moban

Release 0.8.2

Sep 10, 2020

Contents

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

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

CHAPTER 1

Announcement

In version 0.8.0, *moban.plugins.jinja2.tests.files* is moved to moban-ansible package. *moban.plugins.jinja2.filters.github* is moved to moban-jinja2-github package Please install them for backward compatibility.

CHAPTER 2

Quick start

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

Given data.yml as:

```
hello: world
```

and my.template as:

```
{{hello}}
```

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

2.1 Template inheritance and custom template directories

Suppose there exists *shared/base.jj2*, and two templates *child1.jj2* and *child2.jj2* derives from it. You can do:

```
$ moban -t child1.jj2 -td shared -o child1
$ moban -t child2.jj2 -td shared -o child2
```

2.2 Data overload and custom data directories

Effectively each data file you give to moban, it overrides environment variables. Still you can have different layers of data. For example, you can have *shared/company_info.yml*, use *project1.yml* for project 1 and *project2.yml* for project 2. In each of the derived data file, simply mention:

```
overrides: company_info.yml
...
```

Here is the command line to use your data:

```
$ moban -cd shared -c project1.yaml -t README.jj2
```

2.3 Custom jinja2 extension

moban allows the injection of user preferred jinja2 extensions:

```
$ moban -e jj2=jinja2_time.TimeExtension ...
```

2.4 Well, can I nick some existing functions as filters, tests? Or create a global from another library?

Sure, you can use the same '-e' syntax:

```
$ moban -e jinja2=filter:module.path.filter_function \
           jinja2=test:module.path.test_function \
           jinja2=global:identifier=module.path.variable
```

In this case, you would have to include the external library in your own requirements.txt

Here is an example:

```
$ moban -e jinja2=filter:moban.externals.file_system.url_join \
  jinja2=test:moban.externals.file_system.exists \
  jinja2=global:description=moban.constants.PROGRAM_DESCRIPTION \
  -t "{{ 'a'|url_join('b')}} {{'b' is exists}}"
```

2.5 Can I write my own jinja2 test, filter and/or globals?

moban allows the freedom of craftsmanship. Please refer to the docs for more details. Here is an example:

```
import sys
import base64

from moban.plugins.jinja2.extensions import JinjaFilter


@JinjaFilter()
def base64encode(string):
    if sys.version_info[0] > 2:
        content = base64.b64encode(string.encode("utf-8"))
        content = content.decode("utf-8")
    else:
        content = base64.b64encode(string)
    return content
```

And you can use it within your jinja2 template, *mytest.jj2*:

```
{{ 'abc' | base64encode }}
```

Assume that the custom example was saved in *custom-jj2-plugin*

```
$ moban -pd custom-jj2-plugin -t mytest.jj2 ...
```

Moban will then load your custom jinja2 functions

2.6 Slim template syntax for jinja2

with moban-slim installed,

Given a data.json file with the following content

```
$ moban --template-type slim -c data.json  "{{person.firstname}} {{person.lastname}}"
Smith Jones
```

2.7 Handlebars.js template

With moban-handlebars installed,

Given a data.json file with the following content

```
$ moban --template-type handlebars -c data.json  "{{person.firstname}} {{person.
↪lastname}}"
Yehuda Katz
```

For *handlebars.js* users, yes, the example was copied from handlebarjs.com. The aim is to show off what we can do.

Let's continue with a bit more fancy feature:

```
$ moban --template-type handlebars -c data.json "{{#with person}}{{firstname}} {
↪{lastname}} {{/with}}"
```

Moban's way of pybar3 usage:

Let's save the following file a *script.py* under *helper_and_partial* folder:

```
from moban_handlebars.api import Helper, register_partial

register_partial('header', '<h1>People</h1>')


@Helper('list')
def _list(this, options, items):
    result = [u'<ul>']
    for thing in items:
        result.append(u'<li>')
        result.extend(options['fn'](thing))
        result.append(u'</li>')
    result.append(u'</ul>')
    return result
```

And given *data.json* reads as the following:

Let's invoke handlebar template:

```
$ moban --template-type hbs -pd helper_and_partial -c data.json "{{>header}}{{#list␣
↪people}}{{name}} {{age}}{{/list}}"
Handlebars-ing {{>header}... to moban.output
Handlebarsed 1 file.
$ cat moban.output
<h1>People</h1><ul><li>Bill 100</li><li>Bob 90</li><li>Mark 25</li></ul>
```

2.8 Velocity template

With moban-velocity installed,

Given the following data.json:

And given the following velocity.template:

moban can do the template:

```
$ moban --template-type velocity -c data.json -t velocity.template
Old people:

Bill

Bob


Third person is Mark
```

2.9 Can I write my own template engine?

Yes and please check for more details.

