УДК 004.624

Реализация web приложения с помощью CherryPy на языке программирования Python

Козич Полина Александровна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Кизянов Антон Олегович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Глаголев Владимир Александрович
Приамурский государственный университет имени Шолом-Алейхема
к.г.н., доцент кафедры информационных систем, математики и правовой информатики

Аннотация
В данной статье будет рассказано что из себя представляет web-фреймворк cherrypy описаны его достоинства и ключевые особенности. Написаны несколько примеров иллюстрирующих простоту использования данного web-фреймворка.
Ключевые слова: Python, cherrypy

Implementing a web application using CherryPy in the Python programming language

Kozich Polina Alexandrovna
Sholom-Aleichem Priamursky State University
Student

Kizyanov Anton Olegovich
Sholom-Aleichem Priamursky State University
Student

Glagolev Vladimir Alexandrovich
Sholom-Aleichem Priamursky State University
candidate of geographical sciences, Associate Professor of the Department of Information System, Mathematics and Law Informatics

Abstract
This article will explain what a cherrypy web framework is of its advantages and key features. Several examples are written to illustrate the ease of use of this web framework.
Keywords: Python, cherrypy

CherryPy позволяет разработчикам создавать веб-приложения практически так же, как и любые другие объектно-ориентированные программы на Python. Это приводит к уменьшению исходного кода за меньшее время. CherryPy - это веб-инфраструктура Python, которая предоставляет дружественный интерфейс для протокола HTTP для разработчиков Python. Это также называется библиотекой веб-приложений[1].

CherryPy использует сильные стороны Python в качестве динамического языка для моделирования и связывания протокола HTTP в API. Это один из старейших веб-фреймворков для Python, который обеспечивает чистый интерфейс и надежную платформу.

Цель исследования – дать базовое понимание работы с web-фреймворком cherrypy и реализацией нескольких примеров.

Ранее этим вопросом интересовались О.Б. Арушанян, Н.А. Богомолов, А.Д. Ковалев, М.Н. Синицын развивали тему «Архитектура клиентского программного обеспечения для web-приложений, ориентированных на представление данных» [2] в которой обсуждаются подходы к созданию WEB-приложений, ориентированных на представление данных, который основан на существенном использовании клиентского программного обеспечения (ПО), работающего в среде стандартного Интернет-браузера. В.Н. Грищенко с темой «Типовые элементы компонентно-ориентированной разработки web-приложений» [3], разбирались типовые решения и элементы процесса создания Web-приложений на базе моделей и методов компонентного программирования. Решение охватывает разные аспекты разработки, архитектуру и структуру приложений, структуру данных, операции их обработки. Описаны особенности каждого типового элемента. Д.Г. Юдин опубликовал статью «Поддержание качества web-приложения, написанного на языке php» [4] рассмотрел методы повышения и поддержания на должном уровне качества web-приложения. Так же приведено сравнение различных сред для создания тестов.

CherryPy - это web-фреймворк написанный на Python, который предоставляет простой интерфейс для обработки протокола HTTP для разработчиков Python.

CherryPy использует сильные стороны Python в качестве динамического языка для моделирования и связывания протокола HTTP в API. Это один из старейших веб-фреймворков для Python, который обеспечивает чистый интерфейс и надежную платформу.

Сильные стороны CherryPy

- **Простота**
  Разработка проекта в CherryPy - это простая задача достаточно несколько строк кода чтобы получить рабочий вариант. Основные компоненты легко использовать по отдельности для получения чистого кода в дальнейшей поддержке.

- **Мощность**
  CherryPy использует всю мощь Python. Он также предоставляет инструменты и плагины, которые ускоряют разработку и предоставляют инструменты необходимые для разработки реального приложения.
- **Открытый исходный код**
  CherryPy - это Python Web Framework с открытым исходным кодом, что означает, что эта платформа может использоваться в коммерческих целях.
- **Помощь Сообщества**
  У этого есть преданное сообщество, которое обеспечивает полную поддержку по различным вопросам. Сообщество пытается оказать полную помощь разработчикам, начиная с начального уровня до продвинутого уровня.

