pyscaffoldext-cookiecutter Documentation

Release 0.1.post1.dev36+gd769cb9

Anderson Bravalheri

Nov 28, 2022

CONTENTS

Combines the flexibility of **Cookiecutter** templates with the power of **PyScaffold**.

CHAPTER

ONE

CONTENTS

1.1 pyscaffoldext-cookiecutter

Extension that combines the flexibility of **Cookiecutter** templates with the power of **PyScaffold**.

Cookiecutter is a flexible utility that allows the definition of templates for a diverse range of software projects. On the other hand, PyScaffold is focused in a good out-of-the-box experience for developing distributable Python packages (exclusively). Despite the different objectives, it is possible to combine the power of both tools to create a custom Python project setup.

1.1.1 Quickstart

This extension can be directly installed with `pip`:

```
pip install pyscaffoldext-cookiecutter
```

Or, if you prefer `pipx`:

```
pipx install pyscaffold  # if you haven't installed pyscaffold yet
pipx inject pyscaffold pyscaffoldext-cookiecutter
```

Note that, after the installation, `putup -h` will show a new option `--cookiecutter TEMPLATE`. Use this option to point out which template you want to use (path or url). The file structure created by Cookiecutter will be refined by PyScaffold afterwards. For example:

```
putup my-proj1 --cookiecutter ~/my-templates/default
putup my-proj2 --cookiecutter gh:something/from-github
```

Please refer to Cookiecutter documentation for more details on possible URLs and abbreviations.

An additional option `--cookiecutter-params` is also added, so you can have more control over the values Cookiecutter uses when rendering the templates (PyScaffold will not run Cookiecutter's interactive prompt). This option takes the form of a space separated list of `NAME=VALUE` arguments as showed in the example bellow:

```
putup mypkg \
  --cookiecutter gh:pyscaffold/cookiecutter-pypackage \
  --cookiecutter-params command_line_interface=Argparse use_pytest=y
```

Check the `cookiecutter.json` file in the repository (or directory) of the template you are using to see the available parameters. Please notice PyScaffold already add some default parameters, as indicated in the section **Suitable Templates** bellow.

1.1.2 Cookiecutter templates with PyScaffold

The following example shows how to create a new package named `mypkg`, that uses a Cookiecutter template, but is enhanced by PyScaffold's features:

```
putup mypkg --cookiecutter gh:pyscaffold/cookiecutter-pypackage
```

This is roughly equivalent to first create a project using the Cookiecutter template and convert it to PyScaffold afterwards:

```
cookiecutter --no-input gh:pyscaffold/cookiecutter-pypackage project_name=mypkg
putup mypkg --force
```

Note: For complex Cookiecutter templates calling `cookiecutter` and `putup` separately may be a better option, since it is possible to answer specific template questions or at least set values for Cookiecutter variables.

Warning: Although using Cookiecutter templates is a viable solution to customize a project that was set up with PyScaffold, the recommended way is to help improve PyScaffold by contributing an extension.

Suitable templates

Note that PyScaffold will overwrite some files generated by Cookiecutter, like `setup.py`, the `__init__.py` file under the package folder and most of the `docs` folder, in order to provide setuptools-scm and sphinx integration. Therefore not all Cookiecutter templates are suitable for this approach.

Ideally, interoperable templates should focus on the file structure inside the `src` folder instead of packaging or distributing, since PyScaffold already handles it under-the-hood. This also means that your template should adhere to the src-layout if you want to generate files within your Python package.

In addition, PyScaffold runs Cookiecutter with the `--no-input` flag activated and thus the user is not prompted for manual configuration. Instead, PyScaffold injects the following parameters:

```
author
email
full_name => same as author
project_name => the name of the folder where the project will be generated
repo_name => same as project_name
package_name => putup's --package (as in `import`)
namespace => putup's --namespace (if any)
installable_name => putup's --name (an installable name, like in PyPI/pip install)
project_short_description => putup's description
```

(continues on next page)

(continued from previous page)

```
release_date => equivalent to the day you are running putup
year => equivalent to the year you are running putup
```

Any extra parameter should be passed using the `--cookiecutter-params` option.

Accordingly, the template file structure should be similar to:

```
cookiecutter-something/
└── {{cookiecutter.project_name}}/
    └── src/
        └── {{cookiecutter.package_name}}/
            └── ...
```

See Cookiecutter for more information about template creation.

Note: PyScaffold uses Cookiecutter only for its ability to create files. Pre/post hooks that perform any other kind of side effect are not guaranteed to work.

1.1.3 Making Changes & Contributing

This project uses pre-commit, please make sure to install it before making any changes:

```
pip install pre-commit
cd pyscaffoldext-cookiecutter
pre-commit install
```

It is a good idea to update the hooks to the latest version:

```
pre-commit autoupdate
```

Please also check PyScaffold's contribution guidelines,

1.1.4 Note

This project has been set up using PyScaffold 4.1.4. For details and usage information on PyScaffold see https://pyscaffold.org/.

1.2 pyscaffoldext

1.2.1 pyscaffoldext namespace

Subpackages

pyscaffoldext.cookiecutter package

Submodules

pyscaffoldext.cookiecutter.extension module

Extension that integrates cookiecutter templates into PyScaffold.

When used via PyScaffold's Python API (instead of CLI), a `cookiecutter_params` keyword argument can be optionally added to `pyscaffold.api.create_project`, this argument should be a `dict` (or a similar object that can be converted to dict via the `dict` constructor), and is equivalent to `extra_context` in `cookicookiecutter.main.cookiecutter` (PyScaffold will always add some default values, even when no `cookiecutter_params` are given).

class pyscaffoldext.cookiecutter.extension.**Cookiecutter**(*name: Optional[str] = None*)

Bases: `Extension`

Additionally apply a Cookiecutter template. Note that not all templates are suitable for PyScaffold. Please refer to the docs for more information.

activate(*actions: List[Callable[[Dict[str, Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template, Tuple[Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template], Callable[[Path, Optional[str], Dict[str, Any]], Optional[Path]]], dict]], Dict[str, Any]], Tuple[Dict[str, Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template, Tuple[Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template], Callable[[Path, Optional[str], Dict[str, Any]], Optional[Path]]], dict]], Dict[str, Any]]]*) → List[Callable[[Dict[str, Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template, Tuple[Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template], Callable[[Path, Optional[str], Dict[str, Any]], Optional[Path]]], dict]], Dict[str, Any]], Tuple[Dict[str, Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template, Tuple[Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template], Callable[[Path, Optional[str], Dict[str, Any]], Optional[Path]]], dict]], Dict[str, Any]]]]

Register before_create hooks to generate project using Cookiecutter Activate extension. See `pyscaffold.extension.Extension.activate`.

augment_cli(*parser*)

Add an option to parser that enables the Cookiecutter extension See `pyscaffold.extension.Extension.augment_cli`.

persist = False

When `True` PyScaffold will store the extension in the PyScaffold's section of `setup.cfg`. Useful for updates. Set to `False` if the extension should not be re-invoked on updates.

exception pyscaffoldext.cookiecutter.extension.**MissingTemplate**(*message='missing `cookiecutter` option', *args, **kwargs*)

Bases: `RuntimeError`

A cookiecutter template (git url) is required.

DEFAULT_MESSAGE = 'missing `cookiecutter` option'

exception pyscaffoldext.cookiecutter.extension.**NotInstalled**(*message='cookiecutter is not installed, run: pip install cookiecutter', *args, **kwargs*)

Bases: `RuntimeError`

This extension depends on the `cookiecutter` package.

DEFAULT_MESSAGE = 'cookiecutter is not installed, run: pip install cookiecutter'

pyscaffoldext.cookiecutter.extension.**create_cookiecutter**(*struct: Dict[str, Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template, Tuple[Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template], Callable[[Path, Optional[str], Dict[str, Any]], Optional[Path]]], dict]]*, *opts: Dict[str, Any]*) → Tuple[Dict[str, Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template, Tuple[Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template], Callable[[Path, Optional[str], Dict[str, Any]], Optional[Path]]], dict]], Dict[str, Any]]

Create a cookie cutter template See `pyscaffold.actions.Action`.

pyscaffoldext.cookiecutter.extension.**enforce_options**(*struct: Dict[str, Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template, Tuple[Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template], Callable[[Path, Optional[str], Dict[str, Any]], Optional[Path]]], dict]]*, *opts: Dict[str, Any]*) → Tuple[Dict[str, Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template, Tuple[Union[str, None, Callable[[Dict[str, Any]], Optional[str]], Template], Callable[[Path, Optional[str], Dict[str, Any]], Optional[Path]]], dict]], Dict[str, Any]]

Make sure options reflect the cookiecutter usage. See `pyscaffold.actions.Action`.

pyscaffoldext.cookiecutter.extension.**parameters**(*opts: Dict[str, Any]*) → Dict[str, Any]

Parameters to be passed to cookiecutter as `extra_context`

Module contents

1.3 License

The MIT License (MIT)

Copyright (c) 2019 Anderson Bravalheri

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION

OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

1.4 Contributors

- Anderson Bravalheri <andersonbravalheri@gmail.com>

1.5 Changelog

1.5.1 Version 0.1

- Initial release extracted from PyScaffold
- Updated to PyScaffold v4
- Project reformatted using Black
- Updated parameters passed down to Cookiecutter
- Added `cookiecutter_params`, so users have more control on what is being generated

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

INDEX