moban

Release 0.7.3

May 02, 2020

Contents

1 Announcement 3

2 Introduction 5

3 Vision 7

4 Support 9

5 Credit 11

6 Installation 13

7 Quick start 15

8 Templates and configuration files over HTTP(S) 17

9 Templates and configuration files in a git repo 19

10 Templates and configuration files in a python package 21

11 Work with S3 and other cloud based file systems 23

12 CLI documentation 25

13 Tutorial 27

14 Developer Guide 45

15 Change log 49

16 Migration Notes 59

17 Indices and tables 61

Author C.W. and its contributors (See contributors.rst)

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

License MIT

CHAPTER 1

Announcement

For existing moban users, python 2 support has been dropped. Please stay with versions lower than 0.7.0 if you are still using python 2.

From 2020 onwards, minimum requirement is Python 3.6

CHAPTER 2

Introduction

moban started with bringing the high performance template engine (JINJA2) for web into static text generation.

moban can use other python template engine: mako, handlebars, velocity, haml, slim and tornado, can read other data format: json and yaml, and can access both template file and configuration file in any location: zip, git, pypi package, s3, etc.

It has been used in pyexcel and coala project to keep documentation consistent across the documentations of individual libraries in the same organisation.

And here is a list of other usages:

1. Django Mobans, templates for django, docker etc.
2. Math Sheets, generate custom math sheets in pdf

CHAPTER 3

Vision

Any template, any data in any location

CHAPTER 4

Support

If you like moban, please support me on, patreon or bounty source to maintain the project and develop it further.

With your financial support, I will be able to invest a little bit more time in coding, documentation and writing interesting extensions.

CHAPTER 5

Credit

jinja2-fsloader is the key component to enable PyFilesystem2 support in moban v0.6x. Please show your stars there too!

CHAPTER 6

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 7

Quick start

```
$ export HELLO="world"
$ moban "{{HELLO}}"
Templating {{HELLO}}... to moban.output
Templated 1 file.
$ cat moban.output
world
```

Or

```
$ export HELLO="world"
$ echo "{{HELLO}}" | moban
```

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.

Moban in live action:

All use cases are documented here

CHAPTER 8

Templates and configuration files over HTTP(S)

httpfs should be installed first.

```
$ moban -t 'https://raw.githubusercontent.com/moremoban/pypi-mobans/dev/templates/_
↪version.py.jj2'\
  -c 'https://raw.githubusercontent.com/moremoban/pypi-mobans/dev/config/data.yml'\
  -o _version.py
```

In an edge case, if github repo's public url is given, this github repo shall not have sub repos. This library will fail to translate sub-repo as url. No magic.

CHAPTER 9

Templates and configuration files in a git repo

gitfs2 is optional since v0.7.0 but was installed by default since v0.6.1

You can do the following with moban:

```
$ moban -t 'git://github.com/moremoban/pypi-mobans.git!/templates/_version.py.jj2' \
        -c 'git://github.com/moremoban/pypi-mobans.git!/config/data.yml' \
        -o _version.py
Info: Found repo in /Users/jaska/Library/Caches/gitfs2/repos/pypi-mobans
Templating git://github.com/moremoban/pypi-mobans.git!/templates/_version.py.jj2 to _
↪version.py
Templated 1 file.
$ cat _version.py
__version__ = "0.1.1rc3"
__author__ = "C.W."
```

CHAPTER 10

Templates and configuration files in a python package

pypifs is optional since v0.7.0 but was installed by default since v0.6.1

You can do the following with moban:

```
$ moban -t 'pypi://pypi-mobans-pkg/resources/templates/_version.py.jj2' \
        -c 'pypi://pypi-mobans-pkg/resources/config/data.yml' \
        -o _version.py
Collecting pypi-mobans-pkg
....
Installing collected packages: pypi-mobans-pkg
Successfully installed pypi-mobans-pkg-0.0.7
Templating pypi://pypi-mobans-pkg/resources/templates/_version.py.jj2 to _version.py
Templated 1 file.
$ cat _version.py
__version__ = "0.1.1rc3"
__author__ = "C.W."
```

CHAPTER 11

Work with S3 and other cloud based file systems

Please install fs-s3fs:

```
$ pip install fs-s3fs
```

Then you can access your files in s3 bucket:

```
$ moban -c s3://${client_id}:${client_secrect}@moremoban/s3data.yml \
        -o 'zip://my.zip!/moban.output' {{hello}}
$ unzip my.zip
$ cat moban.output
world
```

Where the configuration sits in a s3 bucket, the output is a file in a zip. The content of s3data.yaml is:

```
hello: world
```

CHAPTER 12

CLI documentation

```
usage: moban [-h] [-c CONFIGURATION] [-t TEMPLATE] [-o OUTPUT]
             [-td [TEMPLATE_DIR [TEMPLATE_DIR ...]]] [-cd CONFIGURATION_DIR]
             [-m MOBANFILE] [-g GROUP] [--template-type TEMPLATE_TYPE]
             [-d DEFINE [DEFINE ...]] [-e EXTENSION [EXTENSION ...]] [-f]
             [--exit-code] [-V] [-v]
             [template]

Static text generator using any template, any data and any location.

positional arguments:
  template              string templates

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIGURATION, --configuration CONFIGURATION
                        the data file
  -t TEMPLATE, --template TEMPLATE
                        the template file
  -o OUTPUT, --output OUTPUT
                        the output file

Advanced options:
  For better control

  -td [TEMPLATE_DIR [TEMPLATE_DIR ...]], --template_dir [TEMPLATE_DIR [TEMPLATE_DIR ..
↪.]]
                        add more directories for template file lookup
  -cd CONFIGURATION_DIR, --configuration_dir CONFIGURATION_DIR
                        the directory for configuration file lookup
  -m MOBANFILE, --mobanfile MOBANFILE
                        custom moban file
  -g GROUP, --group GROUP
                        a subset of targets
  --template-type TEMPLATE_TYPE
```

(continues on next page)

(continued from previous page)

```
                        the template type, default is jinja2
  -d DEFINE [DEFINE ...], --define DEFINE [DEFINE ...]
                        to supply additional or override predefined variables,
                        format: VAR=VALUEs
  -e EXTENSION [EXTENSION ...], --extension EXTENSION [EXTENSION ...]
                        to to TEMPLATE_TYPE=EXTENSION_NAME
  -f                    force moban to template all files despite of
                        .moban.hashes

Developer options:
  For debugging and development

  --exit-code           tell moban to change exit code
  -V, --version         show program's version number and exit
  -v                    show verbose, try -v, -vv, -vvv
```

12.1 Exit codes

By default:

- 0 : no error
- 1 : error occured

With *–exit-code*:

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

CHAPTER 13

Tutorial

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

13.1 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

13.1.1 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

As well, you can define your own variable:

```
moban -D hello=maailman -t a.template
```

And when you check 'moban.output', you will find you have overwritten data.yaml.

13.2 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.

> **Warning:** *a.template* could be a symbolic link on Unix/Linux. It will not work if you template a symbolic link on Windows. Use symbolic link at your own calculated risk.

13.2.1 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*.

13.3 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.

13.3.1 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
```

(continues on next page)

(continued from previous page)

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

13.3.2 New development

Since verison 0.6.0, *overrides* syntax support two more use cases:

1 override more than one configuration file

For example:

```
overrides:
  - config-file-a.yaml
  - config-file-b.yaml
```

2 override more than one configuration file

For example:

```
overrides:
  - config-file-a.yaml: keya
  - config-file-b.yaml: keyb
```

13.4 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

13.4.1 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
```

(continues on next page)

(continued from previous page)

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

13.5 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

13.5.1 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

this demonstrations jinja2's include statement

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

13.6 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
```

(continues on next page)

(continued from previous page)

```
    - 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"

}

Warning: *a.template* could be a symbolic link on Unix/Linux. It will not work if you template a symbolic link on Windows. Use symbolic link at your own calculated risk.

13.6.1 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

shijie

this demonstrations jinja2's include statement

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

13.7 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.

13.7.1 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.

13.8 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.

