Postorius Documentation

Release 1.0.2

Mailman Coders

May 26, 2017

Contents

1 Table of Contents 3

This is Postorius, the new official web interface for the GNU Mailman 3 list management system.

CHAPTER 1

Table of Contents

Installation

Note: This installation guide covers Postorius, the web user interface for GNU Mailman 3. To install GNU Mailman follow the instructions in the documentation.

If you are looking for an easy way to set up the whole GNU Mailman 3 suite (GNU Mailman 3, Postorius, Hyperkitty and mailmanclient), check out the mailman-bundler project on GitLab.

Install Postorius

Latest release

If you just want to install the latest release of Postorius, install it from PyPi:

```
$ pip install postorius
```

Latest dev version

If you want to always be up to date with the latest development version, you should install Postorius using git:

```
$ git clone https://gitlab.com/mailman/postorius.git
$ cd postorius
$ python setup.py develop
```

Note: This note only pertains to development installs and should not be used when doing production installs.

When setting up or running your local dev environment, you may run into some errors. You may want to consider installing mailman modules from source as changes may not yet be published to PyPI. Example usage below:

```
$ pip uninstall mailmanclient
$ pip install git+https://gitlab.com/mailman/mailmanclient.git
```

Setup your django project

Since you have now installed the necessary packages to run Postorius, it's time to setup your Django site.

You can find an example project in `example_project` in the root of `postorius'` git repository.

Change the database setting in `example_project/settings.py` to your preferred database, if you want something other than SQlite.

Note: Detailed information on how to use different database engines can be found in the Django documentation.

Third, prepare the database:

```
$ cd example_project
$ python manage.py migrate
```

This will create the `.db file` (if you ar using SQLite) and will setup all the necessary db tables.

To create a superuser which will act as an admin account for Postorius, run the following commands:

```
$ cd example_project
$ python manage.py createsuperuser
```

Running the development server

The quickest way to run Postorius is to just start the development server:

```
$ cd example_project
$ python manage.py runserver
```

Warning: You should use the development server only locally. While it's possible to make your site publicly available using the dev server, you should never do that in a production environment.

Development

This is a short guide to help you get started with Postorius development.

Development Workflow

The source code is hosted on GitLab, which means that we are using Git for version control.

Changes are not made directly in the project's master branch, but in feature-related personal branches, which get reviewed and then merged into the master branch. There is a contribution guide here, that mentions the basics about contributing to any mailman project.

An ideal workflow would be like this:

1. File a bug to suggest a new feature or report a bug (or just pick one of the existing bugs).
2. Create a new branch with your code changes.
3. Make a "merge request" to get your code reviewed and merged.

Installing and running the tests

After checkout you can run the tests using `tox`:

```
$ tox
```

By default this will test against a couple of different environments. If you want to only run the tests in a specific environment or a single module, you can specify this using the `-e` option and/or a double dash:

```
# List all currently configured envs:
$ tox -l
py27-django18
py27-django19

# Test Django 1.8 on Python2.7 only:
$ tox -e py27-django18

# Run only tests in ``test_address_activation``:
$ tox -- postorius.tests.test_address_activation

# You get the idea...
$ tox -e py27-django18 -- postorius.tests.test_address_activation
```

All test modules reside in the `postorius/src/postorius/tests` directory. Please have a look at the existing examples.

Mocking calls to Mailman's REST API

A lot of Postorius' code involves calls to Mailman's REST API (through the mailman.client library). Running these tests against a real instance of Mailman would be bad practice and slow, so `vcrpy` *cassettes* are used instead (see the vcrpy Documentation for details). These files contain pre-recorded HTTP responses.

If you write new tests, it's advisable to add a separate fixture file for each test case, so the cached responses don't interfere with other tests. The cassette files are stored in the `tests/fixtures/vcr_cassettes` directory. Check out the existing test cases for examples.

In order to record new API responses for your test case, you need to first start the mailman core, with the API server listening on port 9001. You can use the `example_project/mailman.cfg` file from the Postorius source.

Note: Make sure, you use a fresh mailman.db file.

Once the core is running, you can record the new cassette file defined in your test case by running tox with the *record* test env:

```
# This will only record the cassette files defined in my_new_test_module:
$ tox -e record -- postorius.tests.my_new_test_module

# This will re-record all cassette files:
$ tox -e record
```

View Auth

Three of Django's default User roles are relvant for Postorius:

- Superuser: Can do everything.
- AnonymousUser: Can view list index and info pages.
- Authenticated users: Can view list index and info pages. Can (un)subscribe from lists.

