projects: A simple wrapper for setuptools

Release 0.1-1

Robert J. Gautier

Dec 05, 2020

Contents

Provides an alternative to the `setuptools.setup()` function that defines a Python project.

This package is very specific to how I like to build projects, and I doubt it will be of much use to anyone else.

I'm offering a brief writeup here because you might come across it if you look at my other projects, so it needs an explanation.

In order to be able to build other *rjgtoys* projects you'll probably need to install this one first.

Once you've done that, the *setup.py* from other *rjgtoys* projects should work more or less as you expect.

CHAPTER 1

Getting it

1.1 Method 1: Install from PyPi

By the time you read this, `rjgtoys-projects` should be available on PyPi:

```
pip install --user rjgtoys-projects
```

1.2 Method 2: Get the source code

To get the source code:

```
git clone https://github.com/bobgautier/rjgtoys-projects.git
```

Then make the package available for your Python:

```
cd rjgtoys-projects
python ./setup.py develop --user
```

If you are using a virtualenv, you should omit the `--user` option used in these examples.

CHAPTER 2

Starting a new project: *rjgtoys-new*

This command line tool simplifies starting a new project; it creates an empty project tree based on a few parameters you provide either in a configuration file or on the command line.

2.1 Command line options

Use `--help` to get a list of options:

```
$ rjgtoys-new --help
usage: Build a new rjgtoys project [-h] [--name NAME] [--id IDENT] --title TEXT [--
↪template PATH] [--update] [--config PATH] [--dry-run]

optional arguments:
  -h, --help       show this help message and exit
  --name NAME      Public name of the project, for docs etc.
  --id IDENT       Internal name of the project, for directories etc.
  --title TEXT     One-line title for the project
  --template PATH  Template name or path (default: rjgtoys)
  --update         Update an existing project (default: False)
  --config PATH    Configuration to use (default: $HOME/.config/rjgtoys/projects/
↪projects.conf)
  --dry-run, -n    Don't generate anything, just validate (default: False)
```

--id A short name for the project, used for directories, files, and so on. This name will be used for the main package source directory of the project.

--name A 'documentation name' for the project; usually very similar or identical to the *id*, but sometimes capitalised, or slightly modified in some other way.

For example, *rjgtoys-xc* (id: *xc*) has the *name XC* because I think that looks better in documentation.

--title A one-line description of the project.

--template The name of a template set to use for the project. This is a path, and relative paths are interpreted relative to a `templates` directory internal to *rjgtoys-projects*.

If you want to experiment with this, be prepared to learn about Cookiecutter[1].

--update If this option is used, `rjgtoys-new` will update an existing directory (and might therefore lose you some work).

--config Specifies a configuration file from which to get further parameters for the project. The configuration file is described below.

--dry-run If used, no files or directories will be created or updated.

The project tree is created (or updated) in a directory named with the *id* you specified.

2.2 Configuration file

```
#
# Example prototype projects.conf
#
# Make an accurate version of this file and put it in
#
#   ~/.config/rjgtoys/projects/projects.conf
#

project:
  family: rjgtoys

copyright:
  year: 2020

author:
  fullname: "Robert J. Gautier"
  email: "bob.gautier@gmail.com"

github:
  base: https://github.com/bobgautier/
```

The file is YAML.

The parameters are as follows. For brevity I've used dotted names here, to indicate the paths through the nested mappings that are expected in the configuration file.

project.family Name of the group to which the project belongs. The default is *rjgtoys*, resulting in projects called `rjgtoys-ID`.

If you start using this tool yourself, please use a different family name; I chose *rjgtoys* in the hope of avoiding a clash with someone else.

copyright.year The year to include in copyright notices.

author.fullname Full name of the author.

author.email Email address of author.

github.base Base URL for github repos; project repositories will be expected to be 'under' this URL.

[1] https://cookiecutter.readthedocs.io/en/1.7.2/

CHAPTER 3

Using *setup.py*

My version of `setup()` adds some new commands, and customises some existing ones.

3.1 Extended 'test' command

The 'test' command uses *pytest* and *coverage*, so that an HTML coverage report is generated.

You will find the coverage report at `htmlcov/index.html` in your project tree.

The command goes to some trouble to enumerate all the Python source code files in your project tree, so that they are included in the coverage report even if they are not referenced by any tests. This is to avoid making coverage look good when only a few modules have any tests at all.

Using *setuptools* to run tests is now deprecated, so eventually I will have to find a different way to do this.

3.2 'lint' command

The 'lint' command runs *pylint*.

The configuration is embedded in *rjgtoys-projects* (so that it's standard for all my projects).

The report is left in the project directory, in `pylint.out`.

3.3 'jenkins' command

The 'jenkins' command can create a Jenkins job to build the project.

This command is just an old experimental proof-of-concept.

CHAPTER 4

Credits

The templating is all done with Cookiecutter[2].

[2] https://cookiecutter.readthedocs.io/en/1.7.2/