
mrbob Documentation

Release 0.1a1

Domen Kožar, Tom Lazar

October 19, 2012

CONTENTS

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

Source code [github.com](#) project

Bug tracker [github.com](#) issues

License BSD

Generated October 19, 2012

Version 0.1a1

Features

- provide questions needed to be answered for rendering structure
- questions can be grouped by using a namespace
- render templates from a folder or Python egg or zip file
- supports Python 2.6 - 3.3, pypy
- 100% test coverage
- uses Jinja2 as default rendering engine (can be replaced)
- multiple ways to specify variables to render templates
- preserves permissions when rendering templates

Introduction

mr.bob is a tool that takes a directory skeleton and copies over directory structure to target folder and might use rendering engine [Jinja2](#) to dynamically generate the files. Additionally, you might be asked some 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](#).

USER GUIDE

1.1 Installation

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

1.2 Usage

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

```
$ mrbob --help
usage: mrbob [-h] [-O TARGET_DIRECTORY] [-c CONFIG] [-V] [-l] [-r RENDERER]
           [template]
```

Filesystem template renderer

positional arguments:

 template Template to use **for** rendering

optional arguments:

 -h, --help show this **help** message and **exit**
 -O TARGET_DIRECTORY, --target-directory TARGET_DIRECTORY
 Where to output rendered structure. Defaults to
 current directory.
 -c CONFIG, --config CONFIG
 Configuration file to specify either [mr.bob] or
 [variables] sections.
 -V, --version Display version number
 -l, --list-questions List all questions needed **for** the template
 -r RENDERER, --renderer RENDERER
 Dotted notation to a renderer **function**. Defaults to
 mrbob.rendering:jinja2_renderer

By default target directory is current folder. 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 zip file:

`https://github.com/iElectric/mr.bob/zipball/master`

Or from a relative path in a zip file:

`https://github.com/iElectric/mr.bob/zipball/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 answered.

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 Configuration

Configuration is done with *.ini* style files. There are two sections for configuration.

mr.bob configures how *mr.bob* behaves

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

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

```
[mr.bob]
renderer = moo.foo:render_mako
```

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

1.5.1 Configuration inheritance

Configuration can be specified in multiple ways. Here is a graph how settings are preferred (questions override any other configuration).

```

Global config at ~/.mrbob
      ^
      |
mrbob --config mrbob.ini
      ^
      |
Questions answered in interactive mode
  
```

1.5.2 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.

1.5.3 `mr .bob` section reference

Parameter	Default	Explanation
renderer	mrbob.rendering:jinja2_renderer	Function for rendering templates
verbose	False	Output more information, useful for debugging

1.6 Collection of community managed templates

You are encouraged to use *bobtemplates.something* Python egg namespace to write templates and contribute them to this list by making a pull request.

- [bobtemplates.ielectric](#)

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 template starter that ships with *mr.bob*:

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

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 Templating

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

By default a little bit customized *Jinja2* templating is used. The big difference is 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 text source as first parameter and dictionary of variables as second.

When rendering structure, permissions will be preserved for files.

2.3 Writing Questions

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

[questions]

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

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
action	lambda x: x	Extra action to be taken except returning value to be used stored in variables
validator	None	Validator can raise <code>mrbbob.configurator.ValidationError</code> and question will be asked again
command_prompt	<code>raw_input()</code>	Function that accepts question and asks user for the answer
help	""	Extra help returned when user inputs a question mark

2.4 Validators

Validators are functions with answer as only parameter. They may return value to be used as an answer and may raise `ValidationError` for question to be asked again.

See `mrbbob.validators` for validators that ship with *mr.bob*.

DEVELOPER GUIDE

3.1 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
```

3.2 Running tests

Easy as:

```
$ ./bin/test
```

3.3 Making a Release

Using *zest.releaser*:

```
$ bin/fullrelease
```


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 in future maybe from http links
- 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 Python API to be used by higher level libraries

SOURCE DOCUMENTATION

6.1 mrbob – Main package

6.1.1 mrbob.configurator – Machinery to figure out configuration

exception `mrbob.configurator.ConfigurationError`

Bases: `mrbob.configurator.MrBobError`

Raised during configuration phase

class `mrbob.configurator.Configurator` (*template, target_directory, bobconfig=None, variables=None*)

Bases: `object`

Controller that figures out settings and renders file structure.

Parameters

- **template** – Template name
- **target_directory** – Filesystem path to a output directory
- **bobconfig** – Configuration for mr.bob behaviour
- **variables** – Given variables

ask_questions()

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

render()

Render file structure given instance configuration.

Basically calls

`mrbob.rendering.render_structure()`.

exception `mrbob.configurator.MrBobError`

Bases: `exceptions.Exception`

Base class for errors

class `mrbob.configurator.Question` (*name, question, default=None, required=False, action=<function <lambda> at 0x2db6230>, validator=None, command_prompt=<built-in function raw_input>, help=''*)

Bases: `object`

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

ask()

Eventually, ask the question.

exception `mrbbob.configurator.TemplateConfigurationError`

Bases: `mrbbob.configurator.ConfigurationError`

Raised reading template configuration

exception `mrbbob.configurator.ValidationError`

Bases: `mrbbob.configurator.MrBobError`

Raised during question validation

`mrbbob.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 `mrbbob.cli` – Command line interface

Command line interface to `mr.bob`

`mrbbob.cli.main(args=['-b', 'latex', '-d', '_build/doctrees', '.', '_build/latex'], quiet=False)`

Main function called by `mrbbob` command.

6.1.3 `mrbbob.parsing` – Parsing `.ini` files

`mrbbob.rendering.render_structure(fs_source_root, fs_target_root, variables, verbose, renderer)`

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`.

6.1.4 `mrbbob.rendering` – Everything related to rendering templates and directory structure

`mrbbob.rendering.render_structure(fs_source_root, fs_target_root, variables, verbose, renderer)`

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`.

6.1.5 `mrbbob.validators` – Useful validators for questions

`mrbbob.validators.boolean(value)`

Converts value to Python boolean given values: y, n, yes, no, true, false, 1, 0

GLOSSARY

dotted notation Importable Python function specified with dots as importing a module separated with a column to denote a function. For example *mrBob.rendering:render_structure*

INDICES AND TABLES

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

PYTHON MODULE INDEX

m

- `mrbbob, ??`
- `mrbbob.cli, ??`
- `mrbbob.configurator, ??`
- `mrbbob.rendering, ??`
- `mrbbob.validators, ??`