bootstrapper Documentation

Release 1.0.0b1

Igor Davydenko

November 16, 2015

Bootstrap Python projects or libraries by checking system pre-requirements if necessary, creating virtual environment, installing all requirements there and finally execute post-bootstrap hooks if any.

Requirements

- Python 2.6+

Installation

As easy as:

```
$ pip install bootstrapper
```

License

bootstrapper is licensed under the terms of BSD License.

Usage

```
$ python -m bootstrapper --help
usage: bootstrapper.py [-h] [--version] [-c CONFIG]
                       [-p PRE_REQUIREMENTS [PRE_REQUIREMENTS ...]] [-e ENV]
                       [-r REQUIREMENTS] [-d] [-C HOOK] [--ignore-activated]
                       [--recreate] [-q]

Bootstrap Python projects and libraries with virtualenv and pip.

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -c CONFIG, --config CONFIG
                        Path to config file. By default: bootstrap.cfg
  -p PRE_REQUIREMENTS [PRE_REQUIREMENTS ...], --pre-requirements PRE_REQUIREMENTS [PRE_REQUIREMENTS .
                        List of pre-requirements to check, separated by space.
  -e ENV, --env ENV     Virtual environment name. By default: env
  -r REQUIREMENTS, --requirements REQUIREMENTS
                        Path to requirements file. By default:
                        requirements.txt
  -d, --install-dev-requirements
                        Install prefixed or suffixed "dev" requirements after
                        installation of original requirements file or library
                        completed without errors.
  -C HOOK, --hook HOOK  Execute this hook after bootstrap process.
  --ignore-activated    Ignore pre-activated virtualenv, like on Travis CI.
  --recreate            Recreate virtualenv on every run.
  -q, --quiet           Minimize output, show only error messages.
```

Configuration

You can configure any option of `bootstrapper`, `virtualenv` and `pip` by setting it in `bootstrap.cfg` file. For example:

```
[bootstrapper]
env = venv
hook = cp -r {PROJECT}/settings_local.py{{.def,}}

[pip]
quiet = True

[virtualenv]
quiet = True
```

By default, next configuration will be used:

```
[bootstrapper]
env = env
requirements = requirements.txt
quiet = False

[pip]
download_cache = ~/.bootstrapper/pip-cache/
```

Note: Download cache option will be used only for pip 1.x as pip 6.0 introduce changes to caching and don't use this option anymore.

Your configuration or arguments from command line overwrite default options, when arguments from command line overwrite your configuration as well.

How it works?

There are two types of Python installables: libraries and projects, where library is Python code which only has `setup.py` and project is Python code that has at least `requirements.txt`, but could have `setup.py` as well.

Bootstrapper created as tool for installs Python projects, but time after time I needed to use it with libraries too, so from version 0.2 script check if passed requirements file exists on disk and if does just run,

```
$ pip install -r requirements.txt ...
```

inside of created virtual environment. But if requirements file does not exist, script sends other arguments to `pip`,

```
$ pip install -U -e . ...
```

and this is all magic.

So in pseudo-code installing Python library or project with bootstrapper is simple process of 4 steps:

```
check_pre_requirements(list)
create_virtual_environment(env)
install_library_or_project(env)
run_hook(hook)
```

Changelog

7.1 1.0.0 (2015-11-15)

- Migrate to semantic versioning
- Ability to install dev requirements after installing original requirements done without errors
- Fix support of ancient pip versions
- Provide docstrings to internal bootstrapper functions
- Discount support of `bootstrapper-X.Y` scripts
- Change preferable method of running script from `bootstrapper` to `python -m bootstrapper`
- Move documentation to Read the Docs

7.2 0.5 (2015-01-07)

- Do not use `--download-cache` option for `pip>=6.0`. More about new pip caching

7.3 0.4 (2014-08-25)

- Exit from bootstrap script if given config file doesn't exist
- Do not run post-bootstrap hook if environment creation or requirements installation ended with error

7.4 0.3.1 (2014-03-08)

- Fix `UnboundLocalError` in function to create virtual environment

7.5 0.3 (2014-03-02)

- Do not recreate virtual environment if already working in activated virtual environment
- Colorize error messages if system has pip 1.5+

- Support multiple command line arguments for pip 1.5 from config files
- Ignore double handling of virtualenv/pip errors

7.6 0.2.2 (2013-12-25)

- More fixes to MS Windows platform
- Ability to use `{pip}` in bootstrap.cfg as path to pip different in MS Windows and Unix systems
- Store full traceback on interrupting workflow or unexcepted error

7.7 0.2.1 (2013-12-20)

- Fix installing requirements in venv on MS Windows platform

7.8 0.2 (2013-12-18)

- Full support of MS Windows platform
- Ability to use bootstrapper for libraries with only `setup.py` as well as for projects with `requirements.txt` or other requirements file
- Remove support of major/minor requirements in favor of tox

7.9 0.1.6 (2013-12-17)

- Initial support of MS Windows platform

7.10 0.1.5 (2013-06-02)

- Real support of Python 3 versions
- Enable Travis CI support
- Refactor bootstrapper to Python module

7.11 0.1.4 (2013-06-02)

- Support Python 3 versions

7.12 0.1.3 (2013-05-28)

- Disable `--use-mirrors` key by default for installing requirements via `pip` cause of latest PyPI CDN changes

7.13 0.1.2 (2013-05-28)

- Make ability to reuse cached pip files by storing them in `~/.bootstrapper` user directory by default

7.14 0.1.1 (2013-01-02)

- Use `--use-mirrors` key by default when `pip` installs requirements to virtual environment

7.15 0.1 (2012-09-26)

- Initial release