HyperKitty Documentation

Release 1.3.4

Mailman Coders

Sep 24, 2020

Contents

HyperKitty is a Django-based application providing a web interface to access GNU Mailman v3 archives, and interact with the lists.

The project page is https://gitlab.com/mailman/hyperkitty.

There is a demo server available, but it's also a development server, so it may be broken at the time you access it. It's usually OK though.

The authors are listed in the `AUTHORS.txt` file.

Contents:

CHAPTER 1

News / Changelog

1.1 1.3.4

(2020-XX-XX)

- Sync owners and moderators from Mailman Core for MailingList. (Fixes #302)
- Implemented a new `HYPERKITTY_JOBS_UPDATE_INDEX_LOCK_LIFE` setting to set the lock lifetime for the `update_and_clean_index` job. (Closes #300)
- Implemented a new `HYPERKITTY_ALLOW_WEB_POSTING` that allows disabling the web posting feature. (Closes #264)
- Add the ability to disable Gravatar using `HYPERKITTY_ENABLE_GRAVATAR` settings. (Closes #303)
- Replaced deprecated `ugettext` functions with `gettext`. (Closes #310)
- Fix export of Email message where the `In-Reply-To` header doesn't include the <> brackets. (Closes #331)
- We now catch a few more exceptions in `hyperkitty_import` when getting messages from a mbox. (Closes #313 and #314)
- Added a new contrib/check_hk_import script to check mboxes before running hyperkitty_import.

1.2 1.3.3

(2020-06-01)

- Allow `SHOW_INACTIVE_LISTS_DEFAULT` setting to be configurable. (Closes #276)
- Fix a bug where the user couldn't chose the address to send reply or new post as. (Closes #288)
- Improve the Django admin command reference from hyperkitty_import. (Closes #281)
- Fix `FILTER_VHOST` to work with web hosts other than the email host. (Closes #254)
- Fixed a bug where `export` can fail if certain headers are wrapped. (Closes #292)

- Fixed `hyperkitty_import` to allow odd line endings in a folded message subject. (Closes #280)
- Fixed a bug that could throw an `IndexError` when exporting messages. (Closes #293)
- Use `errors='replace'` when encoding attachments. (Closes #294)

1.3 1.3.2

(2020-01-12)

- Remove support for Django 1.11. (Closes #273)
- Skip `Thread.DoesNotExist` exception when raised within `rebuild_thread_cache_votes`. (Closes #245)
- Send 400 status code for `ValueError` when archiving. (Closes #271)
- Fix a bug where exception for elasticsearch backend would not be caught. (Closes #263)

1.4 1.3.1

(2019-12-08)

- Add support to delete mailing list. (Closes #3)
- Fix a bug where messages with attachments would skip adding the body when exporting the email. (Closes #252)
- Fix a bug where exporting mbox with messages that have attachments saved to disk would raise exception and return a corrupt mbox. (Closes #258)
- Fix a bug where downloaded attachments are returned as a memoryview object instead of bytes and hence fail to download. (Closes #247)
- Fix a bug where migrations would fail with exceptions on postgresl. (Closes #266)
- Add support for Django 3.0.
- Add support for Python 3.8 with Django 2.2.

1.5 1.3.0

(2019-09-04)

- Unread messages now have a blue envelope icon, instead of a gray one before to to make them more visible.
- Quoted text in emails have different visual background to improve readability.
- Quoted text is now visually quoted to 3 levels of replies with different visual background to improve readability.
- Add a new "All Threads" button in MailingList overview page to point to all the the threads in reverse date order. This should give a continuous list of threads.
- Fixes a bug where "All Threads" button leads to 500 page if there aren't any threads. (Closes #230)
- Add support for Django 2.2.
- Fix a bug where bad Date header could cause `hyperkitty_import` to exit with `TypeError` due to bad date type.

- Change the Overview page to remove the List of months from left side bar and convert different thread categories into tabs.
- Replace unmaintained `lockfile` dependency with `flufl.lock`.
- Remove `SingletonAsync` implementation of `AsyncTask` and use the upstream version for better maintenance.
- Run update_index job hourly by default instead of minutely for performance reasons of whoosh.
- Email body now preserves leading whitespaces on lines and wraps around line boundary. (Closes #239)
- Do not indent replies on small screens. (Closes #224)
- **Add a keyboard shortcut ? to bring up list of keyboard shortcuts.** (Closes #240)

1.6 1.2.2

(2019-02-22)

- `paintstore` is no longer a dependency of Hyperkitty. This change requires that people change their `settings.py` and remove `paintstore` from `INSTALLED_APPS`. (See #72)
- Folded Message-ID headers will no longer break threading. (#216)
- MailingList descriptions are no longer a required field. This makes HyperKity more aligned with Core. (Closes #211)

1.7 1.2.1

(2018-08-30)

- Several message defects that would cause `hyperkitty_import` to abort will now just cause the message to be skipped and allow importing to continue. (#183)
- If an imported message has no Date: header, `hyperkitty_import` will now look for Resent-Date: and the unixfrom date before archiving the message with the current date. (#184)
- Add support for Django 2.1. Hyperkitty now supports Django 1.11-2.1 (#193)

1.8 1.2.0

(2018-07-10)

- Handle email attachments returned by Scrubber as bytes or as strings with no specified encoding. (#171)
- Remove robotx.txt from Hyperkitty. It wasn't working correctly anyway. If you still need it, serve it from the webserver directly. (#176)
- Add the possibility to store attachments on the filesystem, using the `HYPERKITTY_ATTACHMENT_FOLDER` config variable.
- If a message in the mbox passed to `hyperkitty_import` is missing a `Message-ID`, a generated one will be added. (#180)
- There is a new management command `update_index_one_list` to update the search index for a single list. (#175)

1.9 1.1.4

(2017-10-09)

- Use an auto-incrementing integer for the MailingLists's id. **WARNING**: this migration will take a very long time (hours!) if you have a lot of emails in your database.
- Protect a couple tasks against thread and email deletion
- Improve performance in the cache rebuilding async task
- Drop the `mailman2_download` command. (#148)
- Adapt to the newest mailmanclient version (3.1.1).
- Handle the case when a moderated list is opened and there are pending subscriptions. (#152)
- Protect export_mbox against malformed URLs. (#153)

1.10 1.1.1

(2017-08-04)

- Fix the Javascript in the overview page
- Make two Django commands compatible with Django >= 1.10
- Fix sorting in the MailingList's cache value
- Don't show emails before they have been analyzed
- Fix slowdown with PostgreSQL on some overview queries

1.11 1.1.0

(2017-05-26)

- Add an async task system, check out the installation documentation to run the necessary commands.
- Support Django < 1.11 (support for 1.11 will arrive soon, only a dependency is not compatible).
- Switch to the Allauth login library
- Performance optimizations.
- Better REST API.
- Better handling of email sender names.
- Improve graphic design.

1.12 1.0.3

(2015-11-15)

- Switch from LESS to Sass
- Many graphical improvements

- The SSLRedirect middleware is now optional
- Add an "Export to mbox" feature
- Allow choosing the email a reply or a new message will be sent as

1.13 0.9.6

(2015-03-16)

- Adapt to the port of Mailman to Python3
- Merge KittyStore into HyperKitty
- Split off the Mailman archiver Plugin in its own module: mailman-hyperkitty
- Compatibility with Django 1.7

1.14 0.1.7

(2014-01-30)

Many significant changes, mostly on: * The caching system * The user page * The front page * The list overview page

1.15 0.1.5

(2013-05-18)

Here are the significant changes since 0.1.4:

- Merge and compress static files (CSS and Javascript)
- Django 1.5 compatibility
- Fixed REST API
- Improved RPM packaging
- Auto-subscribe the user to the list when they reply online
- New login providers: generic OpenID and Fedora
- Improved page loading on long threads: the replies are loaded asynchronously
- Replies are dynamically inserted in the thread view

1.16 0.1.4

(2013-02-19)

Here are the significant changes:

- Beginning of RPM packaging
- Improved documentation
- Voting and favoriting is more dynamic (no page reload)

- Better emails display (text is wrapped)
- Replies are sorted by thread
- New logo
- DB schema migration with South
- General style update (Boostream, fluid layout)

1.17 0.1 (alpha)

(2012-11-22)

Initial release of HyperKitty.

- login using django user account / browserid / google openid / yahoo openid
- use Twitter Bootstrap for stylesheets
- show basic list info and metrics
- show basic user profile
- Add tