Formal Documentation

Release 0.6.5

Formal Contributors

Sep 01, 2019

Contents

1 About 3

2 User's Manual 5

3 Developer Documentation 7

4 Indices and tables 15

Version 0.6

Release 0.6.5

Date Sep 01, 2019

CHAPTER 1

About

1.1 About: Formal

CHAPTER 2

User's Manual

2.1 Formal User's Manual

Welcome to the Formal Users Manual! This part of the documentation explains how to work with Formal and use the core modules.

2.1.1 Detailed Feature Overview

CHAPTER 3

Developer Documentation

3.1 How to help the project?

Glad to see you're interested in helping out the project!

Generally, you can ping riot if you want to help out and don't exactly know where to start.

Here, we list a few possible opportunities where you can help us and become part of the driving community:

3.1.1 Communication

People need to be more aware of this project as it may be of great value to them. If you're interested in spreading the word and getting people involved, you're very welcome to do so. Again, please ping riot to get crucial info on how to do so.

3.1.2 Testing

There are various degrees to which you can test the project:

- Check the installation processes if they actually work on your platform and everything installs smoothly
- Test-drive your installation
- Build and extend parts of the automatic testing infrastructure
- Optimize and extend the continuous integration infrastructure

3.1.3 Documentation

A lot of documentation is still missing. If you're interested in writing documentation, you should be familiar with the two core tools we use for generating our documentation:

- reStructured Text formatting
- Sphinx

We still need a lot of module, core framework and source code documentation, so there's ample opportunities in this field.

3.1.4 Translations

Most of (if not all) parts of the project can be translated and are waiting for your help. You can use Transifex to translate all the strings we have or work with your favourite PO Editor. Have a look at *Translating Formal* for more details.

3.2 Developer Guidelines

This is the rather dry material for new software developers:

3.2.1 Development Introduction

Here's how we do things in Formal...

If you're looking for instructions on how to set up a development environment, please check out *the workflow documentation*.

Communication

- **'#hackerfleet IRC Channel'_** on the FreeNode IRC Network
- Issue Tracker located at *https://github.com/isomeric/formal/issues*

Note: If you are familiar with IRC and use your own IRC Client then connect to the FreeNode Network and `/join #hackerfleet`.

Standards

We use the following coding standards:

- PEP-008

We also lint our codebase with the following tools:

- pyflakes
- pep8
- mccabe

Please ensure your Development IDE or Editor has the above linters and checkers in place and enabled.

Alternatively you can use the following command line tool:

- flake8

Tools

We use the following tools to develop Formal and share code:

- **Code Sharing:** Git
- **Code Hosting and Bug Reporting:** GitHub
- **Issue Tracker:** Issue Tracker
- **Documentation Hosting:** Read the Docs
- **Package Hosting:** Python Package Index (PyPi)

- **Continuous Integration:** Travis CI
- **Code Quality:** Landscape
- **Translations:** Transifex

We strongly suggest familiarizing with all of them, to make sure you understand our CI.

Big thanks to all of these magnificent and free-for-opensource services!

3.2.2 Contributing to Formal

Here's how you can contribute to Formal

Submitting Bug Reports

We welcome all bug reports. We do however prefer bug reports in a clear and concise form with repeatable steps. One of the best ways you can report a bug to us is by writing a unit test (//similar to the ones in our tests//) so that we can verify the bug, fix it and commit the fix along with the test.

To submit a bug report, please Create an Issue

Writing new tests

We're not perfect, and we're still writing more tests to ensure quality code. If you'd like to help, please Fork Formal, write more tests that cover more of our code base and submit a Pull Request. Many Thanks!

Adding New Features

If you'd like to see a new feature added to Formal, then we'd like to hear about it~ We would like to see some discussion around any new features as well as valid use-cases. To start the discussions off, please either:

- Chat with us on #hackerfleet on the FreeNode IRC Network

  or

- Create an Issue

3.2.3 Setting up a Formal Development Environment

This is the recommended way to setup a development environment for developing Formal.

Getting Started

Here is a summary of the steps to your own development environment:

1. Fork Formal (*if you haven't done so already*)
2. Clone your forked repository using Git
3. Create a virtual environment
4. Install formal
5. Run tests

And you're done!

Setup

The setup guide shall aid you in setting up a development environment for all purposes and facettes of Formal development. It is split up in a few parts and a common basic installation.

Get the sourcecode

After forking the repository, clone it to your local machine:

```
git clone git@github.com:yourgithubaccount/formal.git ~/src/formal
```

Setting up a basic development Instance

First install the management tool:

```
cd ~/src/formal
python setup.py install
```

This installs basic dependencies and Formal itself.

3.2.4 Development Processes

We document all our internal development processes here so you know exactly how we work and what to expect. If you find any issues or problems, please let us know!

