frappuccino

May 26, 2020

Contents:

Note: This project is still in experimental spate, any feebback is welcome.

CHAPTER 1

Frappuccino

Freeze your API.

Frappuccino is a tool to help you determine what the changes of API in your projects are and catch potential breaking changes. The goal is to not only provide a continuous integration feature to fail in case of inadvertently breaking API, but also to provide an easy way to list all those changes in release notes.

To do so Frappuccino takes a snapshot of the API of your project at one point in time and compare it with the API on the master version, and list the differences.

Example:

```
# old function
def read(name, *, options=None):
    with open(name, 'rb') as f:
        return process(data)

# new function
def read(name_or_buffer, *, options=None):
    if isinstance(name, str):
        with open(name, 'rb') as f:
            data = f.read()
    else:
        data = name_or_buffer.read()
    return process(data)
```

There is a subtle breakage of API in the above, as you may not remember positional parameters can be use a keyword arguments. That is to say one of your users may be calling the function like so:

```
read(name='dump.csv')
```

Hence changing the _name_ of the positional parameter from `name` to `name_or_buffer` is a change of API. There are a number of details like this one where you _may_ end up breaking API without realizing. It's hard to keep track of this when working on dev branches, unit test may not catch all of that. Frappuccino is there to help.

CHAPTER 2

Install

Frappuccino is only available via PyPI, as a Python 3 wheel:

```
pip install frappuccino
```

From source, using flit:

```
pip install flit
git clone https://github.com/carreau/frappuccino
cd frappuccino
flit install
```

For a developer install:

```
flit install --symlink
```

CHAPTER 3

How to use

Using frappuccino is pretty straitforward:

- make sure the previous version of your project library is importable.
- run `frappuccino <import name> --save somename.json`
- make sure the new version of your project library is importable, and run with the `--compare` flag `frappuccino <import name> --compare somename.json`

For example:

```
$ source activate astropy==3.2
$ frappuccino astropy astropy.timeseries --save    astropy.json

$ source activate astropy=master
$ frappuccino astropy astropy.timeseries --compare astropy.json

The following signatures differ between versions:

    - astropy.time.core.TimeDelta.to(self, *args, **kwargs)
    + astropy.time.core.TimeDelta.to(self, unit, equivalencies='[]')

    - astropy.table.table.Table.add_column(self, col, index='None', name='None',␣
↪rename_duplicate='False', copy='True')
    + astropy.table.table.Table.add_column(self, col, index='None', name='None',␣
↪rename_duplicate='False', copy='True', default_name='None')

    - astropy.table.table.Table.replace_column(self, name, col)
    + astropy.table.table.Table.replace_column(self, name, col, copy='True')
```

Another example to compare two files:

```
cp frappuccino/tests/old.py
frappuccino/t.py
frappuccino frappuccino.t --save t.json;
sleep 2;  # avoid python bytecode caching.
```

(continues on next page)

(continued from previous page)

```
cp frappuccino/tests/new.py frappuccino/t.py;
frappuccino frappuccino.t --compare t.json
```

CHAPTER 4

FAQ

Q: Should I commit the JSON files to my repositories to not have to rerun on old versions of my projects ?

A: I advise not to, the on-disk file format is not stable yet newer version of Frappuccino might not understand it.

Q: Why a CLI and not as python function/test plug-in

A: You usually need to run Frappuccino twice on two versions of the same library , so usually you can't do that in process.

Q: Results are wrong.

A: Open an issue please.

Q: It crashes.

A: See "Results are wrong".

Q: How can I use it in CI

A: Good question, I'm still thinking about it, in most CI you _should_ have access to your VCS history, so checkout an old version and build it ? You can also upload the .json results to a non URL and grab it. The return code is not yet non-zero anyway as I haven't implemented all of the comparisons function yet.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search