moban Documentation

Release 0.2.4

Onni Software Ltd.

Jan 14, 2019

Contents

Author C.W.

Issues http://github.com/moremoban/moban/issues

License MIT

Version 0.2.4

Generated Jan 14, 2019

moban brings the high performance template engine (JINJA2) for web into static text generation. It is used in pyexcel project to keep documentation consistent across the documentations of individual libraries.

CHAPTER 1

Installation

You can install it via pip:

```
$ pip install moban
```

or clone it and install it:

```
$ git clone http://github.com/moremoban/moban.git
$ cd moban
$ python setup.py install
```

CHAPTER 2

Quick start

Here is a simple example:

```
$ moban -c data.yml -t my.template
$ cat moban.output
```

Given data.yml as:

```
hello: world
```

and my.template as:

```
{{hello}}
```

moban.output will contain:

```
world
```

the tutorial has more use cases.

CHAPTER 3

Usage

```
usage: moban [-h] [-cd CONFIGURATION_DIR] [-c CONFIGURATION]
             [-td [TEMPLATE_DIR [TEMPLATE_DIR ...]]] [-t TEMPLATE] [-o OUTPUT]
             [-f] [-m MOBANFILE]

Yet another jinja2 cli command for static text generation

optional arguments:
  -h, --help            show this help message and exit
  -cd CONFIGURATION_DIR, --configuration_dir CONFIGURATION_DIR
                        the directory for configuration file lookup
  -c CONFIGURATION, --configuration CONFIGURATION
                        the dictionary file
  -td [TEMPLATE_DIR [TEMPLATE_DIR ...]], --template_dir [TEMPLATE_DIR [TEMPLATE_DIR ..
↪.]]
                        the directories for template file lookup
  -t TEMPLATE, --template TEMPLATE
                        the template file
  -o OUTPUT, --output OUTPUT
                        the output file
  --template_type TEMPLATE_TYPE
                        the template type, default is jinja2
  -f                    force moban to template all files despite of
                        .moban.hashes
  -m MOBANFILE, --mobanfile MOBANFILE
                        custom moban file
```

3.1 exit codes

- 0 : no changes
- 1 : has changes
- 2 : error occured

CHAPTER 4

Built-in Filters

4.1 split_length

It breaks down the given string into a fixed length paragraph. Here is the syntax:

```
{% for line in your_string | split_length(your_line_with) %}
{{line}}
{% endfor %}
```

It is used to keep changelog formatted in CHANGELOG.rst.jjs in pypi-mobans project

4.2 github_expand

It expands simple hashtags into github issues. Here is the syntax:

```
{{ your_github_string | github_expand }}
```

It makes it easy to mention github reference in change log in all projects. Here is the place it is applied: CHANGELOG.rst.jjs in pypi-mobans project

Here is Grammar in the changelog.yml:

```
=============== ==============================
Syntax          Meaning
=============== ==============================
`#1`            moban issues 1
`PR#1`          moban pull request 1
`pyexcel#1`     other project issues 1
`pyexcel#PR#1`  other project pulll request 1
=============== ==============================
```

More details can be found in moban's changelog.yml

4.3 *repr*

Returns a single quoted string in the templated file

CHAPTER 5

Built-in Tests

5.1 *exists*

Test if a file exists or not

CHAPTER 6

Tutorial

Please clone the moban repository as the data mentioned in the tutorial are stored in examples folder.

6.1 Level 1 Jinja2 on command line

moban reads data in yaml format, renders a template file in jinja2 format and outputs it to *moban.output*. By default, it looks for *data.yml* as its data file

6.1.1 Evaluation

Please clone the moban project and install moban:

$ git clone **https://github.com/chfw/moban.git** $ cd moban $ python setup.py install

Then go to *docs/level-1-jinja2-cli*. here are different commands to evaluate it:

```
moban -c data.yml -t a.template
```

'moban.output' is the generated file.

```
moban -c data.yml -t a.template -o my.output
```

-o my.output will override the default name

Note: You may simply type the short form:

```
moban -t a.template
```

because moban looks for *data.yml* by default

6.2 Level 2: template inheritance

Template inheritance is a feature in Jinja2. This example show how it was done. *a.template* inherits *base.jj2*, which is located in *.moban.td*, the default template directory.

6.2.1 Evaluation

Please go to *docs/level-2-template-inheritance*, here is the command to launch it:

```
moban -c data.yaml -t a.template
```

a.template inherits *.moban.td/base.jj2*.

6.3 Level 3: data override

What *moban* bring on the table is data inheritance by introducing *overrides* key word in the yaml file:

```
overrides: data.base.yaml
....
```

And *.moban.cd* is the default directory where the base data file can be placed.

6.3.1 Evaluation

Please change directory to *docs/level-3-data-override* directory.

In this example, *data.yaml* overrides *.moban.cd/data.base.yaml*, here is the command to launch it:

```
moban -c data.yaml -t a.template
```

'a.output' is the generated file:

```
========header============

world

shijie

========footer============
```

6.4 Level 4: single command

If you use moban regularly and operates over a number of files, you may consider write a *.moban.yml*, which is a mini script file that commands *moban* to iterate through a number of files

6.4.1 Evaluation

Please go to *docs/level-4-single-command* directory.

Here is the *.moban.yml*, whihc replaces the command in level 3:

```
targets:
  - a.output: a.template
```

where *targets* should lead an array of dictionaries.

Here is how to launch it .. code-block:: bash

    moban

'a.output' is the generated file:

```
========header============

world

shijie

========footer============
```

6.5 Level 5: custom configuration

With *.moban.yml*, you can even change default data directory *.moban.cd* and default template directory *.moan.td*. Read this example:

```
configuration:
  configuration_dir: 'custom-config'
  template_dir:
    - custom-templates
    - cool-templates
    - '.'
targets:
  - a.output: a.template
```

where *configuration* lead a dictionary of key words:

1. *configuration_dir* - the new configuration directory
2. *template_dir* - an array of template directories

6.5.1 Evaluation

Please go to *docs/level-5-custom-configuration* directory.

Here is the command to launch it:

```
moban
```

'a.output' is the generated file:

```
========header============

world

shijie

```

(continues on next page)

(continued from previous page)

```
this demonstrations jinja2's include statement

========footer============
```

6.6 Level 6: Complex Configuration

On top of level 5, you could have a common template, where data and output change. In the following example:

```
configuration:
  configuration_dir: 'custom-config'
  template_dir:
    - custom-templates
    - cool-templates
    - '.'
  template: a.template
targets:
  - output: a.output
    configuration: data.yml
  - output: a.output2
    configuration: data2.yml
```

where *template* under *confiugration* needs a template file, which will be a default template across *targets*. And in this example, the expand form of *targets* is illustrated:

{ "output": 'an output file', "configuration": 'data file', "template": "the template file"

}

6.6.1 Evaluation

Please go to *docs/level-6-complex-configuration* directory.

Here is the command to launch it:

```
moban
```

'a.output' is the generated file:

```
========header============

world

shijie

this demonstrations jinja2's include statement

========footer============
```

a.output2 is:

```
========header============

world2
```

(continues on next page)

(continued from previous page)

```
shijie

this demonstrations jinja2's include statement

========footer============
```

6.7 Level 7: Custom jinja filters, tests and globals

Level 7 example demonstrates advanced plugin capabilities of moban. The following moban file had *plugin_dir* specified:

```
configuration:
  template_dir:
    - my-templates
  plugin_dir:
    - custom-jj2-plugin
  configuration: data.yml
targets:
  - filter.output: filter.jj2
  - test.output: test.jj2
```

Where *custom-jj2-plugin* is a directory holding all jinja2 filters, tests and globals. Under it, there are 4 files:

```
__init__.py     filter.py       test.py     global.py
```

It is very important to have *__init__.py*, otherwise, it will NOT work. Other three files are named to show case the feature. You can choose whichever name you prefer, as long as you and your team could make sense of the names.

6.7.1 Evaluation

Please go to *docs/level-7-use-custom-jinja2-filter-test-n-global* directory,

Here is the command to launch it:

```
$ moban
Templating filter.jj2 to filter.output
Templating test.jj2 to test.output
Templating global.jj2 to global.output
Templated 3 files.
Everything is up to date!
```

Please examine individual template and its associated plugin for more details.

In pratice, the following use cases were found interesting to go along with.

6.8 Misc 1: copying templates

With *.moban.yml*, you can copy templates to your destination.

Please be aware that, your templates and template folder have to be inside declared template folders. It does not copy any file or folder.

Here is example moban file for copying:

```
configuration:
  template_dir:
    - template-sources
copy:
  - simple.file.copy: file-in-template-sources-folder.txt
  - "misc-1-copying/can-create-folder/if-not-exists.txt": file-in-template-sources-
↪folder.txt
  - "test-dir": dir-for-copying
  - "test-recursive-dir": dir-for-recusive-copying/**
```

template copy does:

1. copies any template inside pre-declared template directory to anywhere. moban will

   create directory if needed.

