Django Mail Viewer Documentation

Release 2.1.0

Justin Michalicek

Dec 11, 2022

Contents

Contents:

CHAPTER 1

Django Mail Viewer

View emails in development without actually sending them.

1.1 Documentation

The full documentation is at https://django-mail-viewer.readthedocs.io.

1.2 Quickstart

Install Django Mail Viewer:

```
pip install django-mail-viewer
```

Add it to your *INSTALLED_APPS*:

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

Add Django Mail Viewer's URL patterns:

```
# You may want to only include this in development environments

urlpatterns = [
    ...
```

(continues on next page)

(continued from previous page)

```
    path('', include('django_mail_viewer.urls')),
    ...
]
```

Set your *EMAIL_BACKEND* in settings.py:

```
EMAIL_BACKEND = 'django_mail_viewer.backends.locmem.EmailBackend'
```

1.3 Features

- TODO

1.4 Running Tests

Does the code actually work?

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

1.5 TODO

- Passthrough backend - store the email for display in the views but also pass to another backend which may actually send
- Redis backend using Redis specific functionality for cleaner code and less risk of bugs vs the django cache backend
- Memcached backend
- File based backend - store each email as its own file
- Other backends? ElasticSearch? MongoDB?
- Separate views for each of html, plaintext, attachements, etc. to allow for more customization of display?

1.6 Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage

CHAPTER 2

Installation

At the command line:

```
$ pip install django-mail-viewer
```

CHAPTER 3

Usage

To use Django Mail Viewer in a project, add it to your *INSTALLED_APPS*:

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

Add Django Mail Viewer's URL patterns:

```
# You may want to only include this in development environments

# Django 2
urlpatterns = [
    ...
    path('', include('django_mail_viewer.urls')),
    ...
]

# Django 1.11
urlpatterns = [
    ...
    url(r'^', include('django_mail_viewer.urls')),
    ...
]
```

Set your *EMAIL_BACKEND* in settings.py:

```
EMAIL_BACKEND = 'django_mail_viewer.backends.locmem.EmailBackend'
```

3.1 Email Backends

Configurable email backends are supported to allow storing the emails in different storage back ends for display. There are currentl two supported email backends with more planned.

django_mail_viewer.backends.locmem.EmailBackend: The locmem backend works very similarly to Django's built in locmem backend, storing the email in a list in the local memory of the process. If the process running the server, such as *manage.py runserver* is restarted for any reason then the stored emails are lost. If you are using a multi-process or asynchronous setup such as sending emails asynchrously from a celery task or using django-channels then these emails will likely not be in the local memory for your process serving the view. If you are sending email directly in an http request/response, using celery always eager, etc. then this may work fine for you.

django_mail_viewer.backends.cache.EmailBackend: The cache backend makes use of Django's cache. By default it will use the default cache, but you can also specify a different cache to use. If you use locmem cache then you will have the same limitations as with the locmem backend. Similarly, using Django's dummy cache will result in no email being stored. If you use one of Django's built in Database, Filesystem, or Memcached backends or a third party backend such as a Redis cache backend then you will have access to your email across processes and server restarts.

django_mail_viewer.backends.database.backend.EmailBackend: The cache backend makes use of Django's ORM to store email messages in the database. By default file attachments are stored in your default media storage. You may want to implement your own model by subclassing *AbstractBaseEmailMessage* to customize where file attachments are stored such as to put them in a separate private s3 bucket.

The database backend is in its own Django app so that the models and migrations can be ignored if you do not intend to use this backend. To use it add mailviewer_database_backend to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'django_mail_viewer',
    'django_mail_viewer.backends.database.apps.DatabaseBackendConfig',
    ...
)
```

Set your *EMAIL_BACKEND* in settings.py:

```
EMAIL_BACKEND = 'django_mail_viewer.backends.database.EmailBackend'
```

If you are using your own model to store email then this model should also be specfied in the settings

```
MAILVIEWER_DATABASE_BACKEND_MODEL = 'my_app.MyModel'
```

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/jmichalicek/django-mail-viewer/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 Mail Viewer could always use more documentation, whether as part of the official Django Mail Viewer 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/jmichalicek/django-mail-viewer/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-mail-viewer* for local development.

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

```
$ git clone git@github.com:your_name_here/django-mail-viewer.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-mail-viewer
$ cd django-mail-viewer/
$ 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_mail_viewer 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/jmichalicek/django-mail-viewer/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_mail_viewer
```

CHAPTER 5

Credits

5.1 Development Lead

- Justin Michalicek <jmichalicek@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 2.1

- Added *allow-same-source-origin* to iframe sandbox so that things like *@font-face* can function on localhost
- Added testing on Django 4.1 and Python 3.11
- Stopped testing on old versions of Python and Django which had already been dropped from setup.py
- Some type hinting fixes

6.2 2.0

- Dropped Python 3.5 and 2.7 support
- Testing against Django 2.2 to 3.1
- Dropped testing Django versions less than 2.2
- Added new database backend which stores emails in a model in the database

6.3 1.0.0 (2018-04-23)

- Dropped testing of Django 1.8, 1.9 and 1.10
- Stopped using assignment_tag in favor of Django 1.9+ simple_tag functionality, definitely breaking Django 1.8
- Added testing of Django 2.0
- Updated .editorconfig, added flake8 check, isort, and yapf checks and configs

6.4 0.2.0 (2017-08-20)

- Added stats toxenv to show coverage stats
- Corrected v0.1.0 release date in history
- Added setting the Django *EMAIL_BACKEND* setting to quickstart and usage
- Added django cache backend
- Fixed handling of quoted-printable email encoding
- Dropped testing of Django 1.8, added testing of Django 1.11 and Python 3.6

6.5 0.1.0 (2016-12-23)

- First release on PyPI.