django-logutils Documentation

Release 0.4.3

Sander Smits

May 11, 2017

Contents

Contents:

CHAPTER 1

django-logutils

Various logging-related utilities for Django projects. For now, it provides a LoggingMiddleware class and an EventLogger class.

Documentation

http://django-logutils.readthedocs.org

Quickstart

Install django-logutils:

```
pip install django-logutils
```

LoggingMiddleware

LoggingMiddleware is middleware class for Django, that logs extra request-related information. To use it in your Django projects, add it to your `MIDDLEWARE_CLASSES` setting:

```
MIDDLEWARE_CLASSES = (
    ...
    'django_logutils.middleware.LoggingMiddleware',
    ...
)
```

The extra information consists of:

- event (default: 'request')
- remote ip address: the remote ip address of the user doing the request.

- user email: the email address of the requesting user, if available
- request method: post or get
- request url path
- response status code
- content length of the response body
- request time

N.B.: event can be overriden by using the `LOGUTILS_LOGGING_MIDDLEWARE_EVENT` setting in your project.

The log message itself is a string composed of the remote ip address, the user email, the request method, the request url, the status code, the content length of the body and the request time. Additionally, a dictionary with the log items are added as an extra keyword argument when sending a logging statement.

If settings.DEBUG is True or the request time is more than 1 second, two additional parameters are added to the logging dictionary: `nr_queries` that represents the number of queries executed during the request-response cycle and `sql_time` that represents the time it took to execute those queries. Slow requests are also raised to a loglevel of `WARNING`.

N.B.: the time threshold for slow requests can be overriden by using the `LOGUTILS_REQUEST_TIME_THRESHOLD` setting in your project.

EventLogger

The EventLogger class makes it easy to create dictionary-based logging statements, that can be used by log processors like Logstash. Log events can be used to track metrics and/or to create visualisations.

Here's an example of how you can use it:

```
>>> from django_logutils.utils import EventLogger
>>> log_event = EventLogger('my_logger')
>>> log_event('my_event', {'action': 'push_button'})
```

Development

Install the test requirements:

```
$ pip install -r requirements/test.txt
```

Run the tests to check everything is fine:

```
$ make test
```

To run the tests and opening the coverage html in your browser:

```
$ make coverage
```

To run flake8 and pylint, do:

```
$ make lint
```

To generate the documentation, do:

```
$ make docs
```

CHAPTER 2

Installation

At the command line:

```
$ easy_install django-logutils
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django-logutils
$ pip install django-logutils
```

CHAPTER 3

Usage

To use django-logutils in a project:

```
import django_logutils
```

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/jsmits/django-logutils/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.

Fix Bugs

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

Implement Features

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

Write Documentation

django-logutils could always use more documentation, whether as part of the official django-logutils docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/jsmits/django-logutils/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 :)

Get Started!

Ready to contribute? Here's how to set up *django-logutils* for local development.

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

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

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 2.7, 3.4, 3.5 and 3.6. Check https://travis-ci.org/jsmits/django-logutils/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

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

CHAPTER 5

Credits

Development Lead

- Sander Smits <jhmsmits@gmail.com>

Contributors

- Liam Andrew
- Arjen Vrielink
- Reinout van Rees
- Carsten Byrman

CHAPTER 6

History

0.4.3 (2017-05-11)

- Make sure that log messages are aggregable by tools like Sentry.

0.4.2 (2015-10-30)

- Support StreamingHttpResponse that doesn't have content length.

0.4.1 (2015-08-21)

- Add tests for app settings.

0.4.0 (2015-08-20)

- Make `REQUEST_TIME_THRESHOLD` a setting.

0.3.1 (2015-08-04)

- Update documentation.

0.3.0 (2015-08-04)

- Add `EventLogger` class.

0.2.5 (2015-07-31)

- Reach 100% test coverage.

0.2.4 (2015-07-31)

- Improve project structure.

0.2.3 (2015-07-30)

- Add `log_event` utility function for logging events.

0.2.2 (2015-07-29)

- Add `add_items_to_message` utility function.

0.2.1 (2015-07-29)

- More and better tests.

0.2.0 (2015-07-28)

- Release replacing previous faulty dev release.

0.1.0 (2015-07-28)

- First release on PyPI.