runenv Documentation

Release 1.0.1

Marek Wywiał

February 03, 2017

Contents

Contents:

CHAPTER 1

runenv

Wrapper to run programs with modified environment variables loaded from given file. You can use *runenv* to manage your app settings using 12-factor principles.

You can use same environment file with **runenv** and with **docker** using env-file parameter

- Free software: BSD license
- Documentation: https://runenv.readthedocs.org.

1.1 Features

CLI:

- command-line tool to load environment variables from given file

Python API:

- load variables from a file (*.env* or passed filename)
- load only variables with given *prefix*
- *prefix* can be stripped during load
- detect whether environment was loaded by *runenv* CLI
- force load even if *runenv* CLI was used
- *search_parent* option which allows to look for *env_file* in parent dirs

1.2 Installation

In order to install use *pip*

```
$ pip install -U runenv
```

1.3 Usage

Run from shell

```
$ runenv env.development ./manage.py runserver
```

example *env.development* file

```
BASE_URL=http://127.0.0.1:8000
DATABASE_URI=postgres://postgres:password@localhost/dbname
SECRET_KEY=y7W8pbRcuPuAmgTHsJtEpKocb7XPcV0u

# email settings
EMAIL_HOST=smtp.mandrillapp.com
EMAIL_PORT=587
EMAIL_HOST_USER=someuser
EMAIL_HOST_PASSWORD=hardpassword
EMAIL_FROM=dev@local.host
EMAIL_USE_TLS=1
```

1.4 Python API

load_env(env_file='.env', prefix=None, strip_prefix=True, force=False, search_parent=0)

Loads environment from given `env_file` (default *.env*).

Options:

option	default	description
env_file	*.env*	relative or absolute path to file with environment variables
prefix	*None*	prefix to match variables e.g. *APP_*
strip_prefix	*True*	should the prefix be stripped during loa
force	*False*	load env_file, even though *runenv* CLI command was used
search_parent	*0*	To what level traverse parents in search of file

If `prefix` option is provided only variables starting with it will be loaded to environment, with their keys stripped of that prefix. To preserve prefix, you can set `strip_prefix` to `False`.

Example

```
$ echo 'APP_SECRET_KEY=bzemAG0xfdMgFrHBT3tJBbiYIoY6EeAj' > .env
```

```
$ python
>>> import os
>>> from runenv import load_env
>>> load_env(prefix='APP_')
>>> 'APP_SECRET_KEY' in os.environ
False
>>> 'SECRET_KEY' in os.environ
True
>>> load_env(prefix='APP_', strip_prefix=False)
>>> 'APP_SECRET_KEY' in os.environ
True
```

Notice: Environment will not be loaded if command was fired by *runenv* wrapper, unless you set the **force** parameter to **True**

`load_env` does not load variables when wrapper `runenv` is used. Also `_RUNENV_WRAPPED` is set to `1`

Example

```
$ echo 'APP_SECRET_KEY=bzemAG0xfdMgFrHBT3tJBbiYIoY6EeAj' > .env
```

```
$ python
>>> import os
>>> from runenv import load_env
>>> os.environ['_RUNENV_WRAPPED'] = '1'
>>> load_env()
>>> 'APP_SECRET_KEY' in os.environ
False
>>> load_env(force=True)
>>> 'APP_SECRET_KEY' in os.environ
True
```

1.4.1 Django/Flask integration

To use `load_env` with Django or Flask, put the followin in `manage.py` and `wsgi.py`

```
from runenv import load_env
load_env()
```

1.4.2 Similar projects

- https://github.com/jezdez/envdir - runs another program with a modified environment according to files in a specified directory
- https://github.com/theskumar/python-dotenv - Reads the key,value pair from .env and adds them to environment variable

CHAPTER 2

Installation

At the command line:

```
$ easy_install runenv
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv runenv
$ pip install runenv
```

