

---

# **mrbob Documentation**

*Release 0.2.dev0*

**Domen Kožar, Tom Lazar**

June 09, 2014



|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>User guide</b>  | <b>3</b>  |
| 1.1      | Installation . . . . .   | 3         |
| 1.2      | Usage . . . . .  | 3         |
| 1.3      | Sample template to try out . . . . .   | 4         |
| 1.4      | Listing all questions needed to have corresponding variable for a template . . . . . | 4         |
| 1.5      | Remember answers to a config file . . . . .  | 4         |
| 1.6      | Using non-interactive mode . . . . .   | 5         |
| 1.7      | Configuration . . . . .  | 5         |
| 1.8      | Collection of community managed templates . . . . .                                  | 6         |
| <b>2</b> | <b>Writing your own template</b>   | <b>7</b>  |
| 2.1      | Starting . . . . .   | 7         |
| 2.2      | How it works . . . . .   | 7         |
| 2.3      | Writing Questions . . . . .  | 8         |
| 2.4      | Common needs for templating . . . . .  | 8         |
| 2.5      | Hooks . . . . .  | 9         |
| 2.6      | template section reference . . . . .   | 10        |
| <b>3</b> | <b>Design goals</b>  | <b>11</b> |
| <b>4</b> | <b>Why another tool</b>  | <b>13</b> |
| <b>5</b> | <b>Developer guide</b>   | <b>15</b> |
| 5.1      | Setup developer environment . . . . .  | 15        |
| 5.2      | Running tests . . . . .  | 15        |
| 5.3      | Making a Release . . . . .   | 15        |
| <b>6</b> | <b>Source documentation</b>  | <b>17</b> |
| 6.1      | mrbob – Main package . . . . .   | 17        |
| <b>7</b> | <b>Changelog</b>   | <b>23</b> |
| 7.1      | 0.2 (unreleased) . . . . .   | 23        |
| 7.2      | 0.1.1 (2014-03-03) . . . . .   | 23        |
| 7.3      | 0.1 (2014-03-02) . . . . .   | 23        |
| 7.4      | 0.1a9 (2013-04-26) . . . . .   | 24        |
| 7.5      | 0.1a8 (2013-03-11) . . . . .   | 24        |
| 7.6      | 0.1a7 (2013-01-23) . . . . .   | 24        |
| 7.7      | 0.1a6 (2013-01-02) . . . . .   | 24        |
| 7.8      | 0.1a5 (2012-12-12) . . . . .   | 25        |

|          |                            |           |
|----------|----------------------------|-----------|
| 7.9      | 0.1a4 (2012-12-11)         | 25        |
| 7.10     | 0.1a3 (2012-11-30)         | 25        |
| 7.11     | 0.1a2 (2012-11-29)         | 25        |
| 7.12     | 0.1a1 (2012-10-19)         | 25        |
| <b>8</b> | <b>Glossary</b>            | <b>27</b> |
| <b>9</b> | <b>Indices and tables</b>  | <b>29</b> |
|          | <b>Python Module Index</b> | <b>31</b> |

**Author** Tom Lazar <tom@tomster.org>, Domen Kožar <domen@dev.si>

**Source code** [github.com project](#)

**Bug tracker** [github.com issues](#)

**License** BSD

**Generated** June 09, 2014

**Version** 0.2.dev0

## Features

- asks questions which need to be answered to render structure
- questions can be grouped by using a namespace
- renders templates from a folder, Python egg, or zip file
- supports Python 2.6 - 3.3, pypy
- 100% test coverage
- uses Jinja as the default rendering engine (can be replaced)
- multiple ways to specify variables to render templates
- preserves permissions when rendering templates
- provides hooks before/after asking a question
- provides hooks before/after rendering structure
- can remember given answers for rendered structure

## Flow of mr.bob

## Introduction

**mr.bob** is a tool that takes a directory skeleton, copies over its directory structure to a target folder and can use the [Jinja2](#) (or some other) templating engine to dynamically generate the files. Additionally, it can ask you questions needed to render the structure, or provide a config file to answer them.