1. copies any directory to anywhere. If "**" is followed, moban attempts to do recursive copying.

For more complex use case, please look at its usage in pyexcel project

CHAPTER 7

Developer Guide

7.1 Development guide

7.1.1 Jinja2 extensions for Moban

Since version 0.2, mobanfile supports an extra field *plugin_dir*, along with *template_dir*. When you put your own jinja2 filters, tests and globals in your moban repo, you can let moban know about them via this keyword.

Importantly, you have to have *__init__.py* file in your *plugin_dir*. Otherwise, your plugins will NOT be loaded.

Jinja2 Filter

```
from moban.extensions import JinjaFilter


@JinjaFilter()
def repr(string):
    if isinstance(string, list):
        return ["'{0}'".format(str(element)) for element in string]
    else:
        return "'{0}'".format(str(string))
```

Jinja2 Test

```
from os.path import isdir, isfile, isabs, exists
from os.path import lexists, islink, samefile, ismount

from moban.extensions import jinja_tests


```

(continues on next page)

(continued from previous page)

```
jinja_tests(
    is_dir=isdir,
    directory=isdir,
    is_file=isfile,
    file=isfile,
    is_link=islink,
    link=islink,
    exists=exists,
    link_exists=lexists,
    # path testing
    is_abs=isabs,
    abs=isabs,
    is_same_file=samefile,
    same_file=samefile,
    is_mount=ismount,
    mount=ismount,
)
```

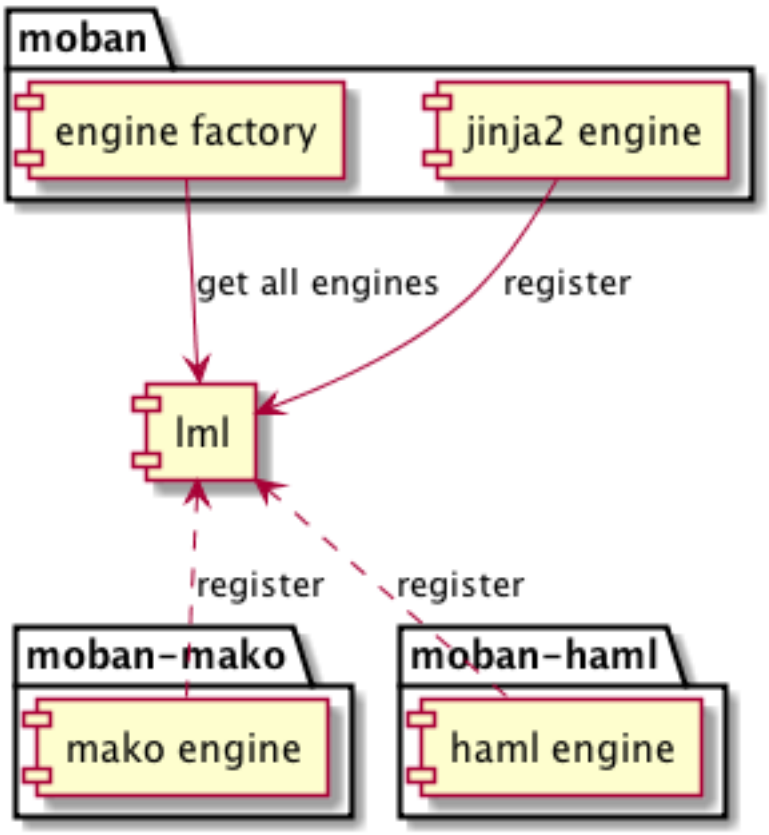
Jinja2 Globals

```
def test_globals():
    output = "globals.txt"
    test_dict = dict(hello="world")
    jinja_global("test", test_dict)
    path = os.path.join("tests", "fixtures", "globals")
    engine = Engine([path], path)
    engine.render_to_file("basic.template", "basic.yml", output)
    with open(output, "r") as output_file:
        content = output_file.read()
        eq_(content, "world\n\ntest")
    os.unlink(output)
```

It is possible to write an installable package including your own jinja2 filters, tests and globals. Please email me for more details.

7.1.2 Template engine extension for Moban

moban version 0.2 started using lml to employ loose couple plugins. Other template engines, such as marko, haml can be plugged into moban seamless.