Основные требования для установки фреймворка CherryPy это

- Python с версией 2.4 или выше
- CherryPy 3.0

Установка CherryPy достаточно проста и выполняется с помощью следующей команды.

```
Pip install cherrypy
```

Есть несколько важных ключевых слов, которые необходимо определить, чтобы понять работу CherryPy.

- **Веб сервер**
  Это интерфейс, работающий с протоколом HTTP. Его цель - преобразовать HTTP-запросы к серверу приложений, чтобы они получали ответы.
- **Запрос**
  Это часть программного обеспечения, которая запрашивает информацию с сервера.

Далее показан пример кода CherryPy.

```
import cherrypy
class demoExample:
  def index(self):
  return "Hello World!!!"
  index.exposed = True
cherrypy.quickstart(demoExample())
```

Как работает код:

- Пакет с именем **CherryPy** всегда импортируется для обеспечения правильной работы.
- В приведенном выше примере функция с именем **index** возвращает параметр «Hello World !!!».
- Последняя строка запускает веб-сервер и вызывает указанный класс (здесь demoExample) и возвращает значение, указанное в индексе функции по умолчанию.

Результат работы можно просмотреть на рисунке 1.

Рис. 1 Результат работы

CherryPy поставляется с собственным веб-сервером (HTTP). Вот почему CherryPy является автономным и позволяет пользователям запускать приложение CherryPy в течение нескольких минут после установки библиотеки.

Веб - сервер выступает в качестве шлюза для приложения, с помощью которого все запросы и ответы хранятся в очереди.

Чтобы запустить веб-сервер, пользователь должен запустить следующую команду.

```
cherryPy.server.quickstart()
```

Встроенный web-сервер CherryPy отвечает за следующие виды деятельности:

- Создание и управление объектами запросов и ответов.
- Контроль и управление процессом CherryPy.

CherryPy разработан на основе концепции многопоточности. Каждый раз, когда разработчик получает или устанавливает значение в пространство имен CherryPy, это делается в многопоточной режиме. cherrypy.request, и cherrypy.response являются контейнерами потоковых данных, которые подразумевают, что ваше приложение вызывает их независимо, зная, какой запрос передается через них во время выполнения.

Каждый запрос обрабатывается своим собственным процессом Python. По этой причине производительность и стабильность работы сервера можно рассматривать как лучшую. Прием новых соединений и отправка данных обратно клиенту выполняется асинхронно из процесса запроса.

CherryPy имеет свой собственный диспетчер маршрутов, что позволяет ему отдавать различную информацию в зависимости от адреса запроса и его метода.

Вот список параметров для метода, требуемого для диспетчера маршрута

- Параметр name - это уникальное имя маршрута для подключения.
- Маршрут - это шаблон для сопоставления URI.
- Контроллер - это экземпляр, содержащий обработчики страниц.

Использование диспетчера маршрутов соединяет шаблон, соответствующий URI, и связывает определенный обработчик страницы.

Ниже представлен пример использования диспетчера маршрутов.

```
import random
import string
import cherrypy

class StringMaker(object):
  @cherrypy.expose
  def index(self):
    return "Hello! How are you?"

  @cherrypy.expose
  def generate(self, length=9):
    return ''.join(random.sample(string.hexdigits, int(length)))

if __name__ == '__main__':
  cherrypy.quickstart(StringMaker ())
```

Чтобы увидеть его работу нужно выполнить следующие шаги.

Шаг 1 - Сохранить вышеупомянутый файл как **Routes.py**.

Шаг 2 - Посетить следующий URL

```
http://localhost:8080/generate?length=10
```

Результат можно посмотреть на рисунке 2.