Apart from these default roles, there are two others relevant in Postorius:

- List owners: Can change list settings, moderate messages and delete their lists.
- List moderators: Can moderate messages.

There are a number of decorators to protect views from unauthorized users.

- `@user_passes_test(lambda u: u.is_superuser)` (redirects to login form)
- `@login_required` (redirects to login form)
- `@list_owner_required` (returns 403 if logged-in user isn't the list's owner)
- `@list_moderator_required` (returns 403 if logged-in user isn't the list's moderator)

Accessing the Mailman API

Postorius uses mailmanclient to connect to Mailman's REST API. In order to directly use the client, `cd` to the `example_project` folder and execute `python manage.py mmclient`. This will open a python shell (IPython, if that's available) and provide you with a client object connected to to your local Mailman API server (it uses the credentials from your settings.py).

A quick example:

```
$ python manage.py mmclient

>>> client
<Client (user:pwd) http://localhost:8001/3.0/>

>>> print(client.system['mailman_version'])
GNU Mailman 3.0.0b2+ (Here Again)

>>> mailman_dev = client.get_list('mailman-developers@python.org')
>>> print(mailman_dev.settings)
{u'description': u'Mailman development',
 u'default_nonmember_action': u'hold', ...}
```

For detailed information how to use mailmanclient, check out its documentation.

Deployment

Note: This guide covers deployment options of Postorius.

Nginx with uwsgi

Note: Please refer to nginx and uwsgi documentation for explanation of the shown snippets.

Below is an example uwsgi configuration file:

```
[uwsgi]

chdir           = /srv/django/mailman
module          = example_project.wsgi
virtualenv      = /srv/django/mailman/env

master          = true
processes       = 4
socket          = /run/uwsgi/mailman.sock
#chmod-socket    = 666

vacuum          = true
plugin          = python2

uid             = http
gid             = http
```

And a nginx server section to with it:

```
upstream mailman {
                server unix:///run/uwsgi/mailman.sock;
}

server {
                listen      80;
                # TODO Replace with your domain
                server_name lists.example.com;
                return 301      https://$server_name$request_uri;

}

## Config for server secured with https
server {
        listen   443;

        # TODO Replace with your domain
        server_name lists.example.com;


        ssl                     on;
        # TODO Replace with your crt and key
        ssl_certificate         /etc/nginx/keys/lists.example.com.crt;
        ssl_certificate_key     /etc/nginx/keys/lists.example.com.key;
```

```
        ssl_session_timeout     5m;
        ssl_ciphers             'AES128+EECDH:AES128+EDH';
        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
        ssl_prefer_server_ciphers       on;
        add_header Strict-Transport-Security "max-age=63072000; includeSubdomains;␣
↪preload";

        charset     utf-8;

        # max upload size
        client_max_body_size 75M;   # adjust to taste

        location /static {
                        # TODO Adjust to your static location
                        alias /srv/django/mailman/public/static;
        }

        # Finally, send all non-media requests to the Django server.
        location / {
                        uwsgi_pass  mailman;
                        include     /etc/nginx/uwsgi_params; # the uwsgi_params file␣
↪you installed
        }
}
```

Apache with mod_wsgi

Note: This guide assumes that you know how to setup a VirtualHost with Apache. If you are using SQLite, the `.db` file as well as its folder need to be writable by the web server.

These settings need to be added to your Apache VirtualHost:

```
Alias /static /srv/django/mailman/public/static
<Directory "/srv/django/mailman/public/static">
    Order deny,allow
    Allow from all
</Directory>

WSGIScriptAlias / /srv/django/mailman/srv/postorius.wsgi
<Directory "/srv/django/mailman/srv">
    Order deny,allow
    Allow from all
</Directory>
```

The first Alias serves the static files (CSS, JS, Images, etc.). The WSGIScriptAlias serves the Django application. The paths need to be changed depending on which location you have your postorius project in.

Final setup instructions

We're almost ready. But you need to create translations and collect the static files from Postorius (which resides somewhere on your pythonpath) to be able to serve them from the site directory. All you have to do is to change into the postorius project directory and run:

```
$ python manage.py compilemessages
$ python manage.py collectstatic
```

After reloading the webserver Postorius should be running!

News / Changelog

The Postorius Django app provides a web user interface to access GNU Mailman.

Postorius is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation, version 3 of the License.

Postorius is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with Postorius. If not, see <http://www.gnu.org/licenses/>.

