BFET Documentation

Release 0.1.8

Lucas Montes

Dec 22, 2022

CONTENTS:

CHAPTER

ONE

BFET

Better Faster Easier Testing. Create Django models quickly and easly, create different data types for testing cases or create default tests files

- Free software: MIT license
- Documentation: https://bfet.readthedocs.io.

1.1 User installation

The easiest way to install bfet is using `pip`:

```
pip install -U bfet
```

1.2 Source code

You can check the latest sources with the command:

```
git clone https://github.com/lluc2397/BFET.git
```

1.3 Testing

After installation, you can launch the test suite from outside the source directory (you will need to have `pytest` installed):

```
pytest bfet
```

1.4 Contributing

Welcome! Happy to see you willing to make the project better. You can get started by reading this:

- [Contributing: The basics](https://github.com/lluc2397/BFET/blob/main/CONTRIBUTING.rst)

1.5 Change log

The log has become rather long. It moved to its own file.

See [CHANGES](https://github.com/lluc2397/BFET/blob/main/HISTORY.rst).

1.6 Authors

The author list is quite long nowadays, so it lives in its own file.

See [AUTHORS.rst](./AUTHORS.rst)

1.6.1 Features

- **TODO**
  - Improve tests
  - Add support for pytest
  - Create templates for admin.py and urls.py for Django
  - Separate files creations

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install BFET, run this command in your terminal:

```
$ pip install bfet
```

This is the preferred method to install BFET, 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 BFET can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/lluc2397/bfet
```

Or download the tarball:

```
$ curl -OJL https://github.com/lluc2397/bfet/tarball/master
```

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

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

CHAPTER

THREE

USING BFET WITH DJANGO

Two of the modules more usefull for Django are DjangoTestingModel and DataCreator.

CHAPTER

FOUR

DATACREATOR

4.1 `DataCreator` - Use different modules included with python like datetime to create values

1. Import from the bfet library DataCreator:

```
from bfet import DataCreator
```

2. Once imported you can use the different methods of DataCreator to cerate values:

```
string = DataCreator.create_random_string()
dict = DataCreator.create_random_json()
email_string = DataCreator.create_random_email()
datetime = DataCreator.create_random_datetime()
```

CHAPTER

FIVE

DJANGOTESTINGMODEL

5.1 `DjangoTestingModel` - Is based on the Django _default_manager's methods create, bulk_create and get_or_create

To use this with your project you need to follow these steps:

1. Import from the bfet library DjangoTestingModel:

```
from bfet import DjangoTestingModel
```

2. Once imported you can start to create models for testing:

```
from foo.models import BarModel


bar_model = DjangoTestingModel.create(BarModel)
```

You can either include manually the fields with the values or let bfet create the values randomly. Under the hood DjangoTestingModel uses DataCreator to populate the values for the fields.

CHAPTER

SIX

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:

6.1 Types of Contributions

6.1.1 Report Bugs

Report bugs at https://github.com/lluc2397/bfet/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.

6.1.2 Fix Bugs

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

6.1.3 Implement Features

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

6.1.4 Write Documentation

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

6.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/lluc2397/bfet/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 :)

6.2 Get Started!

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

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

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

6.3 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 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/lluc2397/bfet/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Tips

To run a subset of tests:

```
$ pytest tests.test_bfet
```

6.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SEVEN

CREDITS

7.1 Development Lead

- Lucas Montes <lluc23@hotmail.com>

7.2 Contributors

None yet. Why not be the first?

CHAPTER

EIGHT

HISTORY

8.1 0.0.0 (2022-09-13)

- First release on PyPI.

8.2 0.1.0 (2022-09-14)

- First working and styled release.

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search