moban

Release 0.4.3

Mar 17, 2019

Contents

Author C.W.

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

License MIT

Version 0.4.3

Generated Mar 17, 2019

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

Our vision is: any template, any data. Our current architecture enables moban to plugin any python template engine: mako, handlebars, velocity, haml, slim and tornado and to plugin any data format: json and yaml. Please look at our issues. We have many more template engines and data format on the road map.

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

```
$ export HELLO="world"
$ moban "{{HELLO}}"
Warning: Both data.yml and /.../.moban.cd/data.yml does not exist
Warning: Attempting to use environment vars as data...
Templating {{HELLO}}... to moban.output
Templated 1 file.
$ cat moban.output
world
```

Or simply

```
$ HELLO="world" moban "{{HELLO}}"
```

A bit formal 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
```

Please note that data.yml will take precedence over environment variables.

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]
             [template]

Yet another jinja2 cli command for static text generation

positional arguments:
  template              string templates

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. if not present, moban
                        will try to use environment vars as data
  -td [TEMPLATE_DIR [TEMPLATE_DIR ...]], --template_dir [TEMPLATE_DIR [TEMPLATE_DIR ..
↪.]]
                        the directories for template file lookup
  -t TEMPLATE, --template TEMPLATE
                        the template file. this overrides any targets
                        defined in a custom moban 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
  --exit-code           tell moban to change exit code
  -m MOBANFILE, --mobanfile MOBANFILE
                        custom moban file
```

3.1 Exit codes

By default:

- 0 : no error
- 1 : error occured

With *–exit-code*:

- 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, but it will fallback to environment variables if a data file cannot be found

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* brings 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
```

'moban.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*, which 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.

6.8 Level 8: Pass a folder full of templates

We already know that in moban file, you can pass on a dictionary in targets section, and it apply the template. The assumption was that the template parameter is a file. Now, what if the parameter is a directory?

When you pass a directory with full of templates, moban will also assume the target is a directory and will generate the output there. When saving the files, it will remove its file suffices automatically.

6.9 level 9: moban dependency as pypi package

Why not enable template reuse? Once a template is written somewhere by somebody, as long as it is good and useful, it is always to reuse it, isn't it? DRY principle kicks in.

Now with moban, it is possible to package up your mobans/templates into a pypi package and distribute it to the world of moban.

Here are the sample file:

```
requires:
   - pypi-mobans
configuration:
  template_dir:
    - "pypi-mobans:templates"
  configuration: config.yml
targets:
  - mytravis.yml: travis.yml.jj2
  - test.txt: demo.txt.jj2
```

where *requires* lead to a list of pypi packages. The short syntax is:

```
requires:
  - python-package-name
```

When you refer to it in configuration section, here is the syntax:

```
configuration:
  - template_dir:
    - "python-package-name:relative-folder-inside-the-package"
```

Note: when you do not have relative directory, please keep semi-colon:

```
configuration:
  template_dir:
    - "python-package-name:"
```

6.9.1 Alternative syntax

The alternative syntax is:

```
requires:
   - type: pypi
     name: pypi-mobans
...
```

6.10 level 10: moban dependency as git repo

Since the support to have a pypi package as dependency, the moban pro user will find it more useful to have git repo so that the changes to static content could get propagate as it happens using git push and git pull.

For now, github.com, gitlab.com and bitbucket.com are supported. Pull request is welcome to add or improve this feature.

Here are the sample file:

```
requires:
   - https://github.com/moremoban/pypi-mobans
configuration:
  template_dir:
    - "pypi-mobans:templates"
    - local
  configuration: config.yml
targets:
  - mytravis.yml: travis.yml.jj2
  - test.txt: demo.txt.jj2
```

where *requires* lead to a list of pypi packages. And when you refer to it, as in level-9 section, please use "pypi-mobans:"

6.10.1 Alternative syntax when submodule exists

The alternative syntax is:

```
requires:
   - type: git
     url: https://github.com/your-git-url
     submodule: true
     branch: your_choice_or_default_branch_if_not_specified
...
```

6.11 Level 11: use handlebars

moban is extensible via lml. Charlie Liu through Google Code-in 2018 has kindly contributed moban-handlebars plugin.

6.11.1 Evaluation

Please go to *docs/level-11-use-handlebars* directory. You will have to:

```
$ pip install moban-handlebars
```

Here is the *.moban.yml*, which replaces *jj2* with handlebars files in level 4:

