
pyscaffoldext-custom-extension Documentation

Release unknown

Simone Robutti

Jan 06, 2023

CONTENTS

1	Contents	3
1.1	pyscaffoldext-custom-extension	3
1.2	License	4
1.3	Contributors	5
1.4	Changelog	5
1.5	pyscaffoldext	6
2	Indices and tables	11
	Python Module Index	13
	Index	15

PyScaffold extension that lets you create your own custom extensions.

CONTENTS

1.1 pyscaffoldext-custom-extension

PyScaffold extension that lets you create your own custom extensions.

1.1.1 Description

This extension was written to help users interested in developing their own extension for PyScaffold. It configures your project so that you can start writing your extension logic and tests right away, taking care of all the wiring required to conform to PyScaffold's needs.

Let's say you want to create an extension named `notebooks` that creates a `notebooks` folder with some template [Jupyter notebook](#). After having installed this extension with:

```
pip install pyscaffoldext-custom-extension
```

you will be able to just use it with:

```
putup --custom-extension notebooks
```

This will create a typical PyScaffold project template with some modifications:

- the topmost namespace will be `pyscaffoldext` to have a unified namespace for PyScaffold extensions,
- assures that the package (as `pip/PyPI` sees it) is named `pyscaffoldext-notebooks` in `setup.cfg`,
- sets the correct `install_requires` as well as the `options.entry_points` parameters in `setup.cfg`,
- automatically activates the extensions `--no-skeleton`, `--pre-commit`, `--cirrus` and since we want clean-coded, high-quality extensions,
- creates a `extension.py` module holding a class which serves you as a template for your extension,

- adds basic unit tests checking that the invocation of your extension works and that it complies with our [flake8](#) code guidelines,
- provides a modified `README.rst` indicating that this is a PyScaffold extensions and how to install it.

1.1.2 Making Changes & Contributing

This project uses [pre-commit](#), please make sure to install it before making any changes:

```
pip install pre-commit
cd pyscaffoldext-custom-extension
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](#),

1.1.3 Note

For more information about PyScaffold and its extension mechanism, check out <https://pyscaffold.org/>.

1.2 License

The MIT License (MIT)

Copyright (c) 2018 Simone Robutti

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.

1.3 Contributors

- Simone Robutti <simone.robutti@teraki.com>
- Florian Wilhelm <florian.wilhelm@gmail.com>
- Anderson Bravalheri <andersonbravalheri@gmail.com>

1.4 Changelog

1.4.1 Version 0.6.3

- Update dependencies to include ConfigUpdater 3.x

1.4.2 Version 0.6.2

- Update configupdater to version ≥ 2.0

1.4.3 Version 0.6.1

- Ensure generated extension is tested in combination with `--namespace`, #24

1.4.4 Version 0.6

- Changes required for PyScaffold v4.0
- Updated templates for testing
- Simplified templates package thanks to `pyscaffold.templates.get_template`
- Removed unnecessary coding: `utf-8` comments
- Replaced `travis` with `cirrus` extension
- Version scheme changed to `no-guess-dev` (from `setuptools_scm`)
- Fixed references to PyScaffold in the docs, by adding it as a doc requirement, #23
- Added Github Actions template for automatically publishing tags to PyPI

1.4.5 Version 0.5

- Add `pyscaffoldext` by default warning the user

1.4.6 Version 0.4.1

- Cosmetic changes

1.4.7 Version 0.4

- Always set namespace to `pyscaffoldext`
- Changes required for PyScaffold 3.2
- Several fixes

1.4.8 Version 0.3

- Docstrings in Google Format
- Added `conftest.py` as template
- Reworked `README.rst` template
- Added default unit test for extension
- Few fixes

1.4.9 Version 0.2

- Check for `pyscaffoldext` naming convention
- Added new `README.rst`
- Check with `flake8`
- Added testing requirements
- Usage of `tox`

1.4.10 Version 0.1

- First release

1.5 pyscaffoldext

1.5.1 pyscaffoldext namespace

Subpackages

`pyscaffoldext.custom_extension` package

Subpackages

`pyscaffoldext.custom_extension.templates` package

Module contents

Submodules

pyscaffoldext.custom_extension.extension module

Main logic to create custom extensions

class pyscaffoldext.custom_extension.extension.**CustomExtension**(name: str | None = None)

Bases: `Extension`

Configures a project to start creating extensions

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

Activate extension, see `activate`.

augment_cli(parser)

Augments the command-line interface parser

A command line argument `--FLAG` where `FLAG=self.name` is added which appends `self.activate` to the list of extensions. As help text the docstring of the extension class is used. In most cases this method does not need to be overwritten.

Parameters

parser – current parser object

`pyscaffoldext.custom_extension.extension.INVALID_PROJECT_NAME = 'The prefix ``pyscaffoldext-`` will be added to the package name (as in PyPI/pip install). If that is not your intention, please use ``--force`` to overwrite.'`

Project name does not comply with convention of an extension

exception pyscaffoldext.custom_extension.extension.**NamespaceError**(message="It's not possible to define a custom namespace when using ``--custom-extension``, *args, **kwargs)

Bases: `RuntimeError`

No additional namespace is allowed

`DEFAULT_MESSAGE = "It's not possible to define a custom namespace when using ``--custom-extension``."`

```
pyscaffoldext.custom_extension.extension.add_doc_requirements(struct: Dict[str, str | None | Callable[[Dict[str, Any]], str | None] | Template | Tuple[str | None | Callable[[Dict[str, Any]], str | None] | Template, Callable[[Path, str | None, Dict[str, Any]], Path | None]] | dict], opts: Dict[str, Any]) → Tuple[Dict[str, str | None | Callable[[Dict[str, Any]], str | None] | Template | Tuple[str | None | Callable[[Dict[str, Any]], str | None] | Template, Callable[[Path, str | None, Dict[str, Any]], Path | None]] | dict], Dict[str, Any]]
```

In order to build the docs new requirements are necessary now.

The default `tox.ini` generated by PyScaffold should already include `-e {toxindir}/docs/requirements.txt` in its dependencies. Therefore, this action will make sure `tox -e docs` run without problems.

It is important to sort the requirements otherwise pre-commit will raise an error for a newly generated file and that would correspond to a bad user experience.

