УДК 004

Пример создание Web-сервера с помощью библиотеки Bottle на языке Python

Жуков Дмитрий Сергеевич
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
В данной статье показан пример создания Web-сервера. Он создан с применением библиотеки Bottle на языке Python. С помощью этой библиотеки удалось быстро создать Web-сервер.
Ключевые слова: python, сервер, web, программа.

An example of creating a Web server using the Bottle library in Python

Zhukov Dmitry Sergeevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article provides an example of creating a Web server. It was created using the Bottle library in Python. Using this library, we were able to quickly create a web server.
Keywords: python, server, web, program.

Веб сервер нужен всем, кто пользуется сетью Интернет для обмена информацией. Однако обращаться к нему приходится по большому счету не человеку, а программам и устройствам. Например, антивирусные программы часто просят сервер отыскать в сети и обновить свои базы данных. Активно сотрудничают с веб серверами мобильные телефоны, телевизоры и прочие устройства, которые имеют доступ в Интернет. При этом можно программы настроить на автоматическое обновление. Тогда участие пользователя в данном процессе будет практически ненужно. Но некоторые обновляющиеся базы данных все же лучше контролировать самостоятельно.

В статье И.Кутепова рассматривается пример построения вычислителя на основе нечеткой логики и применение языка Python для написания интегрированной среды разработки [2]. И.Е.Бронштейн в своей статье описал вывод типов для программного кода на языке Python. Сначала производится обзор описанных в научной литературе алгоритмов вывода типов для языков с параметрическим полиморфизмом. Затем даётся описание нового алгоритма, являющегося модификацией одного из предыдущих: алгоритма декартова произведения. Показывается, как модуль вывода типов, использующий новый алгоритм, анализирует различные конструкции языка

Python. Представляются результаты работы над прототипом [3]. В работе В.В.Найденова протестирована производительность пары аналогичных приложений, реализующих CRUD логику с помощью прослойки ORM. Сравнивается SQLAlchemy – де-факто стандартный ORM для Python с динамическим ORM для C++ собственной разработки – YB.ORM. Сравнивается производительность при использовании CPython и PyPy. Проверяется влияние отключения логов на производительность [4]. В статье И.Е.Бронштейн рассматриваются виды дефектов, которые обычно встречаются в программном коде на языке Python. Показывается, что возможные дефекты для Python не похожи на те, что часто встречаются в коде на Си/Си++ и, следовательно, необходимо исследование дефектов в крупных проектах с открытым исходным кодом. Даётся классификация найденных дефектов на основе того, нужен ли для нахождения ошибки вывод типов. Показывается, что существует небольшая доля "простых" дефектов, но для обнаружения большинства дефектов вывод типов необходим. Рассматривается вопрос, какие конструкции языка Python должны поддерживаться при выводе типов для нахождения реальных дефектов [5]. В работе Д.А.Кузнецов рассматривается структура интерфейса программы «Фармацевтическая экономическая безопасность» для фармацевтических организаций. Описывается функциональное предназначение и возможности пунктов основного меню прикладной программы [6]. Ю.А.Котов, А.В.Шаповалов в своей статье рассмотрели интерфейс программной реализации экспертной системы для восстановления простой замены букв текста. Описаны базовые элементы интерфейса, включающие выбор функциональных методов и базовых операций (замена, сдвиг, перебор). Не менее значимы иностранные исследования в данной сфере [8-9].

Разработку программы следует начать с загрузки нужных библиотек. Нам понадобятся библиотека Bottle для развертывания веб сервера.

Скачать нужные библиотеки можно введя в консоль команду «pip install Bottle».

