django-geopostcodes Documentation

Release 0.3.1

Alex Hayes

Oct 06, 2022

Contents:

1 django-geopostcodes **1**
1.1 Credits . . . 1

2 Installation **3**
2.1 Stable release . . . 3
2.2 From sources . . . 3
2.3 Django Settings . . . 3
2.4 Data Import . . . 4

3 Usage **5**
3.1 Locality . . . 5

4 Contributing **7**
4.1 Types of Contributions . . . 7
4.2 Get Started! . . . 8
4.3 Pull Request Guidelines . . . 9
4.4 Tips . . . 9
4.5 Deploying . . . 9

5 Credits **11**
5.1 Development Lead . . . 11
5.2 Contributors . . . 11

6 History **13**
6.1 0.3.1 (2022-10-06) . . . 13
6.2 0.3.0 (2022-10-06) . . . 13
6.3 0.2.2 (2022-07-25) . . . 13

7 Indices and tables **15**

CHAPTER 1

django-geopostcodes

Django modelling for GeoPostcodes

NOTE: This repository does not include the actual data, you must purchase this from geopostcodes.com and then import it, as per the documentation at django-geopostcodes.readthedocs.org

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

1.1 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install django-geopostcodes, run this command in your terminal:

```
$ pip install django-geopostcodes
```

This is the preferred method to install django-geopostcodes, 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 django-geopostcodes can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/alexhayes/django-geopostcodes
```

Or download the tarball:

```
$ curl -OJL https://github.com/alexhayes/django-geopostcodes/tarball/master
```

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

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

2.3 Django Settings

Then place `django_geopostcodes` in your `INSTALLED_APPS`;

```
INSTALLED_APPS = (
    ...
    'django_geopostcodes',
    ...
)
```

2.4 Data Import

django-geopostcodes by itself only provides models and helpers, you must purchase and import the geo post code data from geopostcodes.com.

Upon purchase from geopostcodes.com you will be able to down a zip file that contains files that can be imported into django-geopostcodes.

Currently django-geopostcodes only supports localities, however feel free to fork and add support for regions and businesses.

Once you've completed your purchase from geopostcodes.com you will be able to download a ZIP file containing data in a number of formats. Inside the ZIP file there should be a folder called `CSV` and within this folder should be a file titled `GeoPC_XX_Places.csv` where `XX` is the two letter ISO country code.

You can import this file into django-geopostcodes as follows;

```
python manage.py import_localities /path/to/GeoPC_XX_Places.csv
```

CHAPTER 3

Usage

To use django-geopostcodes in a project:

```
import django-geopostcodes
```

3.1 Locality

Presently there is a single model `Locality` which contains localities as a flat structure as defined in the imported CSV.

The API is limited to just a custom query set manager `LocalitySet` at this stage.

You can query for localities as follows;

```
from django-geopostcodes.models import Locality
from django.contrib.gis.measure import D

# Find a locality by post code
melbourne = Locality.objects.get(postcode='3000')

# Find other localities 5km from Melbourne
nearby = Locality.objects.filter(point__distance_lte=(melbourne.point, D(km=5)))

# Find a locality that matches any text
localities = Locality.objects.anything_icontains('Victoria')

# Find any locality that starts with 'aus'
within_australia = Locality.objects.anything_istartswith('aus')
```

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/alexhayes/django-geopostcodes/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" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

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

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

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

4.4 Tips

To run a subset of tests:

```
$ pytest tests.test_django_geopostcodes
```

4.5 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 5

Credits

5.1 Development Lead

- Alex Hayes <alex@alution.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.3.1 (2022-10-06)

- Fix incorrect references to *django_geopostcodes* to *django-geopostcodes*.

6.2 0.3.0 (2022-10-06)

- Dropped support for Python < 3.8
- Dropped support for Django < 4.0

6.3 0.2.2 (2022-07-25)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search