```
targets:
  - a.output: a.template.handlebars
  - b.output: base.hbs
```

where *targets* should lead an array of dictionaries, *requires* installs moban-handlebars extension. You can provide file suffixes: ".handlebars" or ".hbs" to your handlebars template.

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

   moban

6.12 Level 12: use template engine extensions

jinja2 comes with a lot of extensions. In order not to be the blocker in the middle, **extensions** is allowed in moban file to initialize jinja2 engine with desired extensions. Two extensions, expression-statement and loop-controls are enabled by default.

The extensions syntax is:

```
extensions:
  template_type:
    - template.engine.specific.extension
```

For example:

```
extensions:
  jinja2:
    - jinja2.ext.i18n
```

Please also note that the following extensions are included by default: *jinja2.ext.do*, *jinja2.ext.loopcontrols*

6.12.1 Evaluation

Please go to *docs/level-12-use-template-engine-extensions* directory.

If you notice the file *a.template*, we are using a for loop control. This is because moban comes with two default extensions loop-controls and expression-statement.

Now, let us try to use the extension *with*. To do that, we have to enable the extension in the *.moban.yml* file following the above syntax. Now, the extension can be used in the jinja2 templates. One such example is shown in the *b.template* file.

Note: For some extensions, you may need to define *template environment parameters*. In that case, you can take help of our *user defined template types* feature. Please read level-18 for more info. We have explained it using an example here.

Let us consider the example of *jinja2_time*. If you want to use *datetime_format* attribute, you need to specify the same using environmental parameters, *i.e env.datetime_format = '%a, %d %b %Y %H:%M:%S'*. In order to do this, you can specify *datetime_format* using environmental parameters, something like:

```
configuration:
  template_types:
    my_own_type:
      base_type: jinja2
      file_extensions:
        - file_type_of_my_choice
      options:
        datetime_format: %a, %d %b %Y %H:%M:%S
        extensions:
          - jinja2_time.TimeExtension
targets:
  - a.output: a.template.file_type_of_my_choice
```

6.13 Level 13: any data override any data

It's thought that why shall we constrain ourselves on yaml file format. Along the development path, json file format was added. What about other file formats?

By default yaml, json is supported. Due to the new capability *overrides* key word can override any supported data format:

```
overrides: data.base.json
....
```

or simple use *.json* data instead of *.yaml* data.

6.13.1 Evaluation

Please change directory to *docs/level-13-any-data-override-any-data* directory.

In this example, *child.yaml* overrides *.moban.cd/parent.json*, here is the command to launch it:

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

'moban.output' is the generated file:

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

world from child.yaml

shijie from parent.json

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

And we can try *child.json*, which you can guess, overrides *.moban.cd/parent.yaml*

```
moban -c child.json -t a.template
```

'moban.output' is the generated file:

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

world from child.json

shijie from parent.yml

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

6.14 Level 14: custom data loader

Continuing from level 13, *moban* since v0.4.0 allows data loader extension. Due to the new capability *overrides* key word can override any data format:

```
overrides: yours.custom
....
```

or simple use *.custom* data instead of *.yaml* data.

However, you will need to provide a data loader for *.custom* yourselves.

6.14.1 Evaluation

Please change directory to *docs/level-14-custom-data-loader* directory.

In this tutorial, a custom data loader was provided to show case its dataloader extension. Here is the mobanfile:

```
configuration:
  plugin_dir:
    - custom-data-loader
  template: a.template
targets:
  - output: a.output
    configuration: child.custom
  - output: b.output
    configuration: override_custom.yaml
```

custom-data-loader is a directory where custom.py lives. The protocol is that the custom loader register itself to a file extension and return a data dictionary confirming mobanfile schema. On call, *moban* will provide an absolute file name for your loader to work on.

Here is the code to do the registration:

```
@PluginInfo(constants.DATA_LOADER_EXTENSION, tags=["custom"])
```

In order to evaluate, you can simply type:

```
$ moban
$ cat a.output
========header============

world from child.cusom

shijie from parent.json

========footer============
$ cat b.output
========header============

world from override_custom.yaml

shijie from parent.custom

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

Warning: Python 2 dictates the existence of __init__.py in the plugin directory. Otheriwse your plugin won't load

6.15 Level 15: template copying becomes an action plugin in targets

With *.moban.yml*, you can copy templates to your destination. More information is documented in *misc-1-copying-template*.

Explicit syntax:

```
targets:
   - output: explicit
     template: template_file
     template_type: copy
```

Implicit syntax:

```
targets:
   - output: explicit
     template: template_file.copy
```

Shorthand syntax:

```
targets:
   - explicit: template_file.copy
```

No implicit nor short hand syntax for the following directory copying unless you take a look at *force-template-type*. When you read *level-17-force-template-type-from-moban-file/README.rst*, you will find out more.

Directory copying syntax:

```
targets:
  - output: dest-dir
    template: source-dir
    template_type: copy
```

Recursive directory copying syntax:

```
targets:
  - output: dest-dir
    template: source-dir/**
    template_type: copy
```

6.15.1 Evaluation

Here is example moban file for copying:

```
configuration:
  template_dir:
    - template-sources
targets:
  - output: simple.file.copy
    template: file-in-template-sources-folder.txt
    template_type: copy
  - output: target_without_template_type
    template: file_extension_will_trigger.copy
  - target_in_short_form: as_long_as_this_one_has.copy
  - output: "misc-1-copying/can-create-folder/if-not-exists.txt"
    template: file-in-template-sources-folder.txt
```

(continues on next page)

(continued from previous page)

```
    template_type: copy
  - output: "test-dir"
    template: dir-for-copying
    template_type: copy
  - output: "test-recursive-dir"
    template: dir-for-recusive-copying/**
    template_type: copy
```

template copy does:

1. copies any template inside pre-declared template directory to anywhere. moban will create directory if needed.
2. copies any directory to anywhere. If "**" is followed, moban attempts to do recursive copying.

6.16 Level 16: group targets by their template type

Since moban version 0.4.0, you can group your targets with their template type. For example, with *copy* target, you can do the following things:

Here is example moban file for copying:

```
configuration:
  template_dir:
    - template-sources
targets:
  - 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/**
```

More information is documented in *misc-1-copying-template*.

template copy does:

1. copies any template inside pre-declared template directory to anywhere. moban will create directory if needed.
2. copies any directory to anywhere. If "**" is followed, moban attempts to do recursive copying.

Note: The suffix *.copy* of *simple.file.copy* will be removed.

6.17 Level 17: force template type

Since moban version 0.4.0, you can enforce all targets to use one and only one template type, regardless of their individual template types.

Here is example moban file for copying:

```
configuration:
  template_dir:
    - template-sources
  force_template_type: copy
```

(continues on next page)

(continued from previous page)

```
targets:
  - 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/**
```

More information is documented in *misc-1-copying-template*.

template copy does:

1. copies any template inside pre-declared template directory to anywhere. moban will create directory if needed.
2. copies any directory to anywhere. If "**" is followed, moban attempts to do recursive copying.

6.18 Level 18: User defined template types

Since moban version 4.1, custom template types can be defined to deviate from default configurations of the template engines. In addition, the configuration possibilities are:

1. associate your own file extensions
2. choose your own template engine extensions

6.18.1 Evaluation

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

Here is the *.moban.yml*, which inserts *template_types* on top of the moban file found in level 4:

```
configuration:
  template_types:
    my_own_type:
      base_type: jinja2
      file_extensions:
        - file_type_of_my_choice
      options:
        extensions:
          - jinja2_time.TimeExtension
targets:
  - a.output: a.template.file_type_of_my_choice
```

where *template_types* is a dictionary of different custom types.

Also, you can define your *template* on the fly by putting the template parameters inside targets. One such example is:

```
targets:
  - output: b.output
    template: a.template.jj2
    template_type:
      base_type: jinja2
      options:
        block_end_string: '*))'
        block_start_string: '((*'
        variable_start_string: '((('
        variable_end_string: ')))'
```

6.19 Level 19: select a group target to run

Since moban version 0.4.2, you can select a group target to run. For example, with *copy* target mixed with normal file list:

configuration:

template_dir:

- template-sources

targets:

