерация кода приложения на основе специализированных описаний, генерация форм ввода и редактирования данных во время работы приложения.
 - [Миграции](#) (изменение структуры базы данных на основе моделей).

Фреймворк Django (1)

- Документация:
 - [Оригинальная документация \(на английском языке\)](#)
 - [Русская документация](#)
- Особенности терминологии:
 - Представление, view (MVC) – шаблон, template (django).
 - Контроллер (MVC) – представление, view (django).
- Проекты и приложения (реализация модульной концепции)
 - Проект – это набор настроек и приложений для определенного веб-сайта.
 - Приложение – это независимое веб-приложение, в большинстве случаев работающее с БД. Приложение можно перенести из одного проекта в другой.
 - Аналогичные концепции в других фреймворках:
 - В ASP.NET используется концепция [области \(area\)](#).
 - В технологии [веб-порталов](#) сам портал можно считать аналогом проекта. [Портлет](#) можно считать аналогом приложения, которому выделяется отдельная визуальная область в браузере.

Фреймворк Django (2)

- Создание приложения:
 - Установим виртуальное окружение (windows cmd):
 - `cd <каталог проекта>`
 - `python -m venv venv` #создадим виртуальное окружение
 - `venv\Scripts\activate` #активируем окружение
 - `pip install django` # установим django
 - `pip list`
 - `pip freeze > requirements.txt`
 - Используем учебник для создания приложения
 - [На русском языке.](#)