1.1.0 – "Welcome to This World"

(2017-05-26)

- Added DMARC mitigation settings
- Switch to Allauth auth library
- Preference page improvements
- Moderation page improvements
- Django support up to Django 1.11
- Added form to edit header matches
- Domain edit form improvements
- All pipelines recognized in alter messages form
- Use django-mailman3 to share common code with HyperKitty
- Various bug fixes, code cleanup, and performance improvements

1.0.3

(2016-02-03)

- Fix security issue

1.0.2

(2015-11-14)

- Bug fix release

1.0.1

(2015-04-28)

- Help texts Small visual alignment fix; removed unnecessary links to

separate help pages. * Import fix in fieldset_forms module (Django1.6 only)

1.0.0 – "Frizzle Fry"

(2015-04-17)

- French translation. Provided by Guillaume Libersat
- Addedd an improved test harness using WebTest. Contributed by Aurélien Bompard.
- Show error message in login view. Contributed by Aurélien Bompard (LP: 1094829).
- Fix adding the a list owner on list creation. Contributed by Aurélien Bompard (LP: 1175967).
- Fix untranslatable template strings. Contributed by Sumana Harihareswara (LP: 1157947).
- Fix wrong labels in metrics template. Contributed by Sumana Harihareswara (LP: 1409033).
- URLs now contain the list-id instead of the fqdn_listname. Contributed by Abhilash Raj (LP: 1201150).
- Fix small bug moderator/owner forms on list members page. Contributed by Pranjal Yadav (LP: 1308219).
- Fix broken translation string on the login page. Contributed by Pranjal Yadav.
- Show held message details in a modal window. Contributed by Abhilash Raj (LP: 1004049).
- Rework of internal testing
- Mozilla Persona integration: switch from django-social-auto to django-browserid: Contributed by Abhilash Raj.
- Fix manage.py mmclient command for non-IPython shells. Contributed by Ankush Sharma (LP: 1428169).
- Added archiver options: Site-wide enabled archivers can not be enabled

on a per-list basis through the web UI. * Added functionality to choose or switch subscription addresses. Contributed by Abhilash Raj. * Added subscription moderation, pre_verification/_confirmation. * Several style changes.

1.0 beta 1 – "Year of the Parrot"

(2014-04-22)

- fixed pip install (missing MANIFEST) (LP: 1307624). Contributed by Aurélien Bompard
- list owners: edit member preferences
- users: add multiple email addresses
- list info: show only subscribe or unsubscribe button. Contributed by Bhargav Golla
- remove members/owners/moderator. Contributed by Abhilash Raj

1.0 alpha 2 – "Is It Luck?"

(2014-03-15)

- dev setup fix for Django 1.4 contributed by Rohan Jain

- missing csrf tokens in templates contributed by Richard Wackerbarth (LP: 996658)
- moderation: fixed typo in success message call
- installation documentation for Apache/mod_wsgi
- moved project files to separate branch
- show error message if connection to Mailman API fails
- added list members view
- added developer documentation
- added test helper utils
- all code now conform to PEP8
- themes: removed obsolete MAILMAN_THEME settings from templates, contexts, file structure; contributed by Richard Wackerbarth (LP: 1043258)
- added access control for list owners and moderators
- added a mailmanclient shell to use as a *manage.py* command (*python manage.py mmclient*)
- use "url from future" template tag in all templates. Contributed by Richard Wackerbarth.
- added "new user" form. Contributed by George Chatzisofroniou.
- added user subscription page
- added decorator to allow login via http basic auth (to allow non-browser clients to use API views)
- added api view for list index
- several changes regarding style and navigation structure
- updated to jQuery 1.8. Contributed by Richard Wackerbarth.
- added a favicon. Contributed by Richard Wackerbarth.
- renamed some menu items. Contributed by Richard Wackerbarth.
- changed static file inclusion. Contributed by Richard Wackerbarth.
- added delete domain feature.
- url conf refactoring. Contributed by Richard Wackerbarth.
- added user deletion feature. Contributed by Varun Sharma.

1.0 alpha 1 – "Space Farm"

(2012-03-23)

Many thanks go out to Anna Senarclens de Grancy and Benedict Stein for developing the initial versions of this Django app during the Google Summer of Code 2010 and 2011.

- add/remove/edit mailing lists
- edit list settings
- show all mailing lists on server
- subscribe/unsubscribe/mass subscribe mailing lists
- add/remove domains
- show basic list info and metrics

- login using django user account or using BrowserID
- show basic user profile
- accept/discard/reject/defer messages
- Implementation of Django Messages contributed by Benedict Stein (LP: #920084)
- Dependency check in setup.py contributed by Daniel Mizyrycki
- Proper processing of acceptable aliases in list settings form contributed by Daniel Mizyrycki