- a.output: a.template.jj2
- 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/**

you can do the following things:

```
$ moban -g copy
```

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

Jinja2 Test

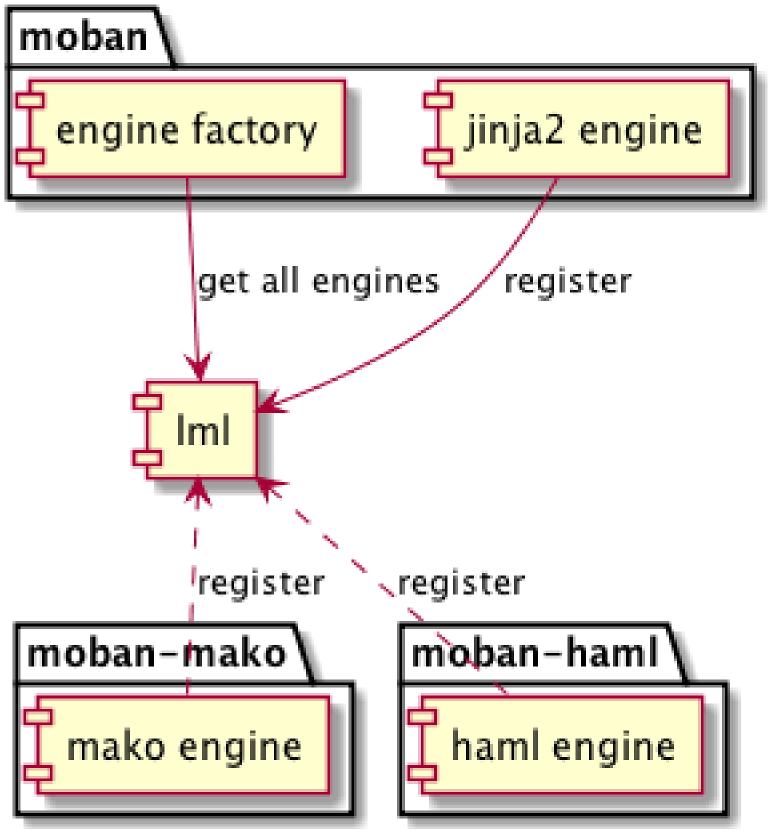
Jinja2 Globals

```
    def __init__(self, template_dirs, extensions=None):
        pass


@patch("lml.plugin.PluginManager.load_me_now", return_value=FakeEngine)
def test_default_mako_type(_):  # fake mako
    engine = ENGINES.get_engine("fake", [], "")
    assert engine.engine.__class__ == FakeEngine


@raises(exceptions.NoThirdPartyEngine)
def test_unknown_template_type():
    ENGINES.get_engine("unknown_template_type", [], "")
```

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.

```
@PluginInfo(
    constants.TEMPLATE_ENGINE_EXTENSION, tags=["file", "extensions", "for", "your",
↪"template"]
)
class Engine(object):
    def __init__(self, template_dirs):
        """
        A list template directories will be given to your engine class
        """

    def get_template(self, template_file):
        """
        Given a relative path to your template file, please return a templatable␣
↪thing that does
        the templating function in next function below
        """

    def apply_template(self, template, data, output):
         """
```

(continues on next page)

(continued from previous page)

```
        Given the template object from `get_template` function, and data as python␣
↪dictionary,
        and output as intended output file, please return "utf-8" encoded string.
        """
```

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 0.4.3 - 16.03.2019

8.1.1 Removed

1. #253: symbolic link in regression pack causes python setup.py to do recursive include

8.1.2 Added

1. #209: Alert moban user when *git* is not available and is used.

8.1.3 Updated

1. #261: since moban group template files per template type, this fill use first come first register to order moban group

8.2 0.4.2 - 08.03.2019

8.2.1 Added

1. #234: Define template parameters on the fly inside *targets* section
2. #62: select a group target to run

8.2.2 Updated

1. #180: No longer two statistics will be shown in v0.4.x. legacy copy targets are injected into a normal targets. cli target is made a clear priority.

2. #244: version 0.4.2 is first version which would work perfectly on windows since 17 Nov 2018. Note that: file permissions are not used on windows. Why the date? because samefile is not avaiable on windows, causing unit tests to fail hence it lead to my conclusion that moban version between 17 Nov 2018 and March 2019 wont work well on Windows.

8.3 0.4.1 - 28.02.2019

8.3.1 Added

1. #235: user defined template types so that custom file extensions, template configurations can be controlled by moban user
2. #232: the package dependencies have been fine tuning to lower versions, most of them are dated back to 2017.