**mr.bob** is meant to deprecate previous tools such as [paster](#) ([PasteScript](#)) and [templer](#).



## 1.1 Installation

```
$ pip install mr.bob
```

## 1.2 Usage

Once you install `mr.bob`, the `mrbob` command is available:

```
$ mrbob --help
usage: mrbob [-h] [-O TARGET_DIRECTORY] [-v] [-c CONFIG] [-V] [-l] [-w] [-n]
            [-q]
            [template]
```

Filesystem template renderer

positional arguments:

|          |  |
|----------|--|
| template | Template name to use for rendering. See <a href="http://mrbob.readthedocs.org/en/latest/userguide.html#usage">http://mrbob.readthedocs.org/en/latest/userguide.html#usage</a> for a guide to template syntax |
|----------|--|

optional arguments:

|   |   |
|---|---|
| <code>-h, --help</code>   | show this help message and exit   |
| <code>-O TARGET_DIRECTORY, --target-directory TARGET_DIRECTORY</code> | Where to output rendered structure. Defaults to current directory                               |
| <code>-v, --verbose</code>  | Print more output for debugging   |
| <code>-c CONFIG, --config CONFIG</code>                               | Configuration file to specify either <code>[mr.bob]</code> or <code>[variables]</code> sections |
| <code>-V, --version</code>  | Display version number  |
| <code>-l, --list-questions</code>                                     | List all questions needed for the template  |
| <code>-w, --remember-answers</code>                                   | Remember answers to <code>.mrbob.ini</code> file inside output directory                        |
| <code>-n, --non-interactive</code>                                    | Don't prompt for input. Fail if questions are required but not answered                         |
| <code>-q, --quiet</code>  | Suppress all but necessary output   |

By default, the target directory is the current folder. The most basic use case is rendering a template from a relative folder:

```
$ mrbob ../template_folder/
```

Or from a package:

```
$ mrbob some.package:template_folder/
```

Or from a downloaded zip file:

```
$ mrbob https://example.com/templates/mytemplate.zip
```

Or from a relative path in a zip file:

```
$ mrbob https://github.com/iElectric/mr.bob/archive/master.zip#mr.bob-master/mrbob/template_sample
```

## 1.3 Sample template to try out

```
$ mrbob mrbob:template_sample/
```

```
Welcome to mr.bob interactive mode. Before we generate directory structure, some questions need to be
```

```
Answer with a question mark to display help.
```

```
Value in square brackets at the end of the questions present default value if there is no answer.
```

```
--> How old are you? [24]:
```

```
--> What is your name?: Foobar
```

```
--> Enter password:
```

```
Generated file structure at /current/directory/
```

## 1.4 Listing all questions needed to have corresponding variable for a template

```
$ mrbob --list-questions mrbob:template_sample/
author.age.default = 24
author.age.help = We need your age information to render the template
author.age.question = How old are you?
author.name.question = What is your name?
author.name.required = True
author.password.command_prompt = getpass:getpass
author.password.question = Enter password
```

## 1.5 Remember answers to a config file

Running:



```
$ mrbob --remember-answers -O new_dir mrbob:template_sample/
...
```

When everything is done, all answers are stored in **new\_dir/.mrbob.ini** so later you reuse them:

```
$ mrbob --config new_dir/.mrbob.ini -O new_dir another_template/
...
```

## 1.6 Using non-interactive mode

Sometimes you might want to automate a script and use `mrbob`. It is wise to tell `mrbob` to not prompt for any input. `mrbob` will use given answers and defaults if answers are missing. In case a question is required and doesn't have a default, error will be thrown.

## 1.7 Configuration

Configuration is done with `.ini` style files. There are two sections for configuration: `:term:mr.bob` and `:term:variables`.