Software Development Life Cycle (SDLC)

We employ the use of the SCRUM Agile Process and use our Issue Tracker to track features, bugs, chores and releases. If you wish to contribute to Formal, please familiarize yourself with SCRUM and GitHub's Issue Tracker.

Bug Reports

- New Bug Reports are submitted via: https://github.com/isomeric/formal/issues
- Confirmation and Discussion of all New Bug Reports.
- Once confirmed, a new Bug is raised in our Issue Tracker
- An appropriate milestone will be set (*depending on current milestone's schedule and resources*)
- A unit test developed that demonstrates the bug's failure.
- A fix developed that passes the unit test and breaks no others.
- A New Pull Request created with the fix.

  This should contain: - A new or modified unit test. - A patch that fixes the bug ensuring all unit tests pass. - The Change Log updated. - Appropriate documentation updated.
- The Pull Request is reviewed and approved by at least two other developers.

Feature Requests

- New Feature Requests are submitted via: https://github.com/isomeric/formal/issues
- Confirmation and Discussion of all New Feature Requests.
- Once confirmed, a new Feature is raised in our Issue Tracker

- An appropriate milestone will be set (*depending on current milestone's schedule and resources*)
- A unit test developed that demonstrates the new feature.
- The new feature developed that passes the unit test and breaks no others.
- A New Pull Request created with the fix.

  This must contains: - A new or modified unit test. - A patch that implements the new feature ensuring all unit tests pass. - The Change Log updated. - Appropriate documentation updated.
- The Pull Request is reviewed and approved by at least two other developers.

Writing new Code

- Submit a New Issue
- Write your code.
- Use flake8 to ensure code quality.
- Run the tests:

```
tox
```

- Ensure any new or modified code does not break existing unit tests.
- Update any relevant doc strings or documentation.
- Update the Change Log appropriately.
- Submit a New Pull Request.

Running the Tests

To run the tests you will need the following installed:

- tox installed as well as
- pytest-cov
- pytest

All of these can be installed via `pip install -r requirements-dev.txt`.

Please also ensure - if you can - that you you have all supported versions of Python that Formal supports installed in your local environment.

To run the tests:

```
tox
```

3.2.5 Development Standards

We aim for the following development standards:

Cyclomatic Complexity

- Code Complexity shall not exceed `10`

  See: Limiting Cyclomatic Complexity

Coding Style

Note: We do accept "black" formatting.

- Code shall confirm to the PEP8 Style Guide.
- Doc Strings shall confirm to the PEP257 Convention.

Note: Arguments, Keyword Arguments, Return and Exceptions must be documented with the appropriate Sphinx Python Domain.

Revision History

- Commits shall be small tangible pieces of work. - Each commit must be concise and manageable. - Large changes are to be done over smaller commits.
- There shall be no commit squashing.
- Rebase your changes as often as you can.

Unit Tests

- Every new feature and bug fix must be accompanied with a unit test. (*The only exception to this are minor trivial changes*).

3.2.6 Translating Formal

To translate Formal, you can use Transifex or any PO editor of your choice.

3.3 Recent Changes

- **Updated** by *ri0t* at *2019-09-01 20:37:04*
- **Construct short version from tuple** by *ri0t* at *2019-09-01 20:22:17*
- **Fixed license and version_info string** by *ri0t* at *2019-09-01 20:22:02*
- **Removed future interpreter** by *ri0t* at *2019-09-01 20:21:43*
- **Removed 3.3 interpreter** by *ri0t* at *2019-09-01 19:31:35*
- **Reformatted** by *ri0t* at *2019-09-01 19:25:45*
- **Minor cleanups and reformatting** by *ri0t* at *2019-09-01 19:25:40*
- **Added sphinx documentation** by *ri0t* at *2019-09-01 19:25:35*
- **Added tox configuration** by *ri0t* at *2019-09-01 19:25:15*
- **Updated travis for newer interpreters and pytest infrastructure** by *ri0t* at *2019-09-01 19:24:46*

3.4 Road Map

We manage our roadmap via milestones on our github issuetracker.

3.5 Contributors

The following users and developers have contributed to Formal:

- Rob Britton (Original author)
- Heiko 'riot' Weinen riot@c-base.org (Current maintainer)
- You?

Anyone not listed here, ping us. We appreciate any and all contributions to Formal and other Hackerfleet components.

3.6 Supporters

- **Free OSS license of IntelliJ IDEA Ultimate:** Jetbrains
- **Repository and issue tracker hosting:** Github
- **Repository and issue tracker hosting:** GitLab

3.7 Glossary

CHAPTER 4

Indices and tables

- Index
- modindex
- search
- *Glossary*
- *Recent Changes*