commonutilslib Documentation

Release 0.1.4

Costas Tyfoxylos

Jun 08, 2021

Contents

Contents:

CHAPTER 1

commonutilslib

A library with some common utility methods for python like tempdir and Pushd

- Documentation: https://common-utils.readthedocs.io/en/latest/usage.html

1.1 Development Workflow

The workflow supports the following steps

- lint
- test
- build
- document
- upload
- graph

These actions are supported out of the box by the corresponding scripts under _CI/scripts directory with sane defaults based on best practices. Sourcing setup_aliases.ps1 for windows powershell or setup_aliases.sh in bash on Mac or Linux will provide with handy aliases for the shell of all those commands prepended with an underscore.

The bootstrap script creates a .venv directory inside the project directory hosting the virtual environment. It uses pipenv for that. It is called by all other scripts before they do anything. So one could simple start by calling _lint and that would set up everything before it tried to actually lint the project

Once the code is ready to be delivered the _tag script should be called accepting one of three arguments, patch, minor, major following the semantic versioning scheme. So for the initial delivery one would call

> $ _tag –minor

which would bump the version of the project to 0.1.0 tag it in git and do a push and also ask for the change and automagically update HISTORY.rst with the version and the change provided.

So the full workflow after git is initialized is:

- repeat as necessary (of course it could be test - code - lint :))
  - code
  - lint
  - test
- commit and push
- develop more through the code-lint-test cycle
- tag (with the appropriate argument)
- build
- upload (if you want to host your package in pypi)
- document (of course this could be run at any point)

1.2 Important Information

This template is based on pipenv. In order to be compatible with requirements.txt so the actual created package can be used by any part of the existing python ecosystem some hacks were needed. So when building a package out of this **do not** simple call

> $ python setup.py sdist bdist_egg

as this will produce an unusable artifact with files missing. Instead use the provided build and upload scripts that create all the necessary files in the artifact.

1.3 Project Features

- Please refer to USAGE.rst

CHAPTER 2

Installation

At the command line:

```
$ pip install commonutilslib
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv commonutilslib
$ pip install commonutilslib
```

Or, if you are using pipenv:

```
$ pipenv install commonutilslib
```

CHAPTER 3

Usage

To develop on commonutilslib:

```
# The following commands require pipenv as a dependency

# To lint the project
_CI/scripts/lint.py

# To execute the testing
_CI/scripts/test.py

# To create a graph of the package and dependency tree
_CI/scripts/graph.py

# To build a package of the project under the directory "dist/"
_CI/scripts/build.py

# To see the package version
_CI/scipts/tag.py

# To bump semantic versioning [--major|--minor|--patch]
_CI/scipts/tag.py --major|--minor|--patch

# To upload the project to a pypi repo if user and password are properly provided
_CI/scripts/upload.py

# To build the documentation of the project
_CI/scripts/document.py
```

To use commonutilslib in a project:

Working with Pushd:

```
from commonutilslib import Pushd

import os
```

(continues on next page)

(continued from previous page)

```
with Pushd(some_path) as pushd:
    print(f'Doing things in {os.getcwd()} coming from {pushd.original_dir} and then␣
↪returning back')
```

Working with tempdir:

```
from commonutilslib import tempdir

import os
with tempdir():
    print(f'Doing things in temporary directory {os.getcwd()} and deleting after done
↪')
```

