

---

# **moban**

***Release 0.6.0***

**Sep 10, 2019**



---

## Contents

---

<b>1</b>	<b>Documentation</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>Quick start</b>	<b>7</b>
<b>4</b>	<b>Work with files in a git repo</b>	<b>9</b>
<b>5</b>	<b>Work with files in a python package</b>	<b>11</b>
<b>6</b>	<b>Work with S3 and other cloud based file systems</b>	<b>13</b>
<b>7</b>	<b>Usage</b>	<b>15</b>
<b>8</b>	<b>Tutorial</b>	<b>17</b>
<b>9</b>	<b>Developer Guide</b>	<b>35</b>
<b>10</b>	<b>Change log</b>	<b>39</b>
<b>11</b>	<b>Indices and tables</b>	<b>49</b>



**Author** C.W.

**Issues** <http://github.com/moremoban/moban/issues>

**License** MIT

**Version** 0.6.0

**Generated** Sep 10, 2019

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

Our vision is: any template, any data in any location. Our current architecture enables moban to plugin any python template engine: mako, handlebars, velocity, haml, slim and tornado, to plugin any data format: json and yaml, and in any location: zip, git, pypi package, s3, etc. Please look at our issues. We have many more template engines and data format on the road map.



# CHAPTER 1

---

## Documentation

---

All use cases are documented [here](#)





## CHAPTER 2

---

### 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 3

---

### Quick start

---

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

Or simply

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

A bit formal example:

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

Given data.yml as:

```
hello: world
```

and my.template as:

```
{{hello}}
```

moban.output will contain:

```
world
```

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



## CHAPTER 4

---

### Work with files in a git repo

---

Please install `gitfs2`:

```
$ pip install gitfs2
```

And then you can do the following:

```
$ 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."
```



---

### Work with files in a python package

---

Please install `pypifs`:

```
$ pip install pypifs
```

And then you can do the following:

```
$ 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."
```





---

### Work with S3 and other cloud based file systems

---

Please install `fs-s3fs`:

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

```
$ moban -c s3://${client_id}:${client_secret}@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 7

---

### Usage

---

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

Yet another jinja2 cli command for static text generation

positional arguments:

template                    string templates

optional arguments:

-h, --help                    show this help message and exit

-cd CONFIGURATION\_DIR, --configuration\_dir CONFIGURATION\_DIR  
                              the directory for configuration file lookup

-c CONFIGURATION, --configuration CONFIGURATION  
                              the dictionary file

-td [TEMPLATE\_DIR [TEMPLATE\_DIR ...]], --template\_dir [TEMPLATE\_DIR [TEMPLATE\_DIR ..  
→.]]  
                              the directories for template file lookup

-t TEMPLATE, --template TEMPLATE  
                              the template file

-o OUTPUT, --output OUTPUT  
                              the output file

--template\_type TEMPLATE\_TYPE  
                              the template type, default is jinja2

-f                             force moban to template all files despite of  
                              .moban.hashes

--exit-code                  tell moban to change exit code

-m MOBANFILE, --mobanfile MOBANFILE  
                              custom moban file

-g GROUP, --group GROUP  
                              a subset of targets

-V, --version                 show program's version number and exit

(continues on next page)

(continued from previous page)

```
-v                show verbose
-d DEFINE [DEFINE ...], --define DEFINE [DEFINE ...]
                  to take a list of VAR=VALUES
```

## 7.1 Exit codes

By default:

- 0 : no error
- 1 : error occurred

With *-exit-code*:

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

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

## 8.1 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

### 8.1.1 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.yml.

## 8.2 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.

### 8.2.1 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*.

## 8.3 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.

### 8.3.1 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=====
```

## 8.4 New development

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

### 8.4.1 1 override more than one configuration file

For example:

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

### 8.4.2 2 override more than one configuration file

For example:

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

## 8.5 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

### 8.5.1 Evaluation

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

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

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

where *targets* should lead an array of dictionaries.

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

```
moban
```

‘a.output’ is the generated file:

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

world

shijie

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

## 8.6 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

### 8.6.1 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=====
```

## 8.7 Level 6: Complex Configuration

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



```

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

```

where *template* under *configuration* 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.

## 8.7.1 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=====

```

## 8.8 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.

### 8.8.1 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.

## 8.9 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.

## 8.10 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"
```

## 8.11 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:”

### 8.11.1 The syntax when submodule exists

The submodule 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
```

## 8.12 Level 11: use handlebars

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

### 8.12.1 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
```

## 8.13 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*

### 8.13.1 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
```

## 8.14 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.

### 8.14.1 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=====
```

## 8.15 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.

### 8.15.1 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. Otherwise your plugin won't load

## 8.16 Level 15: template copying becomes an action plugin in targets

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

Explicit syntax:

```

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

```

Implicit syntax:

```

targets:
- output: explicit
  template: template_file.copy

```

Shorthand syntax:

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

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

Directory copying syntax:

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

Recursive directory copying syntax:

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

## 8.16.1 Evaluation

Here is example moban file for copying:

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

## 8.17 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-recursive-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.

---

## 8.18 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-recursive-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.

## 8.19 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

### 8.19.1 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: ')))'
```

## 8.20 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-recursive-copying/\*\*

you can do the following things:

```
$ moban -g copy
```

## 8.21 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

### 8.21.1 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=====
```

## 8.22 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
```

### 8.22.1 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-recursive-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.

