django-teryt Documentation

Release 0.2.0

Patryk Ściborek

August 02, 2015

Contents

Contents:

CHAPTER 1

django-teryt

django-teryt is a Django app that implements TERYT database. TERYT (Polish: "Krajowy Rejestr Urzędowy Podziału Terytorialnego Kraju", English: "National Official Register of Territorial Division of the Country") is a register maintained by Polish Central Statistical Office (Polish: Główny Urząd Statystyczny; GUS). Among other things it contains:

- identifiers and names of units of territorial division,
- identifiers and names of localities,
- identifiers and names of streets

This app parses XML files from GUS and it imports them to the database. It is meant to be used as a part of a larger system.

1.1 Documentation

The full documentation is at http://django-teryt.rtfd.org.

1.2 Quickstart

Install django-teryt:

```
pip install django-teryt
```

If you are using Django 1.6 or lower you have to install South:

```
pip install 'south>=1.0'
```

Add `teryt` to `INSTALLED_APPS` in your `settings.py` and run:

```
./manage.py migrate teryt
```

Then download TERYT data from GUS website, unpack it and then import it:

```
./manage.py teryt_parse /path/to/WMRODZ.xml /path/to/TERC.xml /path/to/SIMC.xml /path/to/ULIC.xml
```

1.3 Features

- It can import all data from all TERYT files
- It deals with updates (just run ./manage.py teryt_parse –update TERC.xml)
- It keeps flag (aktywny) telling you if some record is still present in TERYT (there are some minor changes in territorial division from time to time)

1.4 Support

All bug reports and pull requests are welcome. You can report them at https://github.com/scibi/django-teryt/issues. It can be in English or in Polish ;)

CHAPTER 2

Installation

At the command line:

```
$ easy_install django-teryt
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-teryt
$ pip install django-teryt
```

CHAPTER 3

Usage

To use django-teryt in a project:

```
import django-teryt
```

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/scibi/django-teryt/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" is open to whoever wants to implement it.

4.1.3 Implement Features

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

4.1.4 Write Documentation

django-teryt could always use more documentation, whether as part of the official django-teryt 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/scibi/django-teryt/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 *django-teryt* for local development.

1. Fork the *django-teryt* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-teryt.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 django-teryt
$ cd django-teryt/
$ 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 teryt 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.

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, and 3.3, and for PyPy. Check https://travis-ci.org/scibi/django-teryt/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_teryt
```

CHAPTER 5

Credits

5.1 Development Lead

- Patryk Ściborek <patryk@sciborek.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2013-12-31)

- First release on PyPI.

6.2 0.2.0 (2015-01-08)

- Added support for Django 1.7 migrations
- Moved common data to abstract base class
- Added common flag filed (aktywny)
- Added 3 model managers to JednostkaAdministracyjna
- Fixed PEP 8 compliance in main source files