8.4 0.4.0 - 20.02.2019

8.4.1 Added

1. #165: Copy as plugins

8.4.2 Updated

1. #219: git clone depth set to 2
2. #186: lowest dependecy on ruamel.yaml is 0.15.5, Jun 2017

8.5 0.3.10 - 03.02.2019

8.5.1 Added

1. #174: Store git cache in XDG_CACHE_DIR
2. #107: Add -v to show current moban version
3. #164: support additional data formats

8.5.2 Updated

1. #178: UnboundLocalError: local variable 'target' referenced before assignment
2. #169: uses GitPython instead of barebone git commands

8.6 0.3.9 - 18-1-2019

8.6.1 Updated

1. #90: allow adding extra jinja2 extensions. *jinja2.ext.do*, *jinja2.ext.loopcontrols* are included by default. what's more, any other template enigne are eligible for extension additions.
2. #158: Empty file base_engine.py is finally removed

8.7 0.3.8 - 12-1-2019

8.7.1 Updated

1. #141: disable file permissions copy feature and not to check file permission changes on windows.
2. #154: introduce first ever positional argument for string base template.
3. #157: the exit code behavior changed. for backward compactibility please use –exit-code. Otherwise, moban will not tell if there is any changes.

8.8 0.3.7 - 6-1-2019

8.8.1 Updated

1. #146: added a low-setup usage mode via environment variables to moban
2. #148: include test related files in the package for package validation when distributing via linux system, i.e. OpenSuse

8.9 0.3.6 - 30-12-2018

8.9.1 Updated

1. #143: moban shall report permission error and continue the rest of the copying task.
2. #122: Since 0.3.6, moban is tested on windows and macos too, using azure build pipelines. It is already tested extensively on travis-ci on linux os.

8.10 0.3.5 - 10-12-2018

8.10.1 Updated

1. #37: moban will report line number where the value is empty and the name of mobanfile. Switch from pyyaml to ruamel.yaml.

8.11 0.3.4.1 - 28-11-2018

8.11.1 Updated

1. #137: missing contributors.rst file

8.12 0.3.4 - 18-11-2018

8.12.1 Added

1. global variables to store the target and template file names in the jinja2 engine
2. moban-handlebars is tested to work well with this version and above

8.12.2 Updated

1. Template engine interface has been clarified and documented

8.13 0.3.3 - 05-11-2018

8.13.1 Added

1. alternative and expanded syntax for requires, so as to accomendate github submodule recursive

8.14 0.3.2 - 04-11-2018

8.14.1 Added

1. *requires* shall support configuration dirs. In other words, configuration file could be stored in python package or git repository.

8.15 0.3.1 - 02-11-2018

8.15.1 Added

1. #97: requires will clone a repo if given. Note: only github, gitlab, bitbucket for now

8.16 0.3.0 - 27-18-2018

8.16.1 Added

1. #89: Install pypi-hosted mobans through requires syntax

8.16.2 Updated

1. #96: Fix for FileNotFoundError for plugins
2. various documentation updates

8.16.3 Removed

1. #88: removed python 2.6 support
2. removed python 3.3 support

8.17 0.2.4 - 14-07-2018

8.17.1 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

8.18 0.2.3 - 10-07-2018

8.18.1 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

8.19 0.2.2 - 16-06-2018

8.19.1 Added

1. #31: create directory if missing during copying

8.19.2 Updated

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

8.20 0.2.1 - 13-06-2018

8.20.1 Updated

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

8.21 0.2.0 - 11-06-2018

8.21.1 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

8.21.2 Updated

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

8.22 0.1.4 - 29-May-2018

8.22.1 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

8.23 0.1.3 - 12-Mar-2018

8.23.1 Updated

1. handle unicode on python 2

8.24 0.1.2 - 10-Jan-2018

8.24.1 Added

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

8.25 0.1.1 - 08-Jan-2018

8.25.1 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

8.26 0.1.0 - 19-Dec-2017

8.26.1 Added

1. #14, provide shell exit code

8.27 0.0.9 - 24-Nov-2017

8.27.1 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.

8.27.2 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'

8.28 0.0.8 - 18-Nov-2017

8.28.1 Added

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

8.28.2 Updated

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

8.29 0.0.7 - 19-Jul-2017

8.29.1 Added

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

8.30 0.0.6 - 16-Jun-2017

8.30.1 Added

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

8.30.2 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.

8.31 0.0.5 - 17-Mar-2017

8.31.1 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

8.32 0.0.4 - 11-May-2016

8.32.1 Updated

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

8.33 0.0.3 - 09-May-2016

8.33.1 Updated

1. Bug fix #4, keep trailing new lines

8.34 0.0.2 - 27-Apr-2016

8.34.1 Updated

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

8.35 0.0.1 - 23-Mar-2016

8.35.1 Added

1. Initial release

CHAPTER 9

Indices and tables

- genindex
- modindex
- search