PyBuilder Pip Tools Documentation

Release 1.1.0

VIB/BEG/UGent, Tim Diels

Jan 31, 2017

Contents

Contents:

CHAPTER 1

User documentation

PyBuilder Pip Tools is a PyBuilder plugin which generates `*requirements*.txt` files from your project plugin/build/install dependencies and keeps your virtual env in sync with them. This is achieved with `pip-compile` and `pip-sync` from pip-tools. No distinction is made between plugin and build dependencies; plugin dependencies are treated as build dependencies.

The plugin adds the `pip_sync` task and 2 project properties: `$pybuilder_pip_tools_urls` and `$pybuilder_pip_tools_build_urls`. The properties allow specifying dependency urls for regular and build/plugin dependencies respectively:

```
project.build_depends_on('pybuilder', '>0.9.0')
project.get_property('pybuilder_pip_tools_build_urls').extend([
      'git+https://github.com/pybuilder/pybuilder.git#egg=pybuilder-0'
])
```

Urls must have a fragment containing `egg={pkg_name}-{version}`. `version` is unused and can be set to `0`, `pkg_name` must match one of the project's (build) dependencies.

The `pip_sync` task first generates these requirements files (with `pip-compile`):

`requirements.txt` Requirements file generated by `pip-compile` using `project.dependencies` (including their version constraints; as specified by `project.depends_on`) as `requirements.in` file.

`requirements_development.txt` Analog to `requirements.txt`, but override `project.dependencies` with their url in `$pybuilder_pip_tools_urls`, if any.

`build_requirements.txt` Analog to `requirements.txt`, using `project.build_dependencies` (`project.build_depends_on`) + `project.plugin_dependencies` instead.

`build_requirements_development.txt` Analog to `requirements.txt`, using `project.build_dependencies`, `project.plugin_dependencies` and `$pybuilder_pip_tools_urls` instead.

For example, for the snippet above this would generate:

- `build_requirements.txt`:

```
pybuilder==0.9.1  # some version >0.9.0
# pybuilder dependencies, also pinned
```

- `build_requirements_development.txt`:

```
  -e git+https://github.com/pybuilder/pybuilder.git#egg=pybuilder-0>0.9.0
  # pybuilder dependencies, pinned

Note: ``pip-compile`` only supports ``-e`` urls, so ``-e`` is prepended.
```

- `requirements.txt`: empty, no `project.dependencies` specified.
- `requirements_development.txt`: empty, no `project.dependencies` specified.

Finally, `pip_sync` runs:

```
pip-sync requirements_development.txt build_requirements_development.txt
```

The non-development requirements files may come in handy for syncing on a test/build server (E.g. travis, GitLab) or for deployment (E.g. making a self-contained executable).

CHAPTER 2

Developer documentation

Documentation for developers/contributors of the project.

2.1 Project decisions

python.distutils plugin generates `setup(dependency_links=...)` from `project.depends_on(url=...)`. *dependency_links* is deprecated and we do not want urls to show up in *setup.py*, so we opt to add 2 properties `*_urls` instead of using `depends_on(url=...)`.

No error is raised when plugin and build requirements conflict. No public function exists to check whether the intersection of 2 specifier sets is empty. Implementing such a function would take much effort judging by the length of the specification of version specifiers, while the issue isn't too hard for a human to figure out when pip-sync inevitably fails to find a matching version.

CHAPTER 3

Changelog

Semantic versioning is used.

3.1 1.1.0

- Enhancement: include plugin dependencies in `build_requirements*.txt` files.

3.2 v1.0.1

- Host documentation on https://readthedocs.io instead of https://pythonhosted.org.

3.3 v1.0.0

Initial release.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search