Requirements Parser

Release 0.5.0

David Fischer, Paul Horton

Apr 20, 2022

CONTENTS

Requirements parser is a Python module for parsing Pip requirement files.

Requirements parser is (now) Apache 2.0 licensed.

Quickstart:

```
import requirements

reqfile = """
Django>=1.5,<1.6
DocParser[PDF]==1.0.0
"""

for req in requirements.parse(reqfile):
    print(req.name, req.specs, req.extras)
```

Will output:

```
Django [('>=', '1.5'), ('<', '1.6')] []
DocParser [('==', '1.0.0')] ['pdf']
```

Contents:

CHAPTER

ONE

USAGE

Requirements parser works very similarly to the way pip actually parses requirement files except that pip typically proceeds to install the relevant packages.

Requirements come in a variety of forms such as requirement specifiers (such as requirements>=0.0.5), version control URIs, other URIs and local file paths.

1.1 Parsing requirement specifiers

```
import requirements
req = "django>=1.5,<1.6"
parsed = list(requirements.parse(req))[0]
parsed.name       # django
parsed.specs      # [('>=', '1.5'), ('<', '1.6')]
parsed.specifier  # True
```

1.2 Parsing version control requirements

```
req = "-e git+git://github.com/toastdriven/django-
↪haystack@259274e4127f723d76b893c87a82777f9490b960#egg=django_haystack"
parsed = list(requirements.parse(req))[0]
parsed.name      # django_haystack
parsed.vcs       # git
parsed.revision  # 259274e4127f723d76b893c87a82777f9490b960
parsed.uri       # git+git://github.com/toastdriven/django-haystack
parsed.editable  # True (because of the -e option)
```

1.3 Parsing local files

```
req = "-e path/to/project"
parsed = list(requirements.parse(req))[0]
parsed.local_file    # True
parsed.path          # path/to/project
```

CHAPTER

TWO

CHANGELOG

2.1 v0.5.0 (2022-01-20)

2.1.1 Feature

- Support all documented options in requirements files #62 (#63) (``f92c0c0`` <https://github.com/madpah/requirements-parser/commit/f92c0c079bce03b1860c78852d2c8c48cf32d539>`_)

2.2 v0.4.0 (2022-01-18)

2.2.1 Feature

- Library is now typed according to PEP561 (``0e1bb6a`` <https://github.com/madpah/requirements-parser/commit/0e1bb6a746857a59c50530155d24da487a40c4be>`_)

2.3 v0.3.1 (2021-12-17)

2.3.1 Fix

- Readthedocs config (``ac1e7fb`` <https://github.com/madpah/requirements-parser/commit/ac1e7fb616a2c15e83b8a5ca630ffd50aad4aedb>`_)

2.4 v0.3.0 (2021-12-17)

2.4.1 Feature

- Added some typing (``169ff6e`` <https://github.com/madpah/requirements-parser/commit/169ff6e79657d8091e6e1a4e21c7da794d507832>`_)
- Removed Python 2 code (``82f9473`` <https://github.com/madpah/requirements-parser/commit/82f94733f912e140fdcc0254020f4208d3e4a892>`_)

2.4.2 Fix

- Removed version from **init** (`*4e83b9d*` <https://github.com/madpah/requirements-parser/commit/4e83b9d3bdd5534da7adfdeb292ad2a1fae73ea8>`_)

Version 0.2.0 (11 Jan 2018)

This release dropped support for Python 3.2.

- Support multiple hashing algorithms at the end of the URL (#24)
- Preserve login info in requirement URI (#33)
- Support subdirectory fragments (#32)
- Support version control URLs with extras (#30)

Version 0.1.0 (2 May 2015)

- Fix a bug involving parsing projects with underscores (#17)
- Parse recursive requirements (#19)

Version 0.0.6 (16 August 2013)

- Fixed a packaging error in v0.0.5

Version 0.0.5 (16 August 2013)

- **Backwards incompatible change** to refactor the parser.
- Parser now handles VCS specific revisions and parses out data such as whether the requirement is a local file or "editable".
- Improved handling of "editable" requirements
- Improved handling of non-VCS URI requirements
- Fixes: #8, #10 and #12

CHAPTER

THREE

LOW LEVEL API

3.1 Higher Level Parsing

Typically this is called via:

```
import requirements
requirements.parse('django>=1.5')
```

requirements.parser.**parse**(*reqstr: Union[str, TextIO]*) → Iterator[*requirements.requirement.Requirement*]

Parse a requirements file into a list of Requirements

See: pip/req.py:parse_requirements()

Parameters **reqstr** – a string or file like object containing requirements

Returns a *generator* of Requirement objects

3.2 Lower Level Parsing

Under the hood, the *Requirement* class does most of the heavy lifting.

class requirements.requirement.**Requirement**(*line: str*)

Represents a single requirement

Typically instances of this class are created with `Requirement.parse`. For local file requirements, there's no verification that the file exists. This class attempts to be *dict-like*.

See: http://www.pip-installer.org/en/latest/logic.html

Members:

- `line` - the actual requirement line being parsed
- `editable` - a boolean whether this requirement is "editable"
- `local_file` - a boolean whether this requirement is a local file/path
- `specifier` - a boolean whether this requirement used a requirement specifier (eg. "django>=1.5" or "requirements")
- `vcs` - a string specifying the version control system
- `revision` - a version control system specifier
- `name` - the name of the requirement
- `uri` - the URI if this requirement was specified by URI

- `subdirectory` - the subdirectory fragment of the URI
- `path` - the local path to the requirement
- `hash_name` - the type of hashing algorithm indicated in the line
- `hash` - the hash value indicated by the requirement line
- `extras` - a list of extras for this requirement (eg. "mymodule[extra1, extra2]")
- `specs` - a list of specs for this requirement (eg. "mymodule>1.5,<1.6" => [('>', '1.5'), ('<', '1.6')])

classmethod parse(*line: str*) → *requirements.requirement.Requirement*

Parses a Requirement from a line of a requirement file.

Parameters **line** – a line of a requirement file

Returns a Requirement instance for the given line

Raises ValueError on an invalid requirement

classmethod parse_editable(*line: str*) → *requirements.requirement.Requirement*

Parses a Requirement from an "editable" requirement which is either a local project path or a VCS project URI.

See: pip/req.py:from_editable()

Parameters **line** – an "editable" requirement

Returns a Requirement instance for the given line

Raises ValueError on an invalid requirement

classmethod parse_line(*line: str*) → *requirements.requirement.Requirement*

Parses a Requirement from a non-editable requirement.

See: pip/req.py:from_line()

Parameters **line** – a "non-editable" requirement

Returns a Requirement instance for the given line

Raises ValueError on an invalid requirement

3.3 Misc Functions

requirements.**parse**(*reqstr: Union[str, TextIO]*) → Iterator[*requirements.requirement.Requirement*]

Parse a requirements file into a list of Requirements

See: pip/req.py:parse_requirements()

Parameters **reqstr** – a string or file like object containing requirements

Returns a *generator* of Requirement objects

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

r

INDEX

M

P

R