13.9 level 9: moban dependency as pypi package

Note: You will need to install pypifs

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:

```
configuration:
  template_dir:
    - "pypi://pypi-mobans-pkg/resources/templates"
  configuration: config.yml
  configuration_dir: "pypi://pypi-mobans-pkg/config"
targets:
  - mytravis.yml: travis.yml.jj2
  - test.txt: demo.txt.jj2
```

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

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

Note: when you do not have relative directory:

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

13.10 level 10: moban dependency as git repo

Note: You will need to install gitfs2

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:

```
configuration:
  template_dir:
    - "git://github.com/moremoban/pypi-mobans.git!/templates"
    - local
  configuration: config.yml
  configuration_dir: "git://github.com/moremoban/pypi-mobans.git!/config"
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:"

13.10.1 The syntax when submodule exists

The sumodule syntax is:

```
configuration:
  template_dir:
    - "git://github.com/moremoban/pypi-mobans.git?submodule=true&branch=your_choice_
↪or_default_branch_if_not_specified!/templates"
    - local
```

If you have reference instead of branch:

```
configuration:
  template_dir:
    - "git://github.com/moremoban/pypi-mobans.git?submodule=true&reference=your_
↪alternative_reference_but_not_used_together_with_branch!/templates"
    - local
```

13.11 Level 11: use handlebars

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

13.11.1 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

13.12 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*

13.12.1 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
```

13.13 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.

13.13.1 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============
```

13.14 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.

13.14.1 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:
```

(continues on next page)

(continued from previous page)

```
  - 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

13.15 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
   - output_is_copied.same_file_extension: when_source_have.same_file_extension
```

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
```

13.15.1 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_is_copied.same_file_extension: when_source_have.same_file_extension
  - output: "misc-1-copying/can-create-folder/if-not-exists.txt"
    template: file-in-template-sources-folder.txt
    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.

13.16 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.

13.17 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
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.

13.18 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

13.18.1 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: )))'
```

13.19 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
```

13.20 Level 20: templates, files in a zip or tar

On top of level 6, you could have files in a zip or tar. In the following example:

```
configuration:
  configuration_dir: 'tar://custom-config.tar'
  template_dir:
    - zip://templates.zip
    - cool-templates
    - '.'
targets:
  - output: 'tar://a.tar/a.output'
    configuration: data.yml
    template: template.in.zip.jj2
  - output: 'zip://a.zip/a.output2'
    configuration: data2.yml
    template: subfolder/template.in.zip.jj2
```

where *template.in.zip.jj2* were loaded from a zip file

13.20.1 Evaluation

Please go to *docs/level-20-templates-configs-in-zip-or-tar* directory.

Here is the command to launch it:

```
moban
```

'a.output' is the generated file in a.tar:

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

world

shijie

this demonstrations jinja2's include statement

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

a.output2 is in a.zip:

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

world2
```

(continues on next page)

(continued from previous page)

```
shijie

this demonstrations jinja2's include statement

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

13.21 Level 21: template copying from a zip to a zip

In level 15, with *.moban.yml*, you can copy templates to your destination. Now with similiar moban syntax, let me show how to create a new zip file where all templates are copied to.

Explicit syntax:

```
targets:
  - output: "zip://your.zip/explicit"
    template: template_file
    template_type: copy
```

Implicit syntax:

```
targets:
  - output: "zip://your.zip/implicit"
    template: template_file.copy
```

Shorthand syntax:

```
targets:
  - "zip://your.zip/shorthand": 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: "zip://your.zip/dest-dir"
    template: source-dir
    template_type: copy
```

Recursive directory copying syntax:

```
targets:
  - output: "zip://your.zip/dest-dir"
    template: source-dir/**
    template_type: copy
```

13.21.1 Evaluation

Here is example moban file for copying:

```
configuration:
  template_dir:
    - "zip://template-sources.zip"
```

(continues on next page)

(continued from previous page)

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

13.22 Level 22: intermediate targets

It is natural to allow intermediate target to be source so that different moban plugins can interact with each other. The good news is since moban verion 0.6.5, it is support.

Note: The bad news is, folder as imtermediate target is not supported yet and will be considered in next incremental build. For now, the date cannot be confirmed.

Here are the syntax:

```
targets:
  - intermediate.jj2: original.jj2
  - final: intermediate.jj2
```

With moban 0.6.4-, above syntax cannot result in *final* file to be generated because *intermediate.jj2* does not exist until moban is run.

13.23 Level 23: moban file inheritance

It is a bit tedious to repeat a few common configuration in moban file. Why not create a parent moban file? Then allow child project to deviate from.

The answer is to use 'overrides' in *.moban.yaml*, so called moban file.

overrides could over ride any data file format in any location in theory. And it support override a specific key set.

13.24 level 24: templates and configuration files over http(s)

Note: You will need to install httpfs

Why not to take a template off the web? 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 mobanfile, it is possible to package up your mobans/templates and configuration files from a HTTP(S) protocol.

Here are the sample file:

```
configuration:
  template_dir:
    - "https://raw.githubusercontent.com/moremoban/pypi-mobans/dev/templates/"
    - local
  configuration: config.yml
  configuration_dir: "https://raw.githubusercontent.com/moremoban/pypi-mobans/dev/
↪config/"
targets:
  - mytravis.yml: travis.yml.jj2
  - test.txt: demo.txt.jj2
```

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

```
configuration:
  template_dir:
    - "https://raw.githubusercontent.com/moremoban/pypi-mobans/dev/templates/"
```

13.24.1 Maintenance note

To the maintainer, in order to eat the dog food. Please checkout pypi-mobans and run a http server inside local pypi-mobans folder.

Then update moban's mobanfile to:

```
configuration:
  template_dir:
    - "http://localhost:8000/templates/"
    - "http://localhost:8000/statics/"
    - ".moban.d"
```

Then run *make update*

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

CHAPTER 14

Developer Guide

14.1 Development guide

14.1.1 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

14.1.2 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.jj2 in pypi-mobans project

14.1.3 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.jj2 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

14.1.4 *repr*

Returns a single quoted string in the templated file

14.2 Built-in Tests

14.2.1 *exists*

Test if a file exists or not

Jinja2 Globals

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

14.2.2 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_fs, options=None):
        """
        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
```

(continues on next page)