Example of global config file `~/mrbob` or command line parameter `mrbob --config foo.ini`.

```
[mr.bob]
verbose = True

[variables]
author.name = Domen Kožar
author.email = domen@dev.si
```

### 1.7.1 Specifying answers

To answer some questions from a config file instead of interactively. Given `me.ini`:

```
[variables]
author.name = Domen Kožar
author.email = domen@dev.si
author.age = 24
```

do:

```
$ mrbob --config me.ini mrbob:template_sample/
```

### 1.7.2 Specifying defaults

Sometimes you might want to override defaults for a template. Given `me.ini`:

```
[defaults]
author.name = Domen Kožar
author.email = domen@dev.si
author.age = 24
```

do:

```
$ mrbob --config me.ini mrbob:template_sample/
```

mrbob will ask you questions but default values will be also taken from config file.

### 1.7.3 Remote configuration

Config file can also be loaded from a remote location:

```
$ mrbob --config https://raw.githubusercontent.com/iElectric/mr.bob/master/mrbob/tests/example.ini mrbob:template_sample/
```

### 1.7.4 Configuration inheritance

Configuration can be specified in multiple ways. See flow of mr.bob on the documentation front page to know how options are preferred.

### 1.7.5 Nesting variables into namespaces called groups

All variables can be specified in namespaces, such as `author.name`. Currently namespaces don't do anything special besides providing readability.

### 1.7.6 mr.bob section reference

| Parameter                        | Default     | Explanation   |
|----------------------------------|-------------|---|
| <code>ignored_files</code>       | No patterns | Multiple Unix-style patterns to specify which files should be ignored: for instance, to ignore Vim swap files, specify <code>*.swp</code>       |
| <code>ignored_directories</code> | No patterns | Multiple Unix-style patterns to specify which directories should be ignored: for instance, to ignore a Git directory, specify <code>.git</code> |
| <code>non_interactive</code>     | False       | Don't prompt for input. Fail if questions are required but not answered   |
| <code>quiet</code>               | False       | Don't output anything except necessary  |
| <code>remember_answers</code>    | False       | Write answers to <code>.mrbob.ini</code> file inside output directory   |
| <code>verbose</code>             | False       | Output more information, useful for debugging   |

## 1.8 Collection of community managed templates

You are encouraged to use the `bobtemplates.something` Python egg namespace to write templates and contribute them to this list by making a [pull request](#).

- [bobtemplates.ielectric](#)
- [bobtemplates.kotti](#)
- [bobtemplates.niteoweb](#)

---

## Writing your own template

---

### 2.1 Starting

Writing your own template is as easy as creating a *.mrbob.ini* that may contain questions. Everything else is extra. To start quickly, use the template starter that ships with *mr.bob*:

```
$ mrbob mrbob:template_starter/  
Welcome to mr.bob interactive mode. Before we generate directory structure, some questions need to be
```

Answer with a question mark to display help.

Value in square brackets at the end of the questions present default value if there is no answer.

```
--> How old are you? [24]:
```

```
--> What is your name?: Foobar
```

```
--> Enter password:
```

```
Generated file structure at /home/ielectric/code/mr.bob
```

See *.mrbob.ini* for sample questions and *sample.txt.bob* for sample rendering.

### 2.2 How it works

Files inside the structure can be just copied to destination, or they can be suffixed with *.bob* and the templating engine will be used to render them.

By default a slightly customized *Jinja2* templating is used. The big differences are that variables are referenced with `{{{ variable }}}` instead of `{{ variable }}` and blocks are `{{% if variable %}}` instead of `{% if variable %}`. To read more about templating see [Jinja2 documentation](#).

Variables can also be used on folder and file names. Surround variables with plus signs. For example *foo/+author+//age+.bob* given variables *author* being *Foo* and *age* being *12*, *foo/Foo/12* will be rendered.

Templating engine can be changed by specifying *renderer* in *mr.bob* config section in *dotted notation*. It must be a callable that expects a text source as the first parameter and a dictionary of variables as the second.

When rendering the structure, permissions will be preserved for files.

## 2.3 Writing Questions

[*question*] section in *.mrBob.ini* specifies a *schema* for how [*variables*] are validated. Example speaks for itself:

### [questions]

```
author.name.question = What is your name?
author.name.required = True

author.age.question = How old are you?
author.age.help = We need your age information to render the template
author.age.default = 24

author.password.question = Enter password
author.password.command_prompt = getpass:getpass
```

Questions will be asked in the order written in *.mrBob.ini*.

### 2.3.1 questions section reference

| Parameter         | Default     | Explanation   |
|-------------------|-------------|---|
| name              |             | Required. Unique identifier for the question  |
| question          |             | Required. Question given interactively to a user when generating structure              |
| default           | None        | Default value when no answer is given. Can be a <i>dotted notation</i>                  |
| required          | False       | Specify if question must be answered  |
| command_prompt    | raw_input() | Function that accepts a question and asks user for the answer                           |
| help              | ""          | Extra help returned when user inputs a question mark                                    |
| pre_ask_question  | None        | <i>dotted notation</i> function to run before asking the question                       |
| post_ask_question | None        | <i>dotted notation</i> function to run after asking the question (also does validation) |

## 2.4 Common needs for templating

### 2.4.1 Default value of the question is dynamic

Use something like:

### [questions]

```
author.name.question = What's your name?
author.name.pre_ask_question = bobtemplates.mytemplate.hooks:pre_author
```

Where *pre\_author* function will modify the question and provide new `mrBob.configurator.Question.default`.

### 2.4.2 Conditionally render a file

Use something like:

### [template]

```
post_render = bobtemplates.mytemplate.hooks:delete_readme
```

And based on *mrBob.Configurator.variables* answers, delete a file or add one.

### 2.4.3 Based on the answer of the question do something

Use something like:

```
[questions]
author.name.question = What's your name?
author.name.post_ask_question = bobtemplates.mytemplate.hooks:post_author
```

Where `post_author` function will take `mrbob.configurator.Configurator`, `question` and it's answer.

### 2.4.4 Ask a question based on answer of previous question

use `post_ask_question` and add another question (is that possible if we are looping through questions? -> While questions: questions.pop())

## 2.5 Hooks

A list of places where you can hook into the process flow and provide your custom code. All hooks can have multiple entries limited by whitespace.

### 2.5.1 Post render hook

If you would like to execute a custom Python script after rendering is complete, you can use `post_render` hook in your `.mrbob.ini`.

```
[template]
post_render = bobtemplates.mytemplate.hooks:my_post_render_function
```

This assumes you have a `bobtemplate.mytemplate` egg with a `hooks.py` module. This module contains a `my_post_render_hook` function, which gets called after `mr.bob` has finished rendering your template.

The function expects one argument (`mrbob.configurator.Configurator`) and looks something like this:

```
def my_post_render_function(configurator):
    if configurator.variables['author.email']:
        # do something here
```

### 2.5.2 Pre render hook

Much like the *Post render hook* example above, you can use `pre_render` variable in your `.mrbob.ini` to specify a function to call before rendering starts.

```
[template]
pre_render = bobtemplates.mytemplate.hooks:my_pre_render_function
```

### 2.5.3 Pre question hook

For maximum flexibility, *mr.bob* allows you to set hooks to questions. Using `pre_ask_question` in your `.mrbob.ini` allows you to run custom code before a certain question.

**The function expects two arguments:**

- `mrBob.configurator.Question`
- `mrBob.configurator.Configurator`

**[questions]**

```
author.name.question = What's your name?
author.name.pre_ask_question = bobtemplates.mytemplate.hooks:pre_author
```

```
def set_fullname(configurator, question):
    question.default = 'foobar'
```

If you want question to be skipped, simply raise `mrBob.bobexceptions.SkipQuestion` inside your hook.

### 2.5.4 Post question hook

Similar to *Pre question hook* example above, you can use `post_ask_question` variable in your `.mrBob.ini` to specify a function to call after a question has been asked. *Post question hook* **must** return the answer of the question.

**The function expects three arguments:**

- `mrBob.configurator.Question`
- `mrBob.configurator.Configurator`
- answer from the question

**[questions]**

```
author.firstname.question = What's your name?
author.lastname.question = What's your surname?
author.lastname.post_ask_question = bobtemplates.mytemplate.hooks:set_fullname
```

```
def set_fullname(configurator, question, answer):
    configurator.variables['author.fullname'] =
        configurator.variables['author.firstname'] + ' ' +
        answer
    return answer
```

Raise `mrBob.bobexceptions.ValidationError` to re-ask the question.

### 2.5.5 Hooks shipped with *mrBob*

See `mrBob.hooks`.

## 2.6 template section reference

| Parameter                | Default                                      | Explanation   |
|--------------------------|--|---|
| <code>renderer</code>    | <code>mrBob.rendering:jinja2_renderer</code> | Function for rendering templates in <i>dotted notation</i>            |
| <code>pre_render</code>  | None   | <i>dotted notation</i> function to run before rendering the templates |
| <code>post_render</code> | None   | <i>dotted notation</i> function to run after rendering the templates  |

---

## Design goals

---

- Cover 80% of use cases, don't become too complex
- Ability to use templates not only from eggs, but also folders and similar
- Python 3 support
- Jinja2 renderer by default, but replaceable
- Ability to render multiple templates to the same target directory





---

### Why another tool

---

- PasteScript is a big package with lots of legacy code and noone seems to care about maintaining it (and porting it to python3)
- a tool should do one thing and that thing good, which is where PasteScript fails
- PasteScript works only with Python eggs, mr.bob can also render templates from folder and zip files
- PasteScript uses Cheetah which doesn't work on PyPy and has C extensions that need to be compiled
- PasteScript is unmaintainable, with really dodgy code
- PasteScript doesn't preserve permissions when copying/rendering files
- mr.bob is just 200 lines of code with some extra features in mind that PasteScript cannot provide, such as a Python API for use by higher level libraries



---

## Developer guide

---

### 5.1 Setup developer environment

```
$ git clone https://github.com/iElectric/mr.bob.git
$ cd mrbob
$ virtualenv .
$ source bin/activate
$ python setup.py develop
$ easy_install mr.bob[test,development]
$ mrbob --help
```

### 5.2 Running tests

Easy as:

```
$ make test
```

### 5.3 Making a Release

Using *zest.releaser*:

```
$ bin/fullrelease
```



---

## Source documentation

---

**Warning:** Python API is far from being “frozen”, use it with zero backwards-compatibility in mind. You are welcome to report suggestions to bug tracker.

### 6.1 mrbob – Main package

#### 6.1.1 mrbob.configurator – Machinery to figure out configuration

`class mrbob.configurator.Configurator` (*template*, *target\_directory*, *bobconfig=None*, *variables=None*, *defaults=None*)

Bases: object

Controller that figures out settings, asks questions and renders the directory structure.

##### Parameters

- **template** – Template name
- **target\_directory** – Filesystem path to a output directory
- **bobconfig** – Configuration for mr.bob behaviour
- **variables** – Given variables to questions
- **defaults** – Overriden defaults of the questions

Additional to above settings, *Configurator* exposes following attributes:

- `template_dir` is root directory of the template
- `is_tempdir` if template directory is temporary (when using zipfile)
- `templateconfig` dictionary parsed from *template* section
- `questions` ordered list of `Question` instances to be asked
- `bobconfig` dictionary parsed from *mrbob* section of the config

**ask\_questions()**

Loops through questions and asks for input if variable is not yet set.

**render()**

Render file structure given instance configuration.  
`mrbob.rendering.render_structure()`.

Basically calls

```
class mrbob.configurator.Question(name, question, default=None, required=False,
                                  command_prompt=<built-in function raw_input>,
                                  pre_ask_question='', post_ask_question='', help='', **extra)
```

Bases: object

Question configuration. Parameters are used to configure questioning and possible validation of the answer.

#### Parameters

- **name** – Unique, namespaced name of the question
- **question** – Question to be asked
- **default** – Default value of the question
- **required** (*bool*) – Is question required?
- **command\_prompt** – Function to executed to ask the question given question text
- **help** – Optional help message
- **pre\_ask\_question** – Space limited functions in dotted notation to ask before the question is asked
- **post\_ask\_question** – Space limited functions in dotted notation to ask aster the question is asked
- **\*\*extra** – Any extra parameters stored for possible extending of *Question* functionality

Any of above parameters can be accessed as an attribute of *Question* instance.

**ask** (*configurator*)

Eventually, ask the question.

**Parameters** *configurator* – `mrbob.configurator.Configurator` instance

`mrbob.configurator.parse_template` (*template\_name*)

Resolve template name into absolute path to the template and boolean if absolute path is temporary directory.

## 6.1.2 mrbob.cli – Command line interface

Command line interface to mr.bob

```
mrbob.cli.main(args=['-b', 'latex', '-D', 'language=en', '-d', '_build/doctrees', '.', '_build/latex'])
```

Main function called by *mrbob* command.

## 6.1.3 mrbob.bobexceptions – Exceptions

mr.bob exceptions module.

**exception** `mrbob.bobexceptions.ConfigurationError`

Bases: `mrbob.bobexceptions.MrBobError`

Raised during configuration phase

**exception** `mrbob.bobexceptions.MrBobError`

Bases: `exceptions.Exception`

Base class for errors

**exception** `mrbob.bobexceptions.SkipQuestion`

Bases: `mrbob.bobexceptions.MrBobError`

Raised during `pre_ask_question` if we should skip it

**exception** `mrbob.bobexceptions.TemplateConfigurationError`

Bases: `mrbob.bobexceptions.ConfigurationError`

Raised reading template configuration

**exception** `mrbob.bobexceptions.ValidationError`

Bases: `mrbob.bobexceptions.MrBobError`

Raised during question validation

## 6.1.4 `mrbob.parsing` – Parsing `.ini` files

## 6.1.5 `mrbob.rendering` – Everything related to rendering templates and directory structure

`mrbob.rendering.render_structure` (*fs\_source\_root, fs\_target\_root, variables, verbose, renderer, ignored\_files, ignored\_directories*)

Recursively copies the given filesystem path *fs\_source\_root* to a target directory *fs\_target\_root*.

Any files ending in `.bob` are rendered as templates using the given `renderer` using the variables dictionary, thereby losing the `.bob` suffix.

strings wrapped in `+ signs` in file- or directory names will be replaced with values from the variables, i.e. a file named `+name+.py.bob` given a dictionary `{'name': 'bar'}` would be rendered as `bar.py`.

## 6.1.6 `mrbob.hooks` – Included hooks

Use any of hooks below or write your own. You are welcome to contribute them!

`mrbob.hooks.set_current_datetime` (*configurator, question*)

If you want to set the default answer of a question to the current date and time, use this function as *Pre question hook*:

```
[questions]
project.question = What year was the project started?
project.pre_ask_question = mrbob.hooks:set_current_datetime
project.datetime_format = %%Y
```

The `datetime_format` property should be of the standard Python `strftime` format. It defaults to `YYYY-MM-DD` if not specified.

Please note that you'll have to escape the `%` character (by using `%%`) due to the fact it's a special character in INI files.

See the following URL for more information: <http://docs.python.org/2/library/datetime.html#strftime-and-strptime-behavior>

`mrbob.hooks.show_message` (*configurator*)

If you want to display a message to the user when rendering is complete, you can use this function as *Post render hook*:

```
[template]
post_render = mrbob.hooks:show_message
message = Well done, %(author.name)s, your code is ready!
```

As shown above, you can use standard Python formatting in `post_render_msg`.

