GeekBrains

Урок 1

Паттерны веб-представления

Паттерны веб-представления. WSGI-фреймворк. Шаблонизатор.

Регламент

1. 8 уроков по 2 часа.
2. Практические задания.
3. Видеозапись будет.
4. Задавайте вопросы.

Что будем изучать на курсе

Архитектура приложений

Принципы объектно-ориентированного программирования

Паттерны проектирования

Это позволит

Использовать лучшие практики,
а не изобретать велосипед

Разговаривать на одном языке с
продвинутыми коллегами

Увеличить вашу ценность как
специалиста

На этом уроке разберем

MVC, Page controller, Front controller

Как работают WSGI-фреймворки

Как написать свой WSGI-фреймворк

Как использовать шаблонизатор

Паттерны веб-представления данных

MVC: как это работает

Модель

- Автономна
- Тестируема

Контроллер

- Принимает запросы
- Взаимодействует с моделью
- Выбирает представление

MVC: как это работает

Представление

- Отображает данные

MVC: что даёт

Множественность вариаций представлений

Концентрация усилий

Параллельная разработка

Простота модификации

MVC: фреймворки

https://www.djangoproject.com

Page Controller (Контроллер страниц): идея

Один URL → один обработчик → одно представление

Page Controller: что даёт

Простота

Прямолиней-ность

Отделяет логику от представления

Подмножество MVC

Page Controller: реализации

https://docs.zope.org/zope2/zope2book/ZPT.html

Front Controller (Контроллер запросов): идея

Централизовать общие правила

Front Controller: что даёт

Контроль

Координация

Конфигурируемость

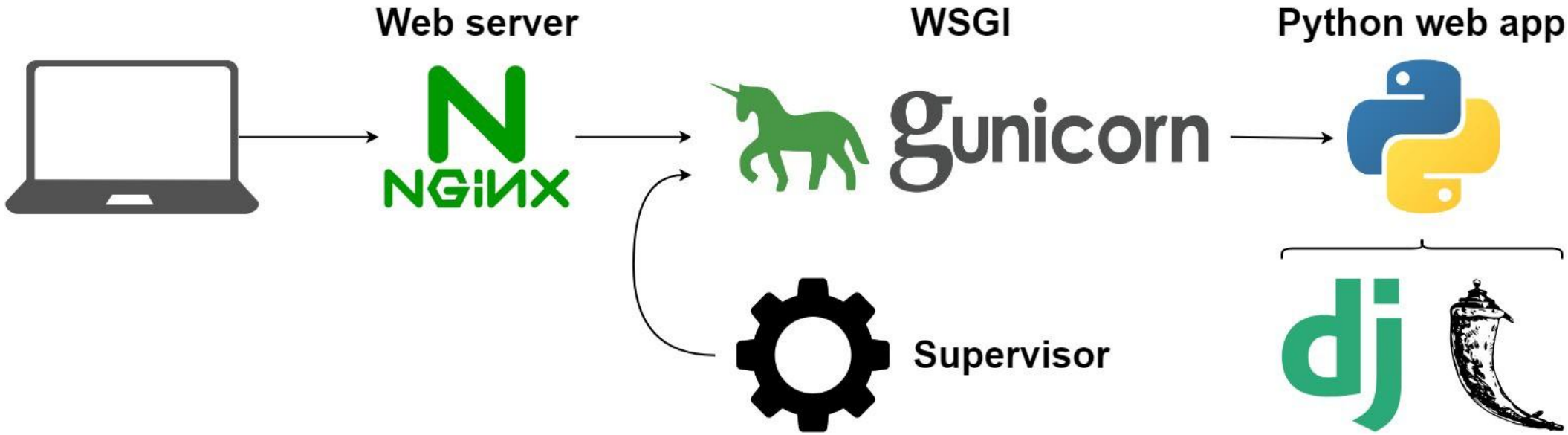
Front Controller: особенности

Уделить внимание оптимизации кода

Уже реализован в готовых фреймворках

WSGI-фреймворк

WSGI-фреймворк



simple_wsgi.py

```python
def application(environ, start_response):
    """
    :param environ: словарь данных от сервера
    :param start_response: функция для ответа серверу
    """
    # сначала в функцию start_response передаем код ответа и заголовки
    start_response('200 OK', [('Content-Type', 'text/html')])
    # возвращаем тело ответа в виде списка из bite
    return [b'Hello world from a simple WSGI application!']

```

Реализуем Page Controller и Front Controller

Использование шаблонизатора

templator.py

```python
from jinja2 import Template


def render(template_name, **kwargs):
    """
    Минимальный пример работы с шаблонизатором
    :param template_name: имя шаблона
    :param kwargs: параметры для передачи в шаблон
    :return:
    """
    # Открываем шаблон по имени
    with open(template_name, encoding='utf-8') as f:
        # Читаем
        template = Template(f.read())
    # рендерим шаблон с параметрами
    return template.render(**kwargs)


if __name__ == '__main__':
    # Пример использования
    output_test = render('authors.html', object_list=[{'name': 'Leo'}, {'name': 'Kate'}])
    print(output_test)

```

Практическое задание

В этой самостоятельной работе тренируем умения:

Использовать паттерны Page Controller, Front Controller

Использовать шаблонизатор

Смысл

Понимать и применять паттерны Page и Front Controllers, понимать, как устроены и работают WSGI-фреймворки

Использовать шаблонизаторы

Последователь ность действий

1. Создать репозиторий для нового проекта (Gitlab, GitHub, ...).
2. С помощью uwsgi или gunicorn запустить пример simple_wsgi.py, проверить, что он работает
3. Написать свой WSGI-фреймворк, используя паттерны Page Controller и Front Controller.

Описание работы фреймворка:

- возможность отвечать на get-запросы пользователя (код ответа + html-страница);
- для разных url-адресов отвечать разными страницами;
- page controller — возможность без изменения фреймворка добавить view для обработки нового адреса;
- front controller — возможность без изменения фреймворка менять обработку всех запросов.

Последователь ность действий

4. Реализовать рендеринг страниц с помощью шаблонизатора Jinja2. Документацию по этой библиотеке можно найти в дополнительных материалах.
5. Добавить любый полезный функционал в фреймворк, например обработку наличия (отсутствия) слеша в конце адреса.

Последователь ность действий

6. Добавить для демонстрации две любые разные страницы (например, главная и about, или любые другие).
7. Сдать ДЗ в виде ссылки на репозиторий.
8. В readme указать пример, как запустить фреймворк с помощью uwsgi и/или gunicorn.

Спасибо!

Каждый день вы становитесь лучше :)