In order plugin other template engines, it is to write a lml plugin. The following is an example starting point for any template engine.

```
from lml.plugin import PluginInfo

from moban.constants import TEMPLATE_ENGINE_EXTENSION


@PluginInfo(TEMPLATE_ENGINE_EXTENSION, tags=["mako"])
class MakoEngine:
    pass
```

After you will have finished the engine plugin, you can either place it in *plugin_dir* in order to get it loaded, or make an installable python package. In the latter case, please refer to yehua: doing that in less than 5 minutes.

CHAPTER 8

Change log

8.1 0.2.4 - 14-07-2018

8.1.1 Added

1. #32: option 1 copy a directory without its subdirectories.
2. #30: command line template option is ignore when a moban file is present

8.2 0.2.3 - 10-07-2018

8.2.1 Added

1. #76: running moban as a module from python command
2. #32: copy a directory recusively
3. #33: template all files in a directory

8.3 0.2.2 - 16-06-2018

8.3.1 Added

1. #31: create directory if missing during copying

8.3.2 Updated

1. #28: if a template has been copied once before, it is skipped in the next moban call

8.4 0.2.1 - 13-06-2018

8.4.1 Updated

1. templates using the same template engine will be templated as a group
2. update lml dependency to 0.0.3

8.5 0.2.0 - 11-06-2018

8.5.1 Added

1. #18: file exists test
2. #23: custom jinja plugins
3. #26: repr filter
4. #47: allow the expansion of template engine
5. #58: allow template type per template

8.5.2 Updated

1. #34: fix plural message if single file is processed

8.6 0.1.4 - 29-May-2018

8.6.1 Updated

1. #21: targets become optional
2. #19: transfer symlink's target file's file permission under unix/linux systems
3. #16: introduce copy key word in mobanfile

8.7 0.1.3 - 12-Mar-2018

8.7.1 Updated

1. handle unicode on python 2

8.8 0.1.2 - 10-Jan-2018

8.8.1 Added

1. #13: strip off new lines in the templated file

8.9 0.1.1 - 08-Jan-2018

8.9.1 Added

1. the ability to present a long text as multi-line paragraph with a custom upper limit
2. speical filter expand github references: pull request and issues
3. #15: fix templating syntax to enable python 2.6

8.10 0.1.0 - 19-Dec-2017

8.10.1 Added

1. #14, provide shell exit code

8.11 0.0.9 - 24-Nov-2017

8.11.1 Added

1. #11, recognize .moban.yaml as well as .moban.yml.
2. #9, preserve file permissions of the source template.
3. *-m* option is added to allow you to specify a custom moban file. kinda related to issue 11.

8.11.2 Updated

1. use explicit version name: *moban_file_spec_version* so that *version* can be used by users. #10 Please note: moban_file_spec_version is reserved for future file spec upgrade. For now, all files are assumed to be '1.0'. When there comes a new version i.e. 2.0, new moban file based on 2.0 will have to include 'moban_file_spec_version: 2.0'

8.12 0.0.8 - 18-Nov-2017

8.12.1 Added

1. #8, verify the existence of custom template and configuration directories. default .moban.td, .moban.cd are ignored if they do not exist.

8.12.2 Updated

1. Colorize error messages and processing messages. crayons become a dependency.

8.13 0.0.7 - 19-Jul-2017

8.13.1 Added

1. Bring the visibility of environment variable into jinja2 templating process: #7

8.14 0.0.6 - 16-Jun-2017

8.14.1 Added

1. added '-f' flag to force moban to template all files despite of .moban.hashes

8.14.2 Updated

1. moban will not template target file in the situation where the changes occured in target file than in the source: the template file + the data configuration after moban has been applied. This new release will remove the change during mobanization process.

8.15 0.0.5 - 17-Mar-2017

8.15.1 Added

1. Create a default hash store when processing a moban file. It will save unnecessary file write to the disc if the rendered content is not changed.
2. Added summary reports

8.16 0.0.4 - 11-May-2016

8.16.1 Updated

1. Bug fix #5, should detect duplicated targets in *.moban.yml* file.

8.17 0.0.3 - 09-May-2016

8.17.1 Updated

1. Bug fix #4, keep trailing new lines

8.18 0.0.2 - 27-Apr-2016

8.18.1 Updated

1. Bug fix #1, failed to save utf-8 characters

8.19 0.0.1 - 23-Mar-2016

8.19.1 Added

1. Initial release

CHAPTER 9

Indices and tables

- genindex
- modindex
- search