Given the following template type function, and saved in custom-plugin dir:

```
from moban.core.content_processor import ContentProcessor


@ContentProcessor("de-duplicate", "De-duplicating", "De-duplicated")
def de_duplicate(content: str, options: dict) -> str:
    lines = content.split(b'\n')
    new_lines = []
    for line in lines:
        if line not in new_lines:
            new_lines.append(line)
    return b'\n'.join(new_lines)
```

You can start using it like this:

```
$ moban --template-type de-duplicate -pd custom-plugin -t duplicated_content.txt
```

2.10 TOML data format

moban-anyconfig should be installed first.

Given the following toml file, sample.toml:

You can do:

```
$ moban -c sample.toml "{{owner.name}} made {{title}}"
Tom Preston-Werner made TOML Example
```

Not limited to toml, you can supply moban with the following data formats:

Table 1: Always supported formats, quoting from python-anyconfig

Format	Type	Requirement
JSON	json	`json` (standard lib) or `simplejson`
Ini-like	ini	`configparser` (standard lib)
Pickle	pickle	`pickle` (standard lib)
XML	xml	`ElementTree` (standard lib)
Java properties	properties	None (native implementation with standard lib)
B-sh	shellvars	None (native implementation with standard lib)

For any of the following data formats, you elect to install by yourself.

Table 2: Supported formats by pluggable backend modules

Format	Type	Required backend
Amazon Ion	ion	`anyconfig-ion-backend`
BSON	bson	`anyconfig-bson-backend`
CBOR	cbor	`anyconfig-cbor-backend` or `anyconfig-cbor2-backend`
ConifgObj	configobj	`anyconfig-configobj-backend`
MessagePack	msgpack	`anyconfig-msgpack-backend`

Or you could choose to install all:

```
$ pip install moban-anyconfig[all-backends]
```

Why not to use python-anyconfig itself, but yet another package?

moban gives you a promise of any location which *python-anyconfig* does not support.

Why do it mean 'any location'?

Thanks to pyfilesystem 2, moban is able to read data back from git repo, pypi package, http(s), zip, tar, ftp, s3 or you name it.

CHAPTER 3

Templates and configuration files over HTTP(S)

httpfs should be installed first.