(continued from previous page)

```
        """

    def apply_template(self, template, data, output):
        """
        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 15

Change log

15.1 0.7.3 - 2.5.2020

Added

1. Added continuous check in travis for setup.py descriptions. No impact to moban user.

15.2 0.7.2 - 1.5.2020

Added

1. Support for templates and configuration files over HTTP(S) protocol with httpfs! Yepee!

15.3 0.7.1 - 25.04.2020

Fixed

1. #365: regression was introduced by v0.6.5. If you uses mobanfile as data configuration file, you are very likely to have this show stopper. Please upgrade to this version.

15.4 0.7.0 - 18.01.2020

Removed

1. #360: make gitfs2 and pypifs optional.
2. #303: python 2.7 support is dropped.

Updated

1. #360: show friendlier error when unknown protocol exception was raised.

15.5 0.6.8 - 7.12.2019

Updated

1. since version 0.5.0, when rendering a single file or string, moban would report 'Templated 1 of 0 files', which should have been 'Templated 1 file.'

Removed

1. python 3.4 support is gone because colorama requires Python '>=2.7, !=3.0.*, !=3.1.*, !=3.2.*, !=3.3.*, !=3.4.*'

15.6 0.6.7 - 1.12.2019

Updated

1. no verbose for error, -v for warning, -vv for warning+info, -vvv for warning+info+debug
2. #351, show template plugin name, i.e. 'copying' for copy instead of 'templating'

Removed

1. Message: 'Warning: Attempting to use environment vars as data...' became warning log
2. Message: 'Warning: Both data.yml and /.../.moban.cd/data.yml does not exist' became warning log
3. with -v, you would see them in such a situation

15.7 0.6.6 - 10.11.2019

Added

1. support moban file inheritance. one base moban file and child repos can inherit and override

15.8 0.6.5 - 13.10.2019

Added

1. #335: support intermediate targets in moban file

15.9 0.6.4 - 4.10.2019

Updated

1. Command options have been grouped. –template_type became –template-type
2. Increment gitfs2 to version 0.0.2. gitfs#4

15.10 0.6.3 - 25.09.2019

Added

1. #260: jinja-cli parity: support command line pipe stream.

15.11 0.6.2 - 15.09.2019

Added

1. #322: Implicit targets with template extensions default to copy
2. #257: '-e' to load extensions for template engines, i.e. jinja2
3. #333: command line template fails with version 0.6.1

15.12 0.6.1 - 10.09.2019

Fixed

1. #328: update backward compatibility

15.13 0.6.0 - 10.09.2019

Added

1. #205: support pyFilesystem2
2. #185: -v will enable moban application logging for development. And -V is for version.
3. #325: -vv show debug trace
4. #126: Allow mobanfile to include data from arbitrary config files
5. #256: jinja2-cli parity: '-d hello=world' to define custom variable on cli

Updated

1. #275: fix moban 0.4.5 test failures on openSUSE Tumbleweed

15.14 0.5.0 - 14.07.2019

Updated

1. #277: Restored dependency git-url-parse, replacing incompatible giturlparse which was used during moban 0.4.x
2. #281: Fixed unicode support on Python 2.7
3. #274: Updated ruamel.yaml dependency pins to restore support for Python 3.4, and prevent installation of versions that can not be installed on Python 3.7
4. #285: Fixed CI testing of minimum requirements
5. #271: Fixed repository caching bug preventing branch switching
6. #292: Reformatted YAML files according to yamllint rules
7. #291: Fixed filename typos in README
8. #280: Added CI to ensure repository is in sync with upstream
9. #280: sync setup.py from pypi-mobans

15.15 0.4.5 - 07.07.2019

Updated

1. #271: support git branch change in later run.

15.16 0.4.4 - 26.05.2019

Updated

1. #265: Use simple *read binary* to read instead of encoding

15.17 0.4.3 - 16.03.2019

Removed

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

Added

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

Updated

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

15.18 0.4.2 - 08.03.2019

Added

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

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.

15.19 0.4.1 - 28.02.2019

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.

15.20 0.4.0 - 20.02.2019

Added

1. #165: Copy as plugins

Updated

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

15.21 0.3.10 - 03.02.2019

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

Updated

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

15.22 0.3.9 - 18-1-2019

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

15.23 0.3.8 - 12-1-2019

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.

15.24 0.3.7 - 6-1-2019

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

15.25 0.3.6 - 30-12-2018

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.

15.26 0.3.5 - 10-12-2018

Updated

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

15.27 0.3.4.1 - 28-11-2018

Updated

1. #137: missing contributors.rst file

15.28 0.3.4 - 18-11-2018

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

Updated

1. Template engine interface has been clarified and documented

15.29 0.3.3 - 05-11-2018

Added

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

15.30 0.3.2 - 04-11-2018

Added

1. configuration dirs may be located by *requires*, i.e. configuration files may be in a python package or git repository.

15.31 0.3.1 - 02-11-2018

Added

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

15.32 0.3.0 - 27-18-2018

Added

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

Updated

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

Removed

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

15.33 0.2.4 - 14-07-2018

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

15.34 0.2.3 - 10-07-2018

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

15.35 0.2.2 - 16-06-2018

Added

1. #31: create directory if missing during copying

Updated

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

15.36 0.2.1 - 13-06-2018

Updated

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

15.37 0.2.0 - 11-06-2018

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

Updated

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

15.38 0.1.4 - 29-May-2018

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

15.39 0.1.3 - 12-Mar-2018

Updated

1. handle unicode on python 2

15.40 0.1.2 - 10-Jan-2018

Added

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

15.41 0.1.1 - 08-Jan-2018

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

15.42 0.1.0 - 19-Dec-2017

Added

1. #14, provide shell exit code

15.43 0.0.9 - 24-Nov-2017

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.

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'

15.44 0.0.8 - 18-Nov-2017

Added

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

Updated

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

15.45 0.0.7 - 19-Jul-2017

Added

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

15.46 0.0.6 - 16-Jun-2017

Added

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

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.

15.47 0.0.5 - 17-Mar-2017

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

15.48 0.0.4 - 11-May-2016

Updated

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

15.49 0.0.3 - 09-May-2016

Updated

1. Bug fix #4, keep trailing new lines

15.50 0.0.2 - 27-Apr-2016

Updated

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

15.51 0.0.1 - 23-Mar-2016

Added

1. Initial release

CHAPTER 16

Migration Notes

CHAPTER 17

Indices and tables

- genindex
- modindex
- search