`mrbob.hooks.to_boolean` (*configurator, question, answer*)

If you want to convert an answer to Python boolean, you can use this function as *Post question hook*:

```
[questions]
idiot.question = Are you young?
idiot.post_ask_question = mrbob.hooks:to_boolean
```

Following variables can be converted to a boolean: **y, n, yes, no, true, false, 1, 0**

`mrbob.hooks.to_decimal` (*configurator, question, answer*)

If you want to convert an answer to Python float, you can use this function as *Post question hook*:

```
[questions]
daemon.question = What interval should the daemon poll for data?
daemon.post_ask_question = mrbob.hooks:to_decimal
```

`mrbob.hooks.to_integer` (*configurator, question, answer*)

If you want to convert an answer to Python integer, you can use this function as *Post question hook*:

```
[questions]
owner.question = What's your age?
owner.post_ask_question = mrbob.hooks:to_integer
```

`mrbob.hooks.validate_choices` (*configurator, question, answer*)

If you want to validate an answer using a limited set of choices, you can use this function as *Post question hook*:

```
[questions]
license.question = What license would you like to use?
license.post_ask_question = mrbob.hooks:validate_choices
license.choices = MIT|BSD|Apache 2.0|Other
license.choices_case_sensitive = yes
license.choices_delimiter = |
```

Currently choices are split using whitespace by default. If you wish to have whitespace within each choice, you may specify a custom delimiter which will be used to split the choices.

This hook may be set to validate the choices in a case sensitive manner. However, this behaviour is disabled by default.

`mrbob.hooks.validate_datetime` (*configurator, question, answer*)

If you want to validate a date using a chosen date format, you can use this function as *Post question hook*:

```
[questions]
project.question = What year was the project started?
project.post_ask_question = mrbob.hooks:validate_datetime
project.datetime_format = %%Y
```

The `datetime_format` property should be of the standard Python `strftime` format. It defaults to `YYYY-MM-DD` if not specified.

Please note that you'll have to escape the `%` character (by using `%%`) due to the fact it's a special character in INI files.

See the following URL for more information: <http://docs.python.org/2/library/datetime.html#strftime-and-strptime-behavior>

`mrbob.hooks.validate_regex` (*configurator, question, answer*)

If you want to validate an answer using a regular expression, you can use this function as *Post question hook*:



**[questions]**

```
project.question = Please specify a name (only lowercase characters)?  
project.post_ask_question = mrbob.hooks:validate_regex  
project.regex = ^[a-z]+$
```



---

## Changelog

---

### 7.1 0.2 (unreleased)

- Nothing changed yet.

### 7.2 0.1.1 (2014-03-03)

- Claim Python 3.4 support [Domen Kožar]
- Fix encoding errors on Python 3 during installation [Domen Kožar]
- Use Jinja2 < 2.7 only for python 3.2 [Domen Kožar]

### 7.3 0.1 (2014-03-02)

- move exceptions to bobexceptions [Jean-Philippe Camguilhem]
- Use jinja2 < 2.7, since 2.7+ doesn't support Python 3.2 [Domen Kožar]
- Filename variable name substitution regex failed to compile on windows [Domen Kožar]
- Do not copy `.DS_Store`. [Godefroid Chapelle]
- Configure patterns of files to ignore through `ignored_files` option of `mr.bob` section. [Godefroid Chapelle]
- Configure patterns of directories to ignore through `ignored_directories` option of `mr.bob` section. [Fotis Gimian]
- Added new post-question hook `mrbob.hooks.to_integer` [Fotis Gimian]
- Added new post-question hook `mrbob.hooks.to_decimal` [Fotis Gimian]
- Added new post-question hook `mrbob.hooks.validate_choices` [Fotis Gimian]
- Added new post-question hook `mrbob.hooks.validate_regex` [Fotis Gimian]
- Added new pre-question hook `mrbob.hooks.set_current_datetime` [Fotis Gimian]
- Added new post-question hook `mrbob.hooks.validate_datetime` [Fotis Gimian]

## 7.4 0.1a9 (2013-04-26)

- Regex to detect variable names when rendering file names was broken when directory path contains pluses. [Godefroid Chapelle]