With httpfs, moban can access any files over http(s) as its template or data file:

```
$ 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 4

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 5

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 6

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:

CHAPTER 7

So what can I do with it

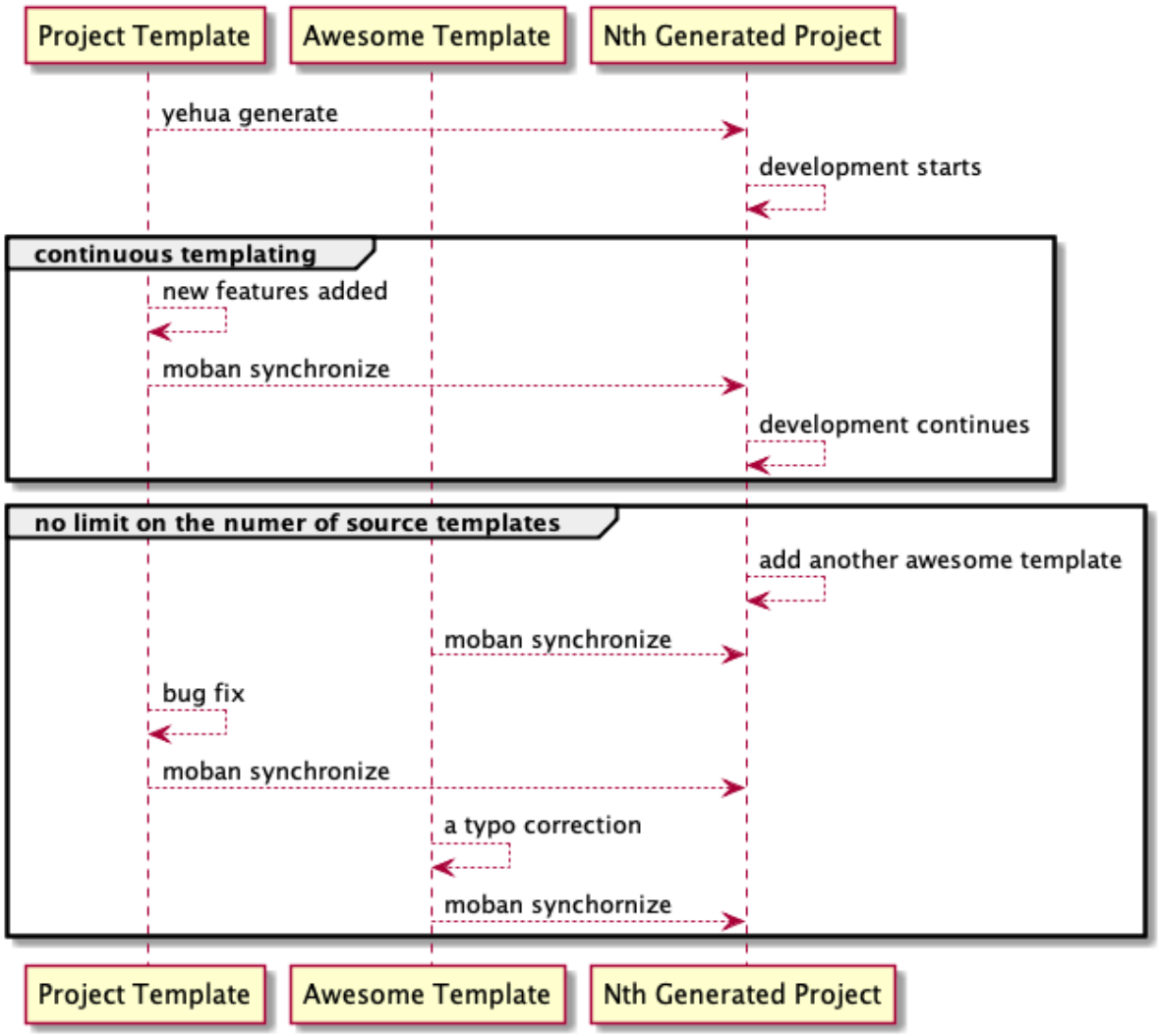
Here is a list of other usages:

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

CHAPTER 8

At scale, continous templating for open source projects

moban enabled **continuous templating** in pyexcel and coala project to keep documentation consistent across the documentations of individual libraries in the same organisation. Here is the primary use case of moban, as of now:


Project Template
Awesome Template
Nth Generated Project
yehua generate
development starts
continuous templating
new features added
moban synchronize
development continues
no limit on the numer of source templates
add another awesome template
moban synchronize
bug fix
moban synchronize
a typo correction
moban synchornize
Project Template
Awesome Template
Nth Generated Project

CHAPTER 9

Usage beyond command line

All use cases are documented

CHAPTER 10

Support

If you like moban, please support me on github, 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 11

Vision

Any template, any data in any location

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.

CHAPTER 12

Credit

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

CHAPTER 13

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 14

CLI documentation

```
usage: moban [-h] [-c CONFIGURATION] [-t TEMPLATE] [-o OUTPUT]
             [-td [TEMPLATE_DIR [TEMPLATE_DIR ...]]]
             [-pd [PLUGIN_DIR [PLUGIN_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
  -pd [PLUGIN_DIR [PLUGIN_DIR ...]], --plugin_dir [PLUGIN_DIR [PLUGIN_DIR ...]]
                        add more directories for plugin lookup
  -m MOBANFILE, --mobanfile MOBANFILE
                        custom moban file
```

(continues on next page)

(continued from previous page)

```
  -g GROUP, --group GROUP
                        a subset of targets
  --template-type TEMPLATE_TYPE
                        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           by default, exist code 0 means no error, 1 means error
                        occured. It tells moban to change 1 for changes, 2 for
                        error occured
  -V, --version         show program's version number and exit
  -v                    show verbose, try -v, -vv, -vvv
```

CHAPTER 15

Tutorial

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

15.1 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

15.1.1 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.

15.2 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.

15.2.1 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*.

15.3 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.

15.3.1 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============
```

15.3.2 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
```

15.4 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

15.4.1 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============
```

15.5 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

15.5.1 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============
```

15.6 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.

15.6.1 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============
```

15.7 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.

Note: if you intend to use extensions for one off usage, please use '-pd' cli option. i.e. *moban -td my-templates/ -t filter.jj2 -pd custom-jj2-plugin*

15.7.1 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.

15.8 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.

15.9 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"
```

15.10 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:"

15.10.1 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
```

15.11 Level 11: use handlebars

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

15.11.1 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

15.12 Level 12: use template engine extensions

There are three possible ways to provide extensions for your template engine. Let's take jinja2 as an example.