Подключим все необходимые функции с этой библиотеки (Рис.1).

```
from bottle import route, run, get, request, static_file, post
```

Рис.1.Подключаем библиотеку

Для того что бы вывести текстовое сообщение при переходе по ссылке необходимо создать функцию с декоратором, параметром которого является относительная ссылка. Это функция должна вернуть строку. (Рис.2) Если запустить программу и перейти по ссылке (http://localhost:8080/message) в браузере то увидим данное сообщение.

```
@route('/message')
def hello():
    return "Today is a beautiful day"
```

Рис.2. Функция вывода текстового сообщения

Для того что бы вывести данные в формате JSON необходимо поменять вывод текстовых данных на именованный список. Пример показан на рисунке 3.

```
@route('/cars')
def getcars():

    cars = [{'name': 'Audi', 'price': 52642},
        {'name': 'Mercedes', 'price': 57127},
        {'name': 'Skoda', 'price': 9000},
        {'name': 'Volvo', 'price': 29000},
        {'name': 'Bentley', 'price': 350000},
        {'name': 'Citroen', 'price': 21000},
        {'name': 'Hummer', 'price': 41400},
        {'name': 'Volkswagen', 'price': 21600}]

    return dict(data=cars)
```

Рис.3. Пример вывода JSON

Теперь если перейдем по ссылке /cars, то получим JSON объект (Рис 4). Данная функция крайне необходима для создания API сайта.

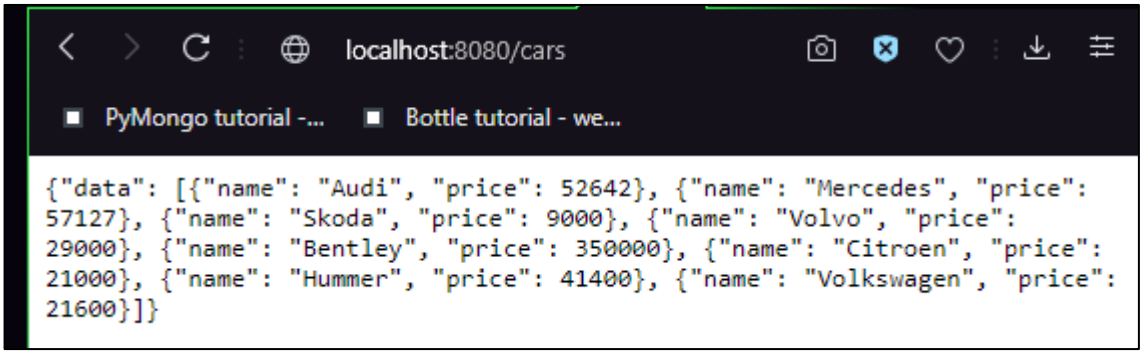

Рис.4. Вывод JSON

Для того что бы функция вывода веб ответа могла обработать данные из параметров GET нужно поменять декоратор route на get (рис.5)

```python
@get('/msg')
def message():

    name = request.query.name
    age = request.query.age

    return "{0} is {1} years old".format(name, age)
```

Рис.5. Функция обработки get параметров

Теперь если перейти по ссылке с параметрами name и age, то увидим результат обработки этих параметров (Рис.6).

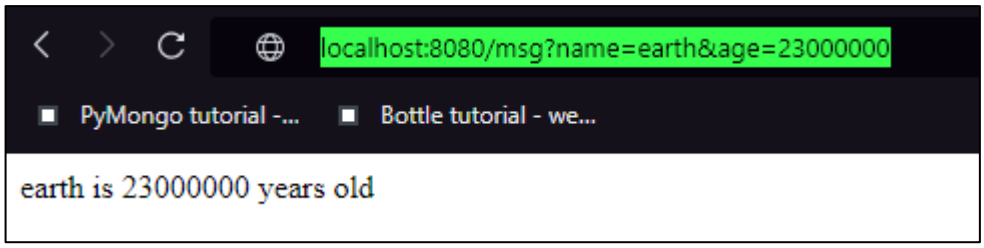

Рис.6. Результат обработки get параметра

Для вывода статических файлов на пример картинок и видео необходимо из ссылки получить путь до этого файла, передать этот путь как параметр функции. Вернуть файл с помощью функции static_file, в которую передаться параметры пути из ссылки и директории хранения статических файлов (Рис.7)

```python
@route('/<filepath:path>')
def server_static(filepath):
    return static_file(filepath, root='./public/')
```

Рис .7. Функция вывода статических файлов

Для примера выведем статический HTML файл (Рис 8). Для этого создадим папку public, добавим туда home.html. Для того что бы вывести его в браузере необходимо перейти по ссылке /home.html (Рис 8.)

