django mail admin Documentation

Release 0.1.0

Denis Bobrov

Mar 11, 2018

Contents

Contents:

CHAPTER 1

django mail admin

The one and only django app to receive & send mail with templates and multiple configurations.

CHAPTER 2

Work In progress!

2.1 Features

- Everything django-mailbox has
- Everything django-post-office has
- Database configurations - activate an outbox to send from, activate a mailbox to receive from
- Templates
- Translatable
- Mailings - using send_many() or 'cc' and 'bcc' or even recipients - all of those accept comma-separated lists of emails

2.1.1 Dependencies

- django >= 1.9
- django-jsonfield

2.2 Documentation

The full documentation is at https://django_mail_admin.readthedocs.io.

2.3 Quickstart

Q: What versions of Django/Python are supported? **A**: Take a look at https://travis-ci.org/delneg/django_mail_admin

Install django mail admin:

```
pip install django_mail_admin
```

Add it to your *INSTALLED_APPS*:

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

- Run `migrate`:

```
python manage.py migrate
```

- Set `django_mail_admin.backends.CustomEmailBackend` as your `EMAIL_BACKEND` in django's `settings.py`:

```
EMAIL_BACKEND = 'django_mail_admin.backends.CustomEmailBackend'
```

2.4 Custom Email Backends

By default, `django_mail_admin` uses custom Email Backends that looks up for Outbox models in database. If you want to use a different backend, you can do so by configuring `BACKENDS`, though you will not be able to use Outboxes and will have to set EMAIL_HOST etc. in django's `settings.py`.

For example if you want to use django-ses:

```
DJANGO_MAIL_ADMIN = {
    'BACKENDS': {
        'default': 'django_mail_admin.backends.CustomEmailBackend',
        'smtp': 'django.core.mail.backends.smtp.EmailBackend',
        'ses': 'django_ses.SESBackend',
    }
}
```

You can then choose what backend you want to use when sending mail:

```
# If you omit `backend_alias` argument, `default` will be used
mail.send(
    'from@example.com',
    ['recipient@example.com'],
    subject='Hello',
)

# If you want to send using `ses` backend
mail.send(
    'from@example.com',
    ['recipient@example.com'],
    subject='Hello',
    backend='ses',
)
```

2.5 Optional requirements

1. *django_admin_row_actions* for some useful actions in the admin interface
2. *requests* & social_django for Gmail

2.6 FAQ

Q: Why did you write this?

A: In order to get both email sending & receiving you'll have to install post_office AND django_mailbox. Even if you do, you'll have to work on admin interface for it to look prettier, somehow link replies properly etc. So I've decided merging those two and clearing the mess in between them as well as adding some other useful features.

Q: Why did you remove support for Python 2?

A: Because f*ck python2. Really, it's been 9 (NINE!) years since it came out. Go ahead and check out https://github.com/brettcannon/caniusepython3

Q: Why is it named django_mail_admin, what does it have to do with admin ?

A: Well, the first version of this package (which was living just in a really large admin.py) was used for easy mail management using standard Django admin interface.

Q: What languages are available?

A: Currently there's Russian and English languages available. Feel free to add yours:

```
source <YOURVIRTUALENV>/bin/activate
python manage.py makemessages -l YOUR_LOCALE -i venv
python manage.py compilemessages -l YOUR_LOCALE
```

Q: Why did you delete support for multi-lingual templates?

A: Well, we have django-model-translations for that. You can easily fork this app and override EmailTemplate model (models/templates.py) accordingly. I think there's no need for such an overhead in a mail-related app.

Q: I don't want my outgoing emails to be queued for sending after saving them in the admin interface, what do i do?

A: Just override OutgoingEmailAdmin's save_model method.

Q: Can i get in touch with you? I want a new feature to be implemented/bug fixed!

A: Feel free to reach me out using issues and pull requests, I'll review them all and answer when I can.

2.7 Running Tests

Does the code actually work?

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

2.8 Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage

CHAPTER 3

Installation

At the command line:

```
$ easy_install django_mail_admin
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv django_mail_admin
$ pip install django_mail_admin
```

CHAPTER 4

Usage

To use django mail admin in a project, add it to your *INSTALLED_APPS*:

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

Add django mail admin's URL patterns:

```
from django_mail_admin import urls as django_mail_admin_urls


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

CHAPTER 5

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:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/delneg/django_mail_admin/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.

5.1.2 Fix Bugs

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

5.1.3 Implement Features

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

5.1.4 Write Documentation

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

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/delneg/django_mail_admin/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 :)

5.2 Get Started!

Ready to contribute? Here's how to set up *django_mail_admin* for local development.

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

```
$ git clone git@github.com:your_name_here/django_mail_admin.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_admin
$ cd django_mail_admin/
$ 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_admin 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.

5.3 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/delneg/django_mail_admin/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

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

CHAPTER 6

Credits

6.1 Development Lead

- Denis Bobrov <delneg@yandex.ru>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 0.1.0 (2017-11-03)

- First release on PyPI.