15.12.1 1. Ready made 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*

Command line

if you intend to use extensions for one off usage, please use '-e' cli option. for example: moban -e jinja2=your_custom_jinja2_extension

15.12.2 2. Ad-hoc declaration

Let's say you are fond of some existing functions, for example, ansible's combine filter. With moban, you can immediately include it for your template via the following syntax:

For example:

```
extensions:
  jinja2:
    - filter:ansible.plugins.filter.core.combine
    - test:moban.externals.file_system.exists
```

Command line

```
$ moban -e jinja2=filter:module.path.filter_function jinja2=test:module.path.test_
↪function jinja2=global:identifier=module.path.variable
```

you can do this:

```
$ moban -e jinja2=filter:module.path.filter_function \
           jinja2=test:module.path.test_function \
           jinja2=global:identifier=module.path.variable
```

15.12.3 3. Make your own extensions

You can choose to write an extension for the template type of your choice. For example, you can write a reusable extension for jinja2. moban will be able to load it as it is.

If you decide that you only want to write them for moban but for your own use, you can follow *Level 7: Custom jinja filters, tests and globals* and write your own. When you would like to make yours avaiable for all moban users, you can follow moban-jinja2-github and moban-ansible

15.12.4 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
```

15.13 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.

15.13.1 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============
```

15.14 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.

15.14.1 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

15.15 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
```

15.15.1 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.

15.16 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:
```

(continues on next page)

(continued from previous page)

```
  - 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.

15.17 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.

15.18 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

15.18.1 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: ')))'
```

15.19 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
```

15.20 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

15.20.1 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

shijie

this demonstrations jinja2's include statement

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

15.21 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
```

15.21.1 Evaluation

Here is example moban file for copying:

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

(continues on next page)

(continued from previous page)

```
    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.

15.22 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.

15.23 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.

15.24 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/"
```

15.24.1 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*

15.25 Level 25: delete intermediate targets

Continue with level 22, we would like to delete intermediate files.

Note: What is intermediate targets? Simply they are the files moban generates but in the end those files are not really used.

For safety reasons, we only delete intermediate targets. We are not allowing moban to delete any files in template folders and staic folder.

Here is the short syntax:

```
targets:
  - delete!: intermediate_file.jj2
```

Here are the full syntax:

```
targets:
  - output: what_ever_here_will_be_ignored
    template: intermediate.jj2
    template_type: delete
  - output: ''
    template: intermediate2.jj2
```

Example mobanfile:

```
targets:
  - intermediate.jj2: original.jj2
  - intermediate2.jj2: original.jj2
  - intermediate3.jj2: original.jj2
  - output: x
    template: intermediate.jj2
    template_type: delete
  - output: ''
    template: intermediate2.jj2
  - delete!: intermediate3.jj2
```

15.26 Level 26: Strip the white spaces

It was requested, a long time ago, to be able to strip the white spaces before and after the rendered content. Due to these factors:

1. templating order needs to be respected first
2. intermediate targets(moban generated files) can be allowed as template
3. and delete the intermediate file

Now, all three factors are now supported. Hence, 'strip' feature can be rolled out.

Here is the short syntax:

```
targets:
  - final: intermediate_file.strip
```

Here are the full syntax:

```
targets:
  - output: final
    template: intermediate_file.what_ever
    template_type: strip
```

Example mobanfile:

```
targets:
  - intermediate.strip: content_with_lots_of_white_spaces.jj2
  - final: intermediate.strip
  - delete!: intermediate.strip
```

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

CHAPTER 16

Developer Guide

16.1 Development guide

16.1.1 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.plugins.jinja2.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))
```

16.1.2 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

16.1.3 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

16.1.4 *repr*

Returns a single quoted string in the templated file

16.2 Built-in Tests

16.2.1 *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.

16.2.2 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):
```

(continues on next page)

(continued from previous page)

```
    def __init__(self, template_fs, options=None):
        """
        an instance of fs.multifs.MultiFS will be given.

        :param fs.multifs.MultiFS template_fs: a MultiFS instance or a FS instance
        :param dict options: a dictionary containing environmental parameters
        """

    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 get_template_from_string(self, string):
        """
        Sometimes, user would pass on command line string as template
        """

    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.

When the template engine failed to obtain the template, i.e. UnicodeEncodingError, TemplateSyntaxError, your engine extension shall raise *moban.exceptions.PassOn* exception, and *moban* would replace your template engine with default engine.

16.2.3 Custom content processors for Moban

Since version 0.7.7, it became easy to write a content processor for moban. What you need is a content processing function, which will be fed the content of *template_file* and which is expected to return a string. And decorate your function with *ContentProcessor*:

```
@ContentProcessor('strip', 'Stripping', 'Stripped'):
def strip(template_file: str) -> str:
    ret = template_file.strip()
    return ret
```

Here is how *copy* template type is coded:

```
from moban.core.content_processor import ContentProcessor


@ContentProcessor("copy", "Copying", "Copied")
def copy(content: str, _: dict) -> str:
    """
    Does no templating, works like 'copy'.
```

(continues on next page)

(continued from previous page)

```
    Respects templating directories, for example: naughty.template
    could exist in any of template directires: dir1,
    dir2, dir3, and this engine will find it for you.  With conventional
    copy command, the source file path must be known.

