Thorgate API Core Documentation

Release 0.2.1

Thorgate

May 14, 2018

Contents:

CHAPTER 1

Thorgate API Core

Opinionated API framework on top of Django REST framework

- Free software: ISC license

See `example` directory for a demo on how to use it.

Supports Python 3.5+, Django 1.11+, Django REST framework 3.6+

1.1 Features

- **API documentation automatically generated from your views**
  - General intro can be added
  - You can add example request/response data
  - Autogenerated Python *requests*-based examples
  - Not interactive yet
- **Integrates JSON API**
  - Cursor pagination with configurable page size
- Viewset classes for using different serializers and querysets for list/detail/edit endpoints
- API-specific 404 view
- Test utilities, e.g. for response validation
- **Versioning (WIP)**
  - Transformer-based approach, inspired by djangorestframework-version-transforms and Stripe

1.2 Credits

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

CHAPTER 2

Installation

2.1 Stable release

To install Thorgate API Core, run this command in your terminal:

```
$ pip install tg-apicore
```

This is the preferred method to install Thorgate API Core, 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 Thorgate API Core can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/thorgate/tg-apicore
```

Or download the tarball:

```
$ curl  -OL https://github.com/thorgate/tg-apicore/tarball/master
```

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

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

CHAPTER 3

Usage

To use Thorgate API Core in a project:

```
import tg_apicore
```

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/thorgate/tg-apicore/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

Thorgate API Core could always use more documentation, whether as part of the official Thorgate API Core 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/thorgate/tg-apicore/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 *tg-apicore* for local development.

1. Fork the *tg-apicore* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/tg-apicore.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 tg-apicore
$ cd tg-apicore/
$ 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:

```
$ make test
$ make lint
$ 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 and 3.6. Check https://travis-ci.org/thorgate/tg-apicore/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_importing
```

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:

```
$ bumpversion 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

- Rivo Laks <code@rivolaks.com>

5.2 Contributors

- Jürno Ader <jyrno42@gmail.com>

CHAPTER 6

History

6.1 0.2.1 (2018-04-14)

- Fix packaging (tg_apicore subdirs weren't included)

6.2 0.2.0 (2018-04-14)

- Added PageNotFoundView (JSON-based 404 views)
- Added DetailSerializerViewSet (different serializers and queryset for list/detail/edit views)
- Added CreateOnlyFieldsSerializerMixin, ModelValidationSerializerMixin and BaseModelSerializer
- Renamed APIDocumentationView.get_patterns() to .urlpatterns()
- Improved example app a lot. It now also includes tests that partially test tg-apicore itself

6.3 0.1.0 (2018-03-08)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search