CHAPTER 3

Usage

Run from shell:

```
$ runenv env.development ./manage.py runserver
```

example *env.development* file:

```
BASE_URL=http://127.0.0.1:8000
DATABASE_URI=postgres://postgres:password@localhost/dbname
SECRET_KEY=y7W8pbRcuPuAmgTHsJtEpKocb7XPcV0u

# email settings
EMAIL_HOST=smtp.mandrillapp.com
EMAIL_PORT=587
EMAIL_HOST_USER=someuser
EMAIL_HOST_PASSWORD=hardpassword
EMAIL_FROM=dev@local.host
EMAIL_USE_TLS=1
```

CHAPTER 4

Python API

load_env(env_file='.env', prefix=None, strip_prefix=True, force=False)

Loads environment from given `env_file`', default *.env*.

If `prefix` provided only variables started with given prefix will be loaded to environment with keys truncated from `prefix`. To preserver prefix, pass `strip_prefix=False`.

Example:

```
$ echo 'DJANGO_SECRET_KEY=bzemAG0xfdMgFrHBT3tJBbiYIoY6EeAj' > .env

$ python
>>> import os
>>> from runenv import load_env
>>> load_env(prefix='DJANGO_')
>>> 'DJANGO_SECRET_KEY' in os.environ
False
>>> 'SECRET_KEY' in os.environ
True
>>> load_env(prefix='DJANGO_', strip_prefix=False)
>>> 'DJANGO_SECRET_KEY' in os.environ
True
```

Notice: Environment will be not loaded if command was fired by *runenv* wrapper unless you use **force=True** parameter

Wrapper `runenv` sets `_RUNENV_WRAPPED=1` variable and `load_env` does not load variables then.

Example:

```
$ echo 'DJANGO_SECRET_KEY=bzemAG0xfdMgFrHBT3tJBbiYIoY6EeAj' > .env

$ python
>>> import os
>>> from runenv import load_env
>>> os.environ['_RUNENV_WRAPPED'] = '1'
>>> load_env()
>>> 'DJANGO_SECRET_KEY' in os.environ
False
>>> load_env(force=True)
>>> 'DJANGO_SECRET_KEY' in os.environ
True
```

4.1 Django integration

To use `load_env` with Django, put in `manage.py` and `wsgi.py` code:

```
from runenv import load_env
load_env()
```

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/onjin/runenv/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

5.1.4 Write Documentation

runenv could always use more documentation, whether as part of the official runenv docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/onjin/runenv/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *runenv* for local development.

1. Fork the *runenv* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/runenv.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv runenv
$ cd runenv/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 runenv tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/onjin/runenv/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_runenv
```

CHAPTER 6

Credits

6.1 Development Lead

- Marek Wywiał <onjinx@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

CHAPTER 8

1.0.1 (2017-02-03)

- fix package description syntax

CHAPTER 9

1.0.0 (2017-02-03)

- changed version to *1.0.0*

CHAPTER 10

0.4.0 (2016-08-08)

- add support for *search_parent* option

0.3.1 (2016-06-21)

- add support for quoted values

CHAPTER 12

0.3.0 (2016-02-14)

- change *Development Status* to *5 - Production/Stable*

0.2.5 (2015-11-30)

- do not look for executable as absolute path

0.2.4 (2015-07-06)

- skip *load_env* if env file does not exists

0.2.3 (2015-06-26)

- support to run programs from PATH

CHAPTER 16

0.2.2 (2015-06-16)

- fix compatibility with python3

0.2.1 (2015-06-16)

- add *strip_prefix* option to *load_env*

CHAPTER 18

0.2.0 (2015-06-16)

- add *load_env* (python api)

CHAPTER 19

0.1.4 (2015-06-15)

- Check if file to run exists and is executable

CHAPTER 20

0.1.3 (2015-06-01)

- Support for env file comments by '#'

0.1.2 (2015-06-01)

- Return code from runned command

CHAPTER 22

0.1.1 (2015-05-31)

- First release on PyPI.

CHAPTER 23

Indices and tables

- genindex
- modindex
- search