For more complex use case, please look at [its usage in pyexcel project](#)



## 9.1 Development guide

### 9.1.1 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

### 9.1.2 `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

### 9.1.3 `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 pull request 1
=====	=====

More details can be found in [moban's changelog.yml](#)

### 9.1.4 repr

Returns a single quoted string in the templated file

## 9.2 Built-in Tests

### 9.2.1 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.

### 9.2.2 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.



### 10.1 0.6.0 - 10.09.2019

#### 10.1.1 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

#### 10.1.2 Updated

1. [#275](#): fix moban 0.4.5 test failures on openSUSE Tumbleweed

### 10.2 0.5.0 - 14.07.2019

#### 10.2.1 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

## 10.3 0.4.5 - 07.07.2019

### 10.3.1 Updated

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

## 10.4 0.4.4 - 26.05.2019

### 10.4.1 Updated

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

## 10.5 0.4.3 - 16.03.2019

### 10.5.1 Removed

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

### 10.5.2 Added

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

### 10.5.3 Updated

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

## 10.6 0.4.2 - 08.03.2019

### 10.6.1 Added

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

## 10.6.2 Updated

1. [#180](#): No longer two statistics will be shown in v0.4.x. legacy copy targets are injected into a normal targets. cli target is made a clear priority.
2. [#244](#): version 0.4.2 is first version which would work perfectly on windows since 17 Nov 2018. Note that: file permissions are not used on windows. Why the date? because samefile is not available 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.

## 10.7 0.4.1 - 28.02.2019

### 10.7.1 Added

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

## 10.8 0.4.0 - 20.02.2019

### 10.8.1 Added

1. [#165](#): Copy as plugins

### 10.8.2 Updated

1. [#219](#): git clone depth set to 2
2. [#186](#): lowest dependency on ruamel.yaml is 0.15.5, Jun 2017

## 10.9 0.3.10 - 03.02.2019

### 10.9.1 Added

1. [#174](#): Store git cache in XDG\_CACHE\_DIR
2. [#107](#): Add -v to show current moban version
3. [#164](#): support additional data formats

### 10.9.2 Updated

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

## 10.10 0.3.9 - 18-1-2019

### 10.10.1 Updated

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

## 10.11 0.3.8 - 12-1-2019

### 10.11.1 Updated

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

## 10.12 0.3.7 - 6-1-2019

### 10.12.1 Updated

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

## 10.13 0.3.6 - 30-12-2018

### 10.13.1 Updated

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

## 10.14 0.3.5 - 10-12-2018

### 10.14.1 Updated

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

## 10.15 0.3.4.1 - 28-11-2018

### 10.15.1 Updated

1. [#137](#): missing contributors.rst file

## 10.16 0.3.4 - 18-11-2018

### 10.16.1 Added

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

### 10.16.2 Updated

1. Template engine interface has been clarified and documented

## 10.17 0.3.3 - 05-11-2018

### 10.17.1 Added

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

## 10.18 0.3.2 - 04-11-2018

### 10.18.1 Added

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

## 10.19 0.3.1 - 02-11-2018

### 10.19.1 Added

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

## 10.20 0.3.0 - 27-11-2018

### 10.20.1 Added

1. [#89](#): Install pypi-hosted mobans through *requires* syntax

## 10.20.2 Updated

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

## 10.20.3 Removed

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

## 10.21 0.2.4 - 14-07-2018

### 10.21.1 Added

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

## 10.22 0.2.3 - 10-07-2018

### 10.22.1 Added

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

## 10.23 0.2.2 - 16-06-2018

### 10.23.1 Added

1. #31: create directory if missing during copying

### 10.23.2 Updated

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

## 10.24 0.2.1 - 13-06-2018

### 10.24.1 Updated

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



## 10.25 0.2.0 - 11-06-2018

### 10.25.1 Added

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

### 10.25.2 Updated

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

## 10.26 0.1.4 - 29-May-2018

### 10.26.1 Updated

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

## 10.27 0.1.3 - 12-Mar-2018

### 10.27.1 Updated

1. handle unicode on python 2

## 10.28 0.1.2 - 10-Jan-2018

### 10.28.1 Added

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

## 10.29 0.1.1 - 08-Jan-2018

### 10.29.1 Added

1. the ability to present a long text as multi-line paragraph with a custom upper limit
2. speical filter expand github references: pull request and issues

3. #15: fix templating syntax to enable python 2.6

## 10.30 0.1.0 - 19-Dec-2017

### 10.30.1 Added

1. #14, provide shell exit code

## 10.31 0.0.9 - 24-Nov-2017

### 10.31.1 Added

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

### 10.31.2 Updated

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

## 10.32 0.0.8 - 18-Nov-2017

### 10.32.1 Added

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

### 10.32.2 Updated

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

## 10.33 0.0.7 - 19-Jul-2017

### 10.33.1 Added

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

## 10.34 0.0.6 - 16-Jun-2017

### 10.34.1 Added

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

### 10.34.2 Updated

1. moban will not template target file in the situation where the changes occurred 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.

## 10.35 0.0.5 - 17-Mar-2017

### 10.35.1 Added

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

## 10.36 0.0.4 - 11-May-2016

### 10.36.1 Updated

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

## 10.37 0.0.3 - 09-May-2016

### 10.37.1 Updated

1. Bug fix #4, keep trailing new lines

## 10.38 0.0.2 - 27-Apr-2016

### 10.38.1 Updated

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

## 10.39 0.0.1 - 23-Mar-2016

### 10.39.1 Added

1. Initial release

# CHAPTER 11

---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`