plate Documentation

Release 0.2

ash84

August 23, 2016

Contents

Contents:

CHAPTER 1

Plate

1.1 Introduce

Plate is API Documentations Tool based on Markdown(md) . Convert Slate based on Ruby-Middleman to Python-Flask based. And add some different functions for usages.

Example site is plate-project.github.io.

1.2 Features

- **Configuration File(config.json)**
  - Set a title, programming languages for example codes using `config.json` based on JSON Format. Also set the path of the API documents and TOC(Table of contents). Anyone can easily set up.
- **Support Multi-API documents**
  - **'plate<https://github.com/Plate-Project/plate'_** support multiple API documents(multi markdown format files) for efficient management and amount of documents. As you with, use one markdown file or separate markdown files by API or another criterion. Also you can set the output order using TOC(index.json).
- **Support dynamic changes of documents**
  - You can update the changes of API documents without restarting server. When web page refresh, if exist any changes, **'plate<https://github.com/Plate-Project/plate'_** reload API documents. Users only focus on writing API documents.
- **Make Static HTML**
  - Convert Markdown(md) to Static HTML using jinja2 template. Use this on github.io and static html service or offline.
- **Multi-Languages Searching**
  - To **'support searching various languages<http://plate.readthedocs.org/en/latest/advanced.html#multi-language-search>'_** such as Japanese, French, German, etc, use not only **'lunr.js<http://lunrjs.com/>'_** but also **'lunr-languages<https://github.com/MihaiValentin/lunr-languages>'_**.
- **Code Copy**
  - It can be easily copy the example codes without mouse drag and drop, immediately apply this to your codes. Set `CLIPBOARD` in `config.json`, can copy codes using clicking copy link.

Plate is very easy for any developers. First of all, follow below Getting Start. And then you have any problems, immediately notify(email, issue board, anything). Always, plate is ready for you.

CHAPTER 2

Getting Start

2.1 Support Python Version

- **Python, version 2.7 ~ 3.4**

2.2 Prerequisites

- **requirements.txt** have all libraries for running plate
- If you install using `quick-start.py`, automatically install all libraries.
- manually, install all libraries:

```
pip install -r requirements.txt
```

2.3 Quick Start with Server

1. Clone plate to your hard drive with `git clone https://github.com/Plate-Project/plate.git`
2. `cd plate`
3. Install your API document web pages using `quick-start.py` .
4. Start with server: `python plate.py`

```
git clone https://github.com/Plate-Project/plate.git
cd plate
python install.py
...

Welcome plate v0.2.6
Start your API Document system.

Typing API document name :<Typing your project>
what is API document name? is "<your project>"

Rename plate to  "<your project>" ...
Complete. Enjoy developing.
```

```
cd ../<your project>
python plate.py
```

2.4 Quick Start with Static HTML

Start with static html:`python pst.py -f <conf file>`

```
python pst.py -f config.json
```

2.5 Config

- Configuration file for Plate
- path : `./config.json`

KEY	DESCRIPTION
PORT	Server Port
TITLE	<head><title>Title String</title></head>
LOGO_IMG	Image path, Display at left top
LOGO_TITLE	Logo Title, Display at left top, If you set `LOGO_IMG`, do not display LOGO_TITLE.
SEARCH_ON	true or false, if true available searching
SUPPORT_LANG	List Type, Programming Language to display example tabs
API_DOC_PATH	Directory Path include markdown API document files.
API_DOC_INDEX_PATH	JSON format have Markdown file list.
COPYRIGHT	copy right string
FAVICON	favicon file name in `/static/images`
CLIPBOARD	Display copy button below codes
STATIC	Making Static HTML Section
DIR	Directory Path for saving static html.
HTML	Static html file name.

CHAPTER 3

Usage

3.1 Table of Contents

- TOC file for setting documents order and file name.
- Set `API_DOC_INDEX_PATH` in `config.json`.
- path : `./document/index.json`

TOC(Table of content) appears as written order. If index.json path is not equal API document file(md) path, must write file path.

3.2 Logo and Title

- You can select the logo image or title text. not **BOTH** .
- If set `LOGO_IMG` in config.json, display specific logo image at left-top.
- If set `LOGO_TITLE` in config.json, display specific title text at left-top.

If you set both, `LOGO_IMG` only display.

```
{
    "LOGO_TITLE"          : "Management API",
    "LOGO_IMG"            : "logo.png"
}
```

3.3 Emphasis Syntax

- You can the emphasis syntax for using `<aside class="CLASSNAME">`.
- `CLASSNAME` is `success` or `notice` or `warning`.

```
<aside class="warning">
    Must encrypt password using a key.
</aside>
```

```
<aside class="notice">
    No mandatory parameter, return 400 Invalid Parameter
</aside>
```

```
<aside class="success">
    Success, return HTTP Status code 200 OK.
</aside>
```

CHAPTER 4

Internal

4.1 Basic Structure

Internally, Plate have **3 steps** :

- Get API Document
- Convert markdown to HTML
- Highlight code according to programming languages

1. Get API Document

- Read markdown from API Document based on markdown format using `API_DOC_PATH` and `API_DOC_INDEX_PATH` in `config.json`.
- Sort by `index.json` of `API_DOC_INDEX_PATH`.