Working with Hasher:

```
from commonutilslib import Hasher

hasher = Hasher()
print(hasher.hash_file(some_file_name))
print(hasher.hash_directory(some_directory_name))
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

4.1 Submit Feedback

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.

4.1.1 Get Started!

Ready to contribute? Here's how to set up *commonutilslib* for local development. Using of pipenv is highly recommended.

1. Clone your fork locally:

   ```
   $ git clone https://github.com/schubergphilis/commonutilslib.git
   ```

2. Install your local copy into a virtualenv. Assuming you have pipenv installed, this is how you set up your clone for local development:

   ```
   $ cd commonutilslib/
   $ pipenv install --ignore-pipfile
   ```

3. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```

   Now you can make your changes locally. Do your development while using the CI capabilities and making sure the code passes lint, test, build and document stages.

4. Commit your changes and push your branch to the server:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

5. Submit a merge request

CHAPTER 5

commonutilslib

5.1 commonutilslib package

5.1.1 Submodules

5.1.2 commonutilslib.commonutilslib module

Main code for commonutilslib.

class `commonutilslib.commonutilslib.`**`Hasher`**(*buffer_size=65536*)

Bases: `object`

Calculated sha1 hashes for files and directories.

`hash_directory`(*path*)

Calculates the sha1 hash of the directory in the provided path.

Parameters **`path`** (*`str`*) – The path to calculate the digest for

Returns The digest of the path

Return type (str)

`hash_file`(*file_name*)

Calculates the sha1 hash of the provided file.

Parameters **`file_name`** (*`str`*) – The filename of the file to calculate the hash for

Returns The hash of the file provided

Return type (str)

class `commonutilslib.commonutilslib.`**`Pushd`**(*directory_name*)

Bases: `object`

Implements bash pushd capabilities.

`cwd = None`

original_dir = None

class commonutilslib.commonutilslib.**RecursiveDictionary**

Bases: `dict`

Implements recursively updating dictionary.

RecursiveDictionary provides the methods rec_update and iter_rec_update that can be used to update member dictionaries rather than overwriting them.

iter_rec_update(*iterator*)

Updates recursively.

rec_update(*other*, ***third*)

Implements the recursion.

Recursively update the dictionary with the contents of other and third like dict.update() does - but don't overwrite sub-dictionaries.

commonutilslib.commonutilslib.**cd**(*new_directory*, *clean_up=<function <lambda>>*)

Changes into a given directory and cleans up after it is done.

Parameters

- **new_directory** – The directory to change to
- **clean_up** – A method to clean up the working directory once done

commonutilslib.commonutilslib.**on_error**(*func*, *path*, *exc_info*)

Error handler for `shutil.rmtree`.

If the error is due to an access error (read only file) it attempts to add write permission and then retries.

If the error is for another reason it re-raises the error.

Usage : `shutil.rmtree(path, onerror=onerror)`

# 2007/11/08 # Version 0.2.6 # pathutils.py # Functions useful for working with files and paths. # http://www.voidspace.org.uk/python/recipebook.shtml#utils

# Copyright Michael Foord 2004 # Released subject to the BSD License # Please see http://www.voidspace.org.uk/python/license.shtml

# For information about bugfixes, updates and support, please join the Pythonutils mailing list. # http://groups.google.com/group/pythonutils/ # Comments, suggestions and bug reports welcome. # Scripts maintained at http://www.voidspace.org.uk/python/index.shtml # E-mail fuzzyman@voidspace.org.uk

commonutilslib.commonutilslib.**tempdir**()

Creates a temporary directory.

5.1.3 commonutilslib.commonutilslibexceptions module

Custom exception code for commonutilslib.

5.1.4 Module contents

commonutilslib package.

Import all parts from commonutilslib here

CHAPTER 6

Credits

6.1 Development Lead

- Costas Tyfoxylos <ctyfoxylos@schubergphilis.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

CHAPTER 8

0.0.1 (26-02-2019)

- First code creation

CHAPTER 9

0.1.0 (26-02-2019)

- First official release

CHAPTER 10

0.1.1 (18-10-2019)

- Update template and bumped dependencies.

CHAPTER 11

0.1.2 (18-10-2019)

- Bumped dependencies

CHAPTER 12

0.1.3 (26-04-2021)

- Bumped dependencies.

CHAPTER 13

0.1.4 (08-06-2021)

- Bumped dependencies.

CHAPTER 14

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index