```
pyscaffoldext.custom_extension.extension.add_entry_point(setupcfg: ConfigUpdater, opts: Dict[str, Any]) → ConfigUpdater
```

Adds the extension's `entry_point` to `setup.cfg`

```
pyscaffoldext.custom_extension.extension.add_files(struct: Dict[str, str | None | Callable[[Dict[str, Any]], str | None] | Template | Tuple[str | None | Callable[[Dict[str, Any]], str | None] | Template, Callable[[Path, str | None, Dict[str, Any]], Path | None]] | dict], opts: Dict[str, Any]) → Tuple[Dict[str, str | None | Callable[[Dict[str, Any]], str | None] | Template | Tuple[str | None | Callable[[Dict[str, Any]], str | None] | Template, Callable[[Path, str | None, Dict[str, Any]], Path | None]] | dict], Dict[str, Any]]
```

Add custom extension files. See `pyscaffold.actions.Action`

```
pyscaffoldext.custom_extension.extension.add_pytest_requirements(setupcfg: ConfigUpdater, _opts) → ConfigUpdater
```

Add `[options.extras_require]` testing requirements for `py.test`

```
pyscaffoldext.custom_extension.extension.get_requirements() → List[str]
```

List of requirements for `install_requires`

```
pyscaffoldext.custom_extension.extension.is_commented(line)
```

```
pyscaffoldext.custom_extension.extension.modify_setupcfg(definition: str | None | Callable[[Dict[str, Any]], str | None] | Template | Tuple[str | None | Callable[[Dict[str, Any]], str | None] | Template, Callable[[Path, str | None, Dict[str, Any]], Path | None]] | dict], opts: Dict[str, Any]) → Tuple[str | None | Callable[[Dict[str, Any]], str | None] | Template, Callable[[Path, str | None, Dict[str, Any]], Path | None]]
```

Modify setup.cfg to add install_requires and pytest settings before it is written. See `pyscaffold.operations`.

```
pyscaffoldext.custom_extension.extension.process_options(struct: Dict[str, str | None |
                                                         Callable[[Dict[str, Any]], str | None] |
                                                         Template | Tuple[str | None |
                                                         Callable[[Dict[str, Any]], str | None] |
                                                         Template, Callable[[Path, str | None,
                                                         Dict[str, Any]], Path | None]] | dict], opts:
                                                         Dict[str, Any]) → Tuple[Dict[str, str |
                                                         None | Callable[[Dict[str, Any]], str |
                                                         None] | Template | Tuple[str | None |
                                                         Callable[[Dict[str, Any]], str | None] |
                                                         Template, Callable[[Path, str | None,
                                                         Dict[str, Any]], Path | None]] | dict],
                                                         Dict[str, Any]]
```

Process the given options enforcing policies and calculating derived ones.

Policies:

- Fixed namespace value of pyscaffoldext (and no extra namespace)
- The project name must start with pyscaffoldext-.
- The package name shouldn't contain the redundant pyscaffoldext_ in the beginning of the name.

See `pyscaffold.actions.Action`.

Module contents

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

p

`pyscaffoldext`, [6](#)

`pyscaffoldext.custom_extension`, [9](#)

`pyscaffoldext.custom_extension.extension`, [7](#)

`pyscaffoldext.custom_extension.templates`, [7](#)

INDEX

A

`activate()` (*pyscaffold-
ext.custom_extension.extension.CustomExtension
method*), 7

`add_doc_requirements()` (*in module pyscaffold-
ext.custom_extension.extension*), 7

`add_entry_point()` (*in module pyscaffold-
ext.custom_extension.extension*), 8

`add_files()` (*in module pyscaffold-
ext.custom_extension.extension*), 8

`add_pytest_requirements()` (*in module pyscaffold-
ext.custom_extension.extension*), 8

`augment_cli()` (*pyscaffold-
ext.custom_extension.extension.CustomExtension
method*), 7

C

`CustomExtension` (*class in pyscaffold-
ext.custom_extension.extension*), 7

D

`DEFAULT_MESSAGE` (*pyscaffold-
ext.custom_extension.extension.NamespaceError
attribute*), 7

G

`get_requirements()` (*in module pyscaffold-
ext.custom_extension.extension*), 8

I

`INVALID_PROJECT_NAME` (*in module pyscaffold-
ext.custom_extension.extension*), 7

`is_commented()` (*in module pyscaffold-
ext.custom_extension.extension*), 8

M

`modify_setupcfg()` (*in module pyscaffold-
ext.custom_extension.extension*), 8

module

`pyscaffoldext`, 6

`pyscaffoldext.custom_extension`, 9

`pyscaffoldext.custom_extension.extension`,
7

`pyscaffoldext.custom_extension.templates`,
7

N

`NamespaceError`, 7

P

`process_options()` (*in module pyscaffold-
ext.custom_extension.extension*), 9

`pyscaffoldext`

module, 6

`pyscaffoldext.custom_extension`

module, 9

`pyscaffoldext.custom_extension.extension`
module, 7

`pyscaffoldext.custom_extension.templates`
module, 7