2. Convert markdown to HTML

- Convert markdown to HTML using markdown python module
- Use markdown extensions : - fence_codes : markdown code block to html pre tag - tables : markdown table syntax to html table tag

```
def conv_md_to_html(md_text):
    import markdown
    return markdown.markdown(md_text, extensions=["fenced_code", "tables"])
```

3. Highlight code according to programming languages

- Use pygemnts for highlighting codes
- Support various programming languages and markup.

```
from pygments import highlight
from pygments.lexers import PythonLexer
from pygments.lexers import JavaLexer
from pygments.formatters import HtmlFormatter

highlighted = highlight(code, PythonLexer(), HtmlFormatter())
highlighted = highlight(code, JavaLexer(), HtmlFormatter())
```

4.2 Realtime monitoring API Document

For monitoring the modification of API Documents, use wachdog . After the server start, the watchdog start and monitor all documents in `API_DOC_PATH` of `config.json` . When the server stop, also the watchdog stop. In this process, use method `watchdocs.watch_api_doc.start_watch` and `watchdocs.watch_api_doc.stop_watch` .

If raise any modification fo files, run `watchdocs.document_trace_handler.on_modified` method. In this method, enqueue a event about the modification of any file to `document_trace_queue` . It is a instance of `DocumentTraceQueue` singleton class.

And then, receive new request from user, Plate check `document_trace_queue` whether a event exist or not. If any event in queue, Plate load all API Documents and convert to html.

CHAPTER 5

Advanced

5.1 Use postman collection json

If use postman2md library, easily convert postman collection json for testing API to markdown. And then you can use converted markdown files as API Documents of Plate. Now must use markdown converting by postman2md, future we will support postman collection json as the subject of plate. Have any interest of postman2md, see https://github.com/Plate-Project/postman2md and welcome your contributions at any time, with issues.

```
import postman2md
# create multi markdown file in the directory.
postman2md.convert(postman_file="example.json.postman_collection")

# create merged markdown file in the directory.
postman2md.convert(postman_file="example.json.postman_collection", multi_file=False)
```

5.2 Multi Language Search

Multi-language Search is very important part in searching. Plate use lunr.js for searching contents in API Documents. But lunr.js has the limit of only english. For available of multi-language search such as Japanese, French, Plate use lunr-languages .

Below is supporting languages :

- German
- French
- Spanish
- Italian
- Japanese
- Dutch
- Danish
- Portuguese
- Finnish
- Romanian
- Hungarian

- Russian
- Norwegian

CHAPTER 6

Modules

6.1 Common

class plate.common.config.**Config**(*result*)
Read JSON Format config file.

static load_conf(*conf_file_path*)
Read .json file

Parameters **conf_file_path** – .json file path

Returns conf instance

plate.common.convmd2html.**convert_md_to_html**(*md_text*)
Convert markdown text to HTML

Parameters **md_text** – markdown text

Returns html

class plate.common.singleton_meta.**SingletonMeta**
Singleton Base Class

Example:

```
class DocumentTraceQueue(object):
    from common import Singleton
    __metaclass__ = Singleton
```

plate.common.syntax_highlighting.**syntax_highlight**(*lang*, *code*)
code highlighting HTML Format

Parameters

- **lang** – programming language
- **code** – code(not html)

Returns highlighted code(html format)

6.2 API Document

class plate.api_document.**APIDocument**(*config=None*)
Making API Document

create_api_docs()
Convert API Document to HTML.

Returns `OrderedDict` instance.

highlight_syntax(*soup*)
Highlight code syntax.

Parameters **soup** – bs4 instance

Returns bs4 instance

modify_html(*soup*)
Modify HTML

Parameters **soup** – bs4 instance

Returns HTML

read_index(*index_file_path*)
Read API Document index file such as `index.json`.

Parameters **index_file_path** – index file path

Returns JSON of index file(`index.json`)

total_reload_docs()
Reload all API Document files.

6.3 Watchdocs

class `plate.watchdocs.document_trace_handler.`**DocumentTraceHandler**(*tracing_files=None*)
`DocumentTraceHandler` is event handler for API Document files.

on_modified(*event*)
Event handler about modified. If raise modify on api document file or index file such as `index.json`, enqueue event to DocumentTraceQueue.

Parameters **event** – the event about event handler

class `plate.watchdocs.document_trace_queue.`**DocumentTraceQueue**
Queue of modification, inserted, deleted document.

clear()
Remove all trace_queue

count()
Count of trace_queue

Returns count

dequeue()
Dequeue event Return the copy of top event in trace_queue

Returns event

enqueue(*event*, *is_index_file*)
Enqueue event to trace_queue

Parameters

- **event** – insert/update/del event
- **is_index_file** – True or False

is_empty()
Is empty trace_queue?

Returns True or False

class plate.watchdocs.api_document_observer.**APIDocumentObserver**(*doc_path=None*, *doc_index_path=None*, *doc_file_path_list=None*)

APIDocumentObserver is observer of API Documents.

is_started
After run start_watch(), is_started is True, or False.

Returns True | False

start_watch()
Start watch docs

stop_watch()
Stop watch docs

class plate.watchdocs.document_trace_file.**DocumentTraceFile**(*tracing_file_path*, *is_index_file=False*)

CHAPTER 7

Contributing

Any suggestions Submit a issue. Show me the pull requests.

CHAPTER 8

Version

Release : 0.2

Version : 0.2.6

8.1 Change Log

- **V0.2.6**
  - Add Test Cases.
- **V0.2.5**
  - Change basic structures.
  - Add unit testing.
- **v0.2.4**
  - Apply Sphinx

CHAPTER 9

License

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

p

A

C

D

E

H

I

L

M

O

P

R

S

T