Requirements Parser

Release 0.5.0

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

Contents:

CHAPTER 1

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 2

Changelog

The *changelog* for *requirements-parser* is now published on GitHub.

See the Changelog on GitHub.

CHAPTER 3

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 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

Index

P

R