## 7.5 0.1a8 (2013-03-11)

- Depend on six>=1.2.0 [Domen Kožar]
- Moved all exceptions to *mrbob.exceptions* module [Domen Kožar]
- Fix loading of zip files [Domen Kožar]
- #28: Remote loading of config files [Nejc Zupan]
- #30: Keep newlines of rendered template [Domen Kožar]

## 7.6 0.1a7 (2013-01-23)

- Don't depend on argparse in python 2.7 and higher, since it's already in stdlib [Domen Kožar]
- #22: Prevent users from specifying target directory inside template dir [Domen Kožar]

## 7.7 0.1a6 (2013-01-02)

- Use `StrictUndefined` with jinja2 renderer so that any missing key is reported as an error [Domen Kožar]
- If a key in a namespace was missing while rendering, no error was raised [Domen Kožar]
- Added hook `mrbob.hooks.show_message` [Domen Kožar]
- `mrbob.validators.boolean` renamed to `mrbob.hooks.to_boolean` [Domen Kožar]
- Renamed `validators.py` to `hooks.py` [Domen Kožar]
- Removed `validators` and `action` settings from `[questions]` as it is superseded by `hooks` [Domen Kožar]
- Added `pre_ask_question` and `post_ask_question` to `[questions]` section [Domen Kožar]
- Added `pre_render`, `post_render` and `post_render_msg` options [Domen Kožar]
- Added `[defaults]` section that will override template defaults. The only difference to `[variables]` is that variables provide default answers [Domen Kožar]
- Moved `renderer` parameter to `[template]` section [Domen Kožar]
- Added `[template]` section that is parsed only from `.mrbob.ini` inside a template directory. [Domen Kožar]
- Correctly evaluate boolean of `quiet` and `verbose` settings [Domen Kožar]
- Added `non_interactive` setting that will not prompt for any input and fail if any of required questions are not answered [Domen Kožar]
- Added `remember_answers` setting that will create `.mrbob.ini` file inside rendered directory with all the answers written to `[variables]` section [Domen Kožar]

- Include changelog in documentation [Domen Kožar]
- `Question` does no longer raise error if unknown parameter is passed from a config file. Instead those parameters are saved to `question.extra` that can be later inspected and validated. This is first step to have advanced question types such as question with a set of predefined answers [Domen Kožar]
- Rewrite all `py.test` stuff to `nosetests`, so we have unified testing now. This also fixes `flake8` segfaults on `pypy` [Domen Kožar]

## 7.8 0.1a5 (2012-12-12)

- #26: Variables were not correctly parsed from config files [Domen Kožar]

## 7.9 0.1a4 (2012-12-11)

- Fix `MANIFEST.in` so that template examples are also included with distribution [Domen Kožar]
- Add `-q/--quiet` option to suppress output which isn't strictly necessary [Sasha Hart]
- Suppress the interactive-mode welcome banner if there are no questions to ask [Sasha Hart]
- Don't raise `KeyError`: 'questions\_order' if [questions] is missing in an ini [Sasha Hart]

## 7.10 0.1a3 (2012-11-30)

- #13: Read user config from `~/.mrBob` (as stated in docs and inline comments). [Andreas Kaiser]

## 7.11 0.1a2 (2012-11-29)

- #12: Fix unicode errors when using non-ASCII in questions or defaults [Domen Kožar]
- Ask questions in same order they were defined in template configuration file [Domen Kožar]

## 7.12 0.1a1 (2012-10-19)

- Initial release. [Domen Kožar, Tom Lazar]



---

## Glossary

---

**dotted notation** Importable Python function specified with dots as importing a module separated with a colon to denote a function. For example *mrbob.rendering:render\_structure*

**mr.bob** This section configures how *mrbob* behaves

**variables** This section answers to the questions that will be passed to templates for rendering





---

## Indices and tables

---

- *genindex*
- *modindex*
- *search*



## m

mr**bob**, 17  
mr**bob**.bobexceptions, 18  
mr**bob**.cli, 18  
mr**bob**.configurator, 17  
mr**bob**.hooks, 19  
mr**bob**.parsing, 19  
mr**bob**.rendering, 19