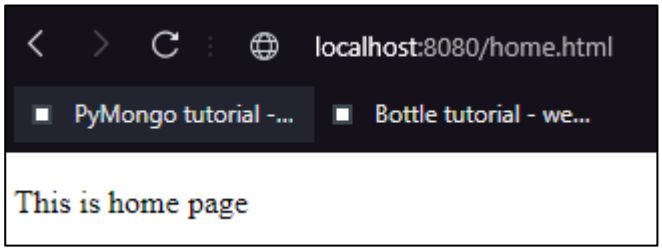

Рис.8. Добавление рефлектора

Для того что бы обработать post запрос необходимо поменять декортаор функции вывода на post. Данные post запроса можно получить из переменной requets.forms внутри функции (Рис 9).

```python
@post('/doform')
def process():

    name = request.forms.get('name')
    occupation = request.forms.get('occupation')
    return "Your name is {0} and you are a(n) {1}".format(name, occupation)
```

Рис.9. Функция обработки post запроса

Далее создадим HTML файл, содержащий форму (Рис 10). Разместим его в папке статических файлов.

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Home page</title>
</head>

<body>

    <form method="post" action="doform">
        <div>
            <label for="name">Name:</label>
            <input type="text" id="name" name="name">
        </div>
        <div>
            <label for="occupation">Occupation:</label>
            <input type="text" id="occupation" name="occupation">
        </div>
        <button type="submit">Submit</button>
    </form>

</body>

</html>
```

Рис .10. Html с формой

Теперь если перейти по ссылке /index.html то увидим форму с двумя текстовыми полями (Рис 11).

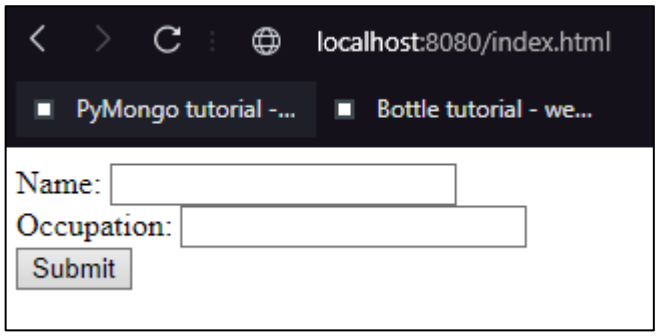

Рис.11. Форма ввода

Если в форму написать pgu и age и нажать кнопку submit, то увидим результат обработки формы (Рис.12)

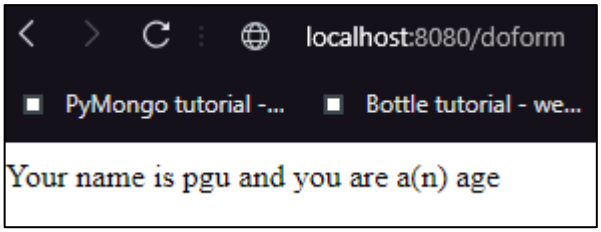

Рис.12. Результат post запроса

Таким образом, была показан пример работы библиотеки bottle. С помощью этой библиотеки можно легко создавать сайты.

Библиографический список

1. Лутц М. Изучаем python. М., 2009.
2. Кутепов И. Применение языка python при проектировании нечеткого контроллера // Компоненты и технологии. 2013. № 8 (145). С. 148-154.
3. Бронштейн И.Е. Вывод типов для языка python // Труды Института системного программирования РАН. 2013. Т. 24. С. 161-190.
4. Найденов В.В. Тестирование производительности orm в языках python и c++ // RSDN Magazine. 2014. № 1. С. 05-08.
5. Бронштейн И.Е. Исследование дефектов в коде программ на языке python // Программирование. 2013. Т. 39. № 6. С. 25-32.
6. Кузнецов Д.А. Интерфейс программы "фармацевтическая экономическая безопасность" // Российский медико-биологический вестник им. академика И.П. Павлова. 2009. № 2. С. 162-165.
7. Котов Ю.А., Шаповалов А.В. Интерфейс программы восстановления простой замены букв текста // Современные тенденции развития науки и технологий. 2016. № 4-4. С. 57-59.
8. Smith A. W. et al. Application program interface that enables communication for a network software platform : пат. 7117504 США. – 2006.
9. Parikh V., Moore R., Cheng H. Application program interface for a graphics system : пат. 6456290 США. – 2002.