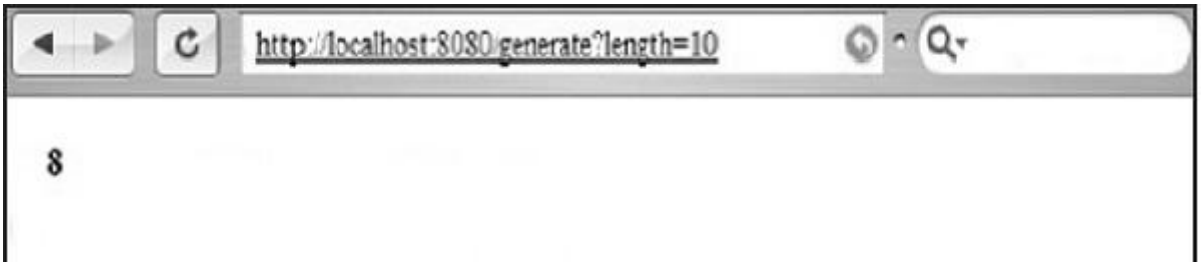

Рис. 2 Результат работы диспетчера маршрутов

Для создания более сложных web-приложений используют особую структуру расположения файлов, один из примеров такой структуры можно посмотреть на рисунке 3.

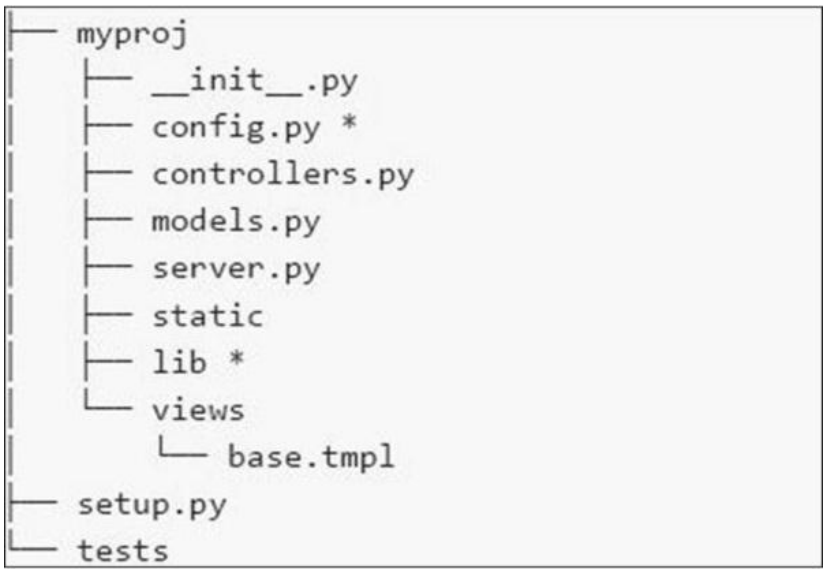

Рис. 3 Пример структуры расположения файлов

Вот краткое описание различных файлов, которые присутствуют в файловой системе

- **config.py** - Каждому приложению нужен файл конфигурации и способ его загрузки. Эта функциональность может быть определена в config.py.
- **controllers.py** - В *файле controllers.py* прописана логика работы базы данных с шаблонами, еще его часто называют шаблоном проектирования MVC.
- **models.py** - этот файл взаимодействует с базой данных напрямую для некоторых служб или для хранения постоянных данных.
- **server.py** - этот файл взаимодействует с готовым к работе веб-сервером, который используется в реальных приложениях.
- **static** - включает в себя все CSS, JS и файлы изображений.
- **views** - включает в себя все файлы шаблонов для данного приложения.

По шагам построим более сложное web-приложение на CherryPy.

Шаг 1 - Создайте каталог, который будет содержать приложение.

Шаг 2 - Внутри каталога создайте файл controllers.py со следующим содержимым

```
import cherrypy

class Root(object):
  def __init__(self, data):
```

```
    self.data = data
  @cherrypy.expose
  def index(self):
    return 'Hi! Welcome to your application'

def main(filename):
  data = {}
   cherrypy.config.update({
    'tools.encode.on': True, 'tools.encode.encoding': 'utf-8',
    'tools.decode.on': True,
    'tools.trailing_slash.on': True,
    'tools.staticdir.root': os.path.abspath(os.path.dirname(__file__)),
  })
  cherrypy.quickstart(Root(data), '/', {
    '/media': {
      'tools.staticdir.on': True,
      'tools.staticdir.dir': 'static'
    }
  })
if __name__ == '__main__':
main(sys.argv[1])
```

Шаг 3 - Рассмотрим приложение, в котором пользователь вводит значение через форму. Включим в приложение две формы - index.html и submit.html.

