frkl Documentation

Release 0.3.0

Markus Binsteiner

Jun 20, 2018

Contents:

1 Overview **1**
1.1 Features 2

2 Installation **3**
2.1 Stable release 3
2.2 From sources 3

3 Usage **5**

4 Contributing **7**
4.1 Types of Contributions 7
4.2 Get Started! 8
4.3 Pull Request Guidelines 9
4.4 Tips 9

5 Credits **11**
5.1 Development Lead 11
5.2 Contributors 11

6 History **13**
6.1 0.3.0 (2018-05-24) 13
6.2 0.1.0 (2017-05-05) 13

7 Indices and tables **15**

CHAPTER 1

Overview

frkl is basically a string/object transformation library, with the main goal being enabling as minimal as possible initial state. I rather suspect that this here is one of those things whose value won't be immediately obvious (if there is one at all – still not sure about that myself), and examples might be a better way of explaining what its use is.

frkl is most useful for cases where you have a list of similar configuration items, which might or might not inherit from each other. In those cases, you don't want to duplicate information that is needed for each of the items. To illustrate, here's some yaml:

```
vars:
  location: at_home
  task_type: cleaning
tasks:
  - clean_bathroom
  - clean_living_room
  - clean_desk:
      location: at_work
```

This task list describes how we want to clean three things, two of which are at home, and one is at work. Our robot would not like this way of describing it though, since it is much harder to parse. For example, there is no 'proper' schema, the list for example has mixed types, strings and a dictionary with one key/value pair. What our robot would want is:

```
- task:
    name: clean_bathroom
  vars:
    location: at_home
    task_type: cleaning
- task:
    name: clean_living_room
  vars:
    location: at_home
    task_type: cleaning
```

(continues on next page)

(continued from previous page)

```
- task:
    name: clean_desk
  vars:
    location: at_work
    task_type: cleaning
```

Basically, this is what *frkl* does: expanding (and also modifying if wanted) configuration from as minimal as possible to as comprehensive as necessary.

Now, of course, in this example the reduction in size is not that big. And, one might argue, not having a fixed schema might not be a good idea in the first place. I can even see the point, but I do like being able to express myself as simple and minimal as possible. Obviously we are introducing more fragility by loosening up our schema. But we gain clarity, and ease of use. Whether this trade-off is justifiable or not depends on the situation I think. This library is for the situations where it is :-)

Also, just so you know, *frkl* has a few more tricks up its sleeve. For those, check out the yet to be written configuration at this yet to be created link

frkl is written in Python, and GPL v3 licensed (for now).

- Documentation: https://frkl.readthedocs.io.

1.1 Features

- transform configurations, focusing on clarity and the removal of redundancy
- plug-able architecture
- pre-made string/object processors/filters (regex, url abbreviation, jinja templates, etc.)
- auto-downloading, merging of configuration items
- mix and match of local and remote configuration items

CHAPTER 2

Installation

2.1 Stable release

To install frkl, run this command in your terminal:

```
$ pip install frkl
```

This is the preferred method to install frkl, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for frkl can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/makkus/frkl
```

Or download the tarball:

```
$ curl  -OL https://github.com/makkus/frkl/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use frkl in a project:

```
import frkl
```

CHAPTER 4

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/makkus/frkl/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.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/makkus/frkl/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 :)

4.2 Get Started!

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

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

   ```
   $ git clone git@github.com:your_name_here/frkl.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 frkl
   $ cd frkl/
   $ 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 frkl tests
   $ python setup.py test or 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.

4.3 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/makkus/frkl/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

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

CHAPTER 5

Credits

5.1 Development Lead

- Markus Binsteiner <makkus@posteo.de>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.3.0 (2018-05-24)

- Major refactoring

6.2 0.1.0 (2017-05-05)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search