pyfb-direction Documentation

Release 1.0.1

Mathias WOLFF

Mar 27, 2019

Contents

Contents:

CHAPTER 1

Pyfb-direction

Phonenumber's direction package

1.1 Documentation

The full documentation is at https://pyfb-direction.readthedocs.io.

1.2 Quickstart

Install pyfb-direction:

```
pip install pyfb-direction
```

Add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'pyfb_direction.apps.PyfbDirectionConfig',
    ...
)
```

Add pyfb-direction's URL patterns:

```
from pyfb_direction import urls as pyfb_direction_urls
```

(continues on next page)

(continued from previous page)

```
urlpatterns = [
    ...
    url(r'^', include(pyfb_direction_urls)),
    ...
]
```

1.3 Features

- Telephony prefix management linking prefix with destination, country, network type and Carrier
- Admin interface
- CSV import / export
- web template with Bootstrap 4
- APIs

1.4 Running Tests

Does the code actually work?

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install tox
(myenv) $ tox
```

1.5 Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage

CHAPTER 2

Installation

At the command line:

```
$ python3 -m venv pyfb-direction
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv pyfb-direction
$ pip install pyfb-direction
```

CHAPTER 3

Usage

To use pyfb-direction in a project, add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'pyfb_direction.apps.PyfbDirectionConfig',
    ...
)
```

Add pyfb-direction's URL patterns:

```
from pyfb_direction import urls as pyfb_direction_urls


urlpatterns = [
    ...
    url(r'^', include(pyfb_direction_urls)),
    ...
]
```

CHAPTER 4

Internationalization

All user-facing text content should be marked for translation. Even if this application is only run in English, our open source users may choose to use another language. Marking content for translation ensures our users have this choice.

Follow the internationalization coding guidelines in the edX Developer's Guide when developing new features.

4.1 Updating Translations

This project uses Transifex to translate content. After new features are developed the translation source files should be pushed to Transifex. Our translation community will translate the content, after which we can retrieve the translations.

Pushing source translation files to Transifex requires access to the edx-platform. Request access from the Open Source Team if you will be pushing translation files. You should also configure the Transifex client if you have not done so already.

The *make* targets listed below can be used to push or pull translations.

Target	Description
pull_translations	Pull translations from Transifex
push_translations	Push source translation files to Transifex

4.2 Fake Translations

As you develop features it may be helpful to know which strings have been marked for translation, and which are not. Use the *fake_translations* make target for this purpose. This target will extract all strings marked for translation, generate fake translations in the Esperanto (eo) language directory, and compile the translations.

You can trigger the display of the translations by setting your browser's language to Esperanto (eo), and navigating to a page on the site. Instead of plain English strings, you should see specially-accented English strings that look like this:

> Thé Fütüré øf Ønlïné Édüçätïøn σ ι# Før änýøné, änýwhéré, änýtïmé σ #

CHAPTER 5

Testing

{{ cookiecutter.project_name }} has an assortment of test cases and code quality checks to catch potential problems during development. To run them all in the version of Python you chose for your virtualenv:

```
$ make validate
```

To run just the unit tests:

```
$ make test
```

To run just the unit tests and check diff coverage

```
$ make diff_cover
```

To run just the code quality checks:

```
$ make quality
```

To run the unit tests under every supported Python version and the code quality checks:

```
$ make test-all
```

To generate and open an HTML report of how much of the code is covered by test cases:

```
$ make coverage
```

CHAPTER 6

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/mwolff44/pyfb-direction/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” is open to whoever wants to implement it.

6.1.3 Implement Features

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

6.1.4 Write Documentation

pyfb-direction could always use more documentation, whether as part of the official pyfb-direction 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/mwolff44/pyfb-direction/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 *pyfb-direction* for local development.

1. Fork the *pyfb-direction* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/pyfb-direction.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 pyfb-direction
$ cd pyfb-direction/
$ 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 pyfb_direction 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.

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 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/mwolff44/pyfb-direction/pull_requests and make sure that the tests pass for all supported Python versions.

6.4 Tips

To run a subset of tests:

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

CHAPTER 7

Credits

7.1 Development Lead

- Mathias WOLFF <mathias@celea.org>

7.2 Contributors

None yet. Why not be the first?

CHAPTER 8

History

8.1 1.0.1 (2019-01-30)

- SQL view for Kamailio

8.2 1.0.0 (2018-11-15)

- First release on PyPI.