collective.recipe.pip Documentation

Release latest

March 17, 2015

Contents

Buildout recipe which parses pip config files allowing to use both pip and buildout for same project independently

CHAPTER 1

Overview

This recipe allows to parse pip configuration files (usually named requirements.txt) into just list of the eggs to use in other parts of the buildout.

The recipe mirrors the parsed eggs list into its section, so that e.g. `${pip:eggs}` will give the list of parsed eggs.

The list of eggs which come from urls (eg from github) are also exported to the urls param: `${pip:urls}` will give the list of parsed egg urls.

For now single option of the recipe is `configs` - list of config files to parse.

The config files are parsed during the initialization of the `Recipe` instance, i.e. after `buildout.cfg` is read but before any recipe is installed or updated.

CHAPTER 2

Example usage: Use an environment variable

Let's create test config files

```
>>> write('requirements.txt',
... """
... some.egg
... -e http://some.package.git.url#egg=develop.egg
... --use-wheel
... http://sourceforge.net/projects/pychecker/files/latest/download?source=files#egg=pychecker==0.8.
... fabric>=0.9b1
... # some comment
... xlrd # reading excel worksheets
... html5lib==0.95
... """)

>>> write('requirements-included.txt',
... """
... some.included.egg
... """)

>>> write('requirements-included2.txt',
... """
... some.included.egg2
... """)

>>> mkdir('file.package')
>>> write('file.package/setup.py',
... """
... from setuptools import setup
... setup(name='file.package')
... """)

>>> write('requirements2.txt',
... """
... -r requirements-included.txt
... --requirement requirements-included2.txt
... some2.egg
... django>=1.3,<1.4
... django-extensions #django extension requirements (not mandatory, but useful on dev)
... -e http://some2.package.git.url#egg=develop2.egg
... -e file.package
... --extra-index-url=http://some.index.url
... -f http://git.fabfile.org
... """)
```

We'll start by creating a buildout that uses the recipe:

```
>>> write('buildout.cfg',
... r"""
... [buildout]
... parts = pip print
...
... [some-section]
... eggs = ${pip:eggs}
...
... [pip]
... recipe = collective.recipe.pip
... configs = requirements.txt
...           requirements2.txt
... versions = versions
...
... [versions]
...
... [print]
... recipe = mr.scripty
... install =
...     ... print(self.buildout['some-section']['eggs'])
...     ... print('\n[versions]')
...     ... print('\n'.join(i + ' = ' + k for i, k in sorted(self.buildout['versions'].items())))
...     ... print('\n[urls]')
...     ... print(self.buildout['pip']['urls'])
...     ... print('# done')
...     ... return []
... """)
```

The *mr.scripty* recipe is used to print out the value of the ${some-section:some-option} option.

Running the buildout gives us:

```
>>> import sys
>>> sys.stdout.write('start\n' + system(buildout))
start...
Installing pip.
Installing print.
develop.egg
develop2.egg
django-extensions
django>=1.3,<1.4
fabric>=0.9b1
html5lib==0.95
pychecker==0.8.19
some.egg
some.included.egg
some.included.egg2
some2.egg
xlrd
[versions]
django = >=1.3,<1.4
fabric = >=0.9b1
html5lib = 0.95
pychecker = 0.8.19
zc.buildout = ...
zc.recipe.egg = ...
[urls]
/sample-buildout/file.package
```

```
git+http://some.package.git.url#egg=develop.egg
git+http://some2.package.git.url#egg=develop2.egg
http://sourceforge.net/projects/pychecker/files/latest/download?source=files#egg=pychecker==0.8.19
...
```

2.1 Contact

If you have questions, bug reports, suggestions, etc. please create an issue on the GitHub project page.

2.2 License