    And this engine does not really touch the dest file but only read
    the source file. Everything else is taken care of by moban
    templating mechanism.
    """
    return content
```

CHAPTER 17

Change log

17.1 0.8.2 - 04.09.2020

Fixed

1. Use any functions, any data structure of any python packages as jinja2 filters, tests, globals

17.2 0.8.1 - 04.09.2020

Fixed

1. #399: content processor should be called only once
2. content processor shall pass on options to content processors

17.3 0.8.0 - 02.09.2020

Removed

1. moban.plugins.jinja2.tests.files is moved to moban-ansible package
2. moban.plugins.jinja2.filters.github is moved to moban-jinja2-github package

Fixed

1. #396: custom jinja2 plugins(filters, tests and globals) are not visible if a template is passed as a string.

17.4 0.7.10 - 16.08.2020

Updated

1. #393: Rendered content output to stdout once

17.5 0.7.9 - 06.08.2020

Updated

1. #390: single render action will print to stdout by default

17.6 0.7.8 - 09.06.2020

Added

1. #313: Non-textual source files should default to copy

17.7 0.7.7 - 24.5.2020

Added

1. *-pd* for command line to include custom plugin directories

Fixed

1. strip did not work in 0.7.6

17.8 0.7.6 - 22.5.2020

Added

1. #38: finally be able strip the rendered content

17.9 0.7.5 - 21.5.2020

Added

1. #167: reverse what moban have done: delete

17.10 0.7.4 - 13.5.2020

Fixed

1. #378: suppress stdout message from deprecated pip install. but please do not use and migrate deprecated‘requires‘ syntax.

17.11 0.7.3 - 2.5.2020

Added

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

17.12 0.7.2 - 1.5.2020

Added

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

17.13 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.

17.14 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.

17.15 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.*'

17.16 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

17.17 0.6.6 - 10.11.2019

Added

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

17.18 0.6.5 - 13.10.2019

Added

1. #335: support intermediate targets in moban file

17.19 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

17.20 0.6.3 - 25.09.2019

Added

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

17.21 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

17.22 0.6.1 - 10.09.2019

Fixed

1. #328: update backward compatibility

17.23 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

17.24 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

17.25 0.4.5 - 07.07.2019

Updated

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

17.26 0.4.4 - 26.05.2019

Updated

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

17.27 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

17.28 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.

17.29 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.

17.30 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

17.31 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

17.32 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

17.33 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.

17.34 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

17.35 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.

17.36 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.

17.37 0.3.4.1 - 28-11-2018

Updated

1. #137: missing contributors.rst file

17.38 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

17.39 0.3.3 - 05-11-2018

Added

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

17.40 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.

17.41 0.3.1 - 02-11-2018

Added

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

17.42 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

17.43 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

17.44 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

17.45 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

17.46 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

17.47 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

17.48 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

17.49 0.1.3 - 12-Mar-2018

Updated

1. handle unicode on python 2

17.50 0.1.2 - 10-Jan-2018

Added

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

17.51 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

17.52 0.1.0 - 19-Dec-2017

Added

1. #14, provide shell exit code

17.53 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'

17.54 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.

17.55 0.0.7 - 19-Jul-2017

Added

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

17.56 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.

17.57 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

17.58 0.0.4 - 11-May-2016

Updated

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

17.59 0.0.3 - 09-May-2016

Updated

1. Bug fix #4, keep trailing new lines

17.60 0.0.2 - 27-Apr-2016

Updated

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

17.61 0.0.1 - 23-Mar-2016

Added

1. Initial release

CHAPTER 18

Migration Notes

18.1 Trouble shooting guide

1. Why a file was not templated but copied instead?

   It has been coded so that template engine can choose to pass on the template if it failed to handle. Moban will take over and use default 'copy' action.

   In order to find out what went wrong, you can use '-vvv' to enable all logs to assist you.

CHAPTER 19

Indices and tables

- genindex
- modindex
- search