pkg$_{u}tilsdocumentation$

Release 0.0.5

Karr Lab

Jun 26, 2020

Contents

Utilities for linking setuptools with package version metadata, GitHub README.md files, requirements.txt files, and restoring overridden entry points during for editable installations.

CHAPTER 1

Contents

1.1 Installation

1.1.1 Requirements

First, install Python and pip. The following command illustrates how to install Python and pip on Ubuntu Linux:

```
apt-get install python python-pip
```

1.1.2 Optional requirements

Second, optionally install pandoc to convert Markdown-formatted README files for GitHub into reStructuredText-formatted files for PyPI:

```
apt-get install pandoc
```

1.1.3 Installing this package

Use the following command to install this package from PyPI:

```
pip install pkg_utils
```

The latest version of this package can be installed from GitHub using this command:

```
pip install git+https://github.com/KarrLab/pkg_utils.git#egg=pkg_utils
```

Support for the pandoc can be installed using the following option:

```
pip install pkg_utils[pandoc]
pip install git+https://github.com/KarrLab/pkg_utils.git#egg=pkg_
↪utils[pandoc]
```

1.2 Tutorial

1.2.1 Linking setuptools with package version numbers

The following example shows how to link a package number stored in `package/_version.py` with setuptools:

```
import os
import setuptools
try:
    import pkg_utils
except ImportError:
    import subprocess
    import sys
    subprocess.check_call(
        [sys.executable, "-m", "pip", "install", "pkg_utils"])
    import pkg_utils

# package name
name = 'my_package'
dirname = os.path.dirname(__file__)

# get package metadata
md = pkg_utils.get_package_metadata(dirname, name)

# install package
setuptools.setup(
    ...
    version=md.version,
)
```

1.2.2 Linking setuptools with GitHub README.md files

The following example shows how to link GutHub Markdown-formatted README.md files with setuptools which requires long descriptions in reStructuredText format. Note, this feature requires the pandoc option.

```
import os
import setuptools
try:
    import pkg_utils
except ImportError:
    import subprocess
    import sys
    subprocess.check_call(
        [sys.executable, "-m", "pip", "install", "pkg_utils"])
    import pkg_utils

# package name
name = 'my_package'
dirname = os.path.dirname(__file__)

# convert README.md to README.rst
pkg_utils.convert_readme_md_to_rst(dirname)

# get package metadata
md = pkg_utils.get_package_metadata(dirname, name)
```

(continues on next page)

(continued from previous page)

```
# install package
setuptools.setup(
    ...
    long_description=md.long_description,
)
```

1.2.3 Linking setuptools with requirements

The following example illustrates how to link setuptools with requirements.txt files:

```
import os
import setuptools
try:
    import pkg_utils
except ImportError:
    import subprocess
    import sys
    subprocess.check_call(
        [sys.executable, "-m", "pip", "install", "pkg_utils"])
    import pkg_utils

# package name
name = 'my_package'
dirname = os.path.dirname(__file__)

# get package metadata
md = pkg_utils.get_package_metadata(dirname, name)

# install package
setuptools.setup(
    ...
    install_requires=md.install_requires,
    extras_require=md.extras_require,
    tests_require=md.tests_require,
    dependency_links=md.dependency_links,
)
```

This extracts dependencies from the following files:

- `requirements.txt`: dependencies
- `requirements.optional.txt`: optional dependencies
- `tests/requirement.txt`: dependencies to run the tests
- `docs/requirement.txt`: dependencies to build the documentation

The `requirements.txt` files should follow the pip format:

```
package_1
package_2[package_2_option_2] >= 1.0.0; python_version >= "2.7.14"
```

The `requirements.optional.txt` should follow the same format, but with section headings to indicate the options:

```
[my_option_1]
package_1
package_2[package_2_option_2] >= 1.0.0; python_version >= "2.7.14"

[my_option_2]
package_3
package_4
```

In addition to the installation options described in `requirements.optional.txt`, pkg_utils will create `tests`, `docs` and `all` options to install the test, documentation, and all dependencies.

1.2.4 Restoring overridden console scripts during editable installations

The following example illustrates how to restore overridden console scripts during editable installations. This useful for generating console scripts for specific versions of Python.

```
import os
import setuptools
try:
    import pkg_utils
except ImportError:
    import subprocess
    import sys
    subprocess.check_call(
        [sys.executable, "-m", "pip", "install", "pkg_utils"])
    import pkg_utils

# package name
name = 'my_package'
dirname = os.path.dirname(__file__)

# read old console scripts
console_scripts = pkg_utils.get_console_scripts(dirname, name)

# install package
setuptools.setup(...)

# restore old console scripts
pkg_utils.add_console_scripts(dirname, name, console_scripts)
```

1.2.5 Putting it all together

The following example shows how to use all of the features of this package:

```
import os
import setuptools
try:
    import pkg_utils
except ImportError:
    import subprocess
    import sys
    subprocess.check_call(
        [sys.executable, "-m", "pip", "install", "pkg_utils"])
    import pkg_utils
```

(continues on next page)

(continued from previous page)

```
# package name
name = 'my_package'
dirname = os.path.dirname(__file__)

# get package metadata
md = pkg_utils.get_package_metadata(dirname, name)

# read old console scripts
console_scripts = pkg_utils.get_console_scripts(dirname, name)

# install package
setuptools.setup(
    ...
    version=md.version,
    long_description=md.long_description,
    install_requires=md.install_requires,
    extras_require=md.extras_require,
    tests_require=md.tests_require,
    dependency_links=md.dependency_links,
)

# restore old console scripts
pkg_utils.add_console_scripts(dirname, name, console_scripts)
```

1.3 Testing

The package can be tested by running these commands:

```
pip install pytest
python -m pytest tests
```

1.4 About

1.4.1 License

The software is released under the MIT license

```
The MIT License (MIT)

Copyright (c) 2017 Karr Lab

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.
```

(continues on next page)

(continued from previous page)

```
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.
```

1.4.2 Development team

This package was developed by the Karr Lab at the Icahn School of Medicine at Mount Sinai in New York, USA.

1.4.3 Questions and comments

Please contact the Karr Lab with any questions or comments.