Django DB Log Documentation

Release 0.1.2

Eduard Erja

Sep 17, 2018

Contents

Contents:

CHAPTER 1

Django DB Log

Custom DB Log Handler for Django Projects.

1.1 Documentation

The full documentation is at https://django-db-log.readthedocs.io.

1.2 Features

- Capture logs and save them in database.
- Examine logs in the administration page of the website.
- Job scheduler to delete old logs from the database.

1.3 Quickstart

Install Django DB Log:

```
pip install django-db-log-plugin
```

Add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'django_apscheduler',
    'django_db_log',
    ...
)
```

Add Django DB Log's URL patterns:

```
from django_db_log import urls as django_db_log_urls


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

Add the LOGGING configuration in the settings.py file.

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'verbose': {
           'format': '[%(asctime)s] %(levelname)s %(module)s.%(funcName)s %(lineno)d:
↪%(message)s'
        },
        'simple': {
            'format': ' %(levelname)s  %(message)s',
        },
    },
    'handlers': {
        'log_db': {
            'level': 'ERROR',
            'class': 'django_db_log.handlers.DBHandler',
            'model': 'django_db_log.models.ErrorLog',
            'expiry': 86400,
            'formatter': 'simple',
        },
    },
    'loggers': {
        'django': {
            'handlers': ['log_db'],
            'level': 'ERROR',
            'propagate': False,
        },
    },
}
```

Add the following constants in your settings file. These will be used to determine the lookup days to delete old logs from db.

```
INTERVAL_SCHEDULER_JOB_SECONDS = 43200
GENERAL_LOGS_DELETE_DAYS = 2
INFO_LOGS_DELETE_DAYS = 2
DEBUG_LOGS_DELETE_DAYS = 2
ERROR_LOGS_DELETE_DAYS = 10
```

Run migrations

```
python manage.py migrate
```

1.4 Running Tests

Does the code actually work?

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

1.5 Credits

- Django_APScheduler

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage

CHAPTER 2

Installation

At the command line:

```
$ easy_install django-db-log-plugin
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django_db_log
$ pip install django-db-log-plugin
```

CHAPTER 3

Usage

To use Django DB Log in a project, add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'django_apscheduler',
    'django_db_log',
    ...
)
```

Add Django DB Log's URL patterns:

```
from django_db_log import urls as django_db_log_urls


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

Add the LOGGING configuration in the settings.py file.

```
LOGGING = {
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'verbose': {
           'format': '[%(asctime)s] %(levelname)s %(module)s.%(funcName)s %(lineno)d:
↪%(message)s'
        },
        'simple': {
            'format': ' %(levelname)s  %(message)s',
        },
    },
```

(continues on next page)

(continued from previous page)

```
    'handlers': {
        'log_db': {
            'level': 'ERROR',
            'class': 'django_db_log.handlers.DBHandler',
            'model': 'django_db_log.models.ErrorLog',
            'expiry': 86400,
            'formatter': 'simple',
        },
    },
    'loggers': {
        'django': {
            'handlers': ['log_db'],
            'level': 'ERROR',
            'propagate': False,
        },
    },
}
```

Add the following constants in your settings file. These will be used to determine the lookup days to delete old logs from db.

```
INTERVAL_SCHEDULER_JOB_SECONDS = 43200
GENERAL_LOGS_DELETE_DAYS = 2
INFO_LOGS_DELETE_DAYS = 2
DEBUG_LOGS_DELETE_DAYS = 2
ERROR_LOGS_DELETE_DAYS = 10
```

Run migrations

```
python manage.py migrate
```

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/eduarde/django_db_log/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 DB Log could always use more documentation, whether as part of the official Django DB Log 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/eduarde/django_db_log/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_db_log* for local development.

1. Fork the *django_db_log* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django_db_log.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_db_log
$ cd django_db_log/
$ 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_db_log 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/eduarde/django_db_log/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_django_db_log
```

CHAPTER 5

Credits

5.1 Development Lead

- Eduard Erja <eduard.erja@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.0 (2018-08-29)

- First release on PyPI.