Buzzy Documentation

Release 0

Sebastian Pawluś

November 10, 2015

Contents

Low level static page generator, with simple API!

Why to use static sites generator?

There are many cases, when your website will be static and using dynamic pages framework like Django, Ruby on Rails, Flask, Sinatra would be a bit of overhead.

Why to write yet another static website generator?

Not really sure, yet!

CHAPTER 1

Install

Buzzy currently runs only Python 2.7.x and earlier versions of Python are not supported and Python 3 was not tested yet.

You can install it from PyPi, by simply **pip**:

```
$ pip install buzzy
```

A recommended approach would be to create a virtual environment for buzzy project via virtualenv before installing it.

CHAPTER 2

Quick Start

Create a regular python file, copy paste the content presented below.

```
import buzzy

class StaticSite(buzzy.Base):

    @buzzy.register
    def thing(self):
            yield buzzy.render.content("Hello world", "index.html")

if __name__ == "__main__":
    StaticSite()
```

Each render function created with buzzy needs to be decorated with **register**. This way buzzy will know which method in class should be called during the build process.

```
$ python project.py build
2014-03-01 20:54:55,599 - StaticSite - INFO - build generated
```

Now you should have content inside your build directory, which will be called **_build**.

```
$ ls _build
index.html

$ cat _build/index.html
Hello world
```

You should see there one file **index.html**, and the content of this file will be 'Hello world'.

```
$ python project.py server
2014-03-01 20:54:55,599 - StaticSite - INFO - build generated
2014-03-01 20:54:55,600 - StaticSite - INFO - serving at port 8000
```

Go to your browser to http://127.0.0.1:8000/, done!

CHAPTER 3

Renderers

buzzy.render.**content**(*content*, *target_file*)

A renderer class to create a file from a content.

Parameters

- **content** – content to put inside he file
- **target_file** – name of the destination file

```
@buzzy.register
def view(self):
    yield buzzy.render.content("index.html", "hello world")
```

buzzy.render.**template**(*template*, *target_file*, ***context*)

A renderer class to render file from a template. **jinja2** package is required

Parameters

- **template** – jinja2 template located in the **TEMPLATE_DIR**
- **target_file** – name of the destination file
- ****context** – as many named parameters as needed, all will be put as a context inside the template

```
@buzzy.register
def view(self):
    yield buzzy.render.template("index.html", "index.tpl", text="hello world")
```

buzzy.render.**markdown**(*source*, *target_file*)

A renderer class to render file from a markdown markup. **markdown** package is required

Parameters

- **target_file** – name of the destination file
- **source** – for source of the markup file

```
@buzzy.register
def view(self):
    yield buzzy.render.markdown("index.html", "index.md")
```

CHAPTER 4

Settings

- **BUILD_DIR**, *default* = '_build'

  Build directory, where static page will be generated after executing **build** method.

- **INCLUDE**, *default* = []

  List of files and directories that will be copy over to the build directory without any modifications.

- **TEMPLATES_DIR**, *default* = 'templates'

  Templates directory, jinja2 base template directory used with **render.template**.

- **SERVER_PORT**, *default* = '8000'

  Developer server port, from which will page will be server after executing **server** method.

- **WATCH_EXCLUDE**, *default* = ['.git*', '.hg*', '*.orig']

  List of files to be excluded from watch process. When **watch** command is called, the build directory will be reload every time when page got changed. This setting prevents from calling rebuild for some files. The **BUILD_DIR** is will be excluded as well.

CHAPTER 5

Commands

- **build**

  Regenerates the content inside **BUILD_DIR**

- **server**

  Runs developemnt server. It will *watch* development directory, if files inside will get changed it will trigger **build** command.

- Custom command

  By using **@buzzy.command** decorator you can register your own command

```
@buzzy.command
def mycommand(self):
    deploy_site()
```

```
$ python project.py mycommand
```

CHAPTER 6

Why yield

There are three main reasons why to use **yield** here:

- yield is cool, and is overly underrated as python mechanism,
- render function may call yield many times, which means that one function may generate more than one file,
- yield is memory efficient, we are operating here on file contents in memory, yield will reduce some pain here.

CHAPTER 7

Source Code

https://github.com/xando/buzzy

B