Шаг 4 - В приведенном выше коде для контроллеров метод есть index (), который является функцией по умолчанию и загружается первым, если вызывается конкретный контроллер.

Шаг 5 - Реализация метода index() может быть изменена следующим образом:

```
@cherrypy.expose
  def index(self):
    tmpl = loader.load('index.html')
    return tmpl.generate(title='Sample').render('html', doctype='html')
```

Шаг 6 - Это загрузит index.html при запуске данного приложения и направит его в указанный поток вывода. Файл index.html выглядит следующим образом:

```
<!DOCTYPE html >
<html>
  <head>
```

```
    <title>Sample</title>
  </head>
  <body class = "index">
    <div id = "header">
      <h1>Sample Application</h1>
    </div>
    <p>Welcome!</p>
    <div id = "footer">
      <hr>
    </div>
  </body>
</html>
```

Шаг 7. Важно добавить метод в класс Root в controller.py, если вы хотите создать форму, которая принимает такие значения, как имена и заголовки.

```
@cherrypy.expose
  def submit(self, cancel = False, **value):
    if cherrypy.request.method == 'POST':
      if cancel:
        raise cherrypy.HTTPRedirect('/') # to cancel the action
      link = Link(**value)
      self.data[link.id] = link
      raise cherrypy.HTTPRedirect('/')
    tmp = loader.load('submit.html')
    streamValue = tmp.generate()

    return streamValue.render('html', doctype='html')
```

Шаг 8 - код, который будет включен в submit.html, выглядит следующим образом

```
<!DOCTYPE html>
  <head>
    <title>Input the new link</title>
  </head>
  <body class = "submit">
    <div id = " header">
      <h1>Submit new link</h1>
    </div>
    <form action = "" method = "post">
      <table summary = "">
```

```
        <tr>
          <th><label for = " username">Your name:</label></th>
          <td><input type = " text" id = " username" name = " username" /></td>
        </tr>
        <tr>
          <th><label for = " url">Link URL:</label></th>
          <td><input type = " text" id=" url" name= " url" /></td>
        </tr>
        <tr>
          <th><label for = " title">Title:</label></th>
          <td><input type = " text" name = " title" /></td>
        </tr>
        <tr>
          <td></td>
          <td>
            <input type = " submit" value = " Submit" />
            <input type = " submit" name = " cancel" value = "Cancel" />
          </td>
        </tr>
      </table>
    </form>
    <div id = "footer">
    </div>
  </body>
</html>
```

Шаг 9 – Посетить URL http://localhost:8080/submit/
Результат можно посмотреть на рисунке 4

Рис. 4 Результат создания сложного web-приложение на CherryPy

Вывод

Не обязательно использовать большие и сложные системы для создания простых и легковесных web-приложений. Достаточно использовать минималистические web-фреймворки такие как CherryPy. Они упрощают порог понимания работы web-сервисов и позволяют в короткие сроки создать мини проект под свои задачи.

Библиографический список

1. Web-фреймворк cherrypy URL: https://cherrypy.org/ (Дата обращения: 10.12.2018
2. Арушанян О.Б., Богомолов Н.А., Ковалев А.Д., Синицын М.Н. Архитектура клиентского программного обеспечения для web-приложений, ориентированных на представление данных // Вычислительные методы и программирование: новые вычислительные технологии 2004. №2 С. 24-37. URL: https://elibrary.ru/item.asp?id=8811001 (Дата обращения: 10.12.2018)
3. Грищенко В.Н. Типовые элементы компонентно-ориентированной разработки web-приложений // Кибернетика и системный анализ 2005. № 3 С. 166-174. URL: https://elibrary.ru/item.asp?id=9281581 (Дата обращения: 10.12.2018)
4. Юдин Д. Г. Поддержание качества web-приложения, написанного на языке php // Научно-технический вестник санкт-петербургского государственного университета информационных технологий, механики и оптики 2008. № 51 С. 93-98. URL: https://elibrary.ru/item.asp?id=11692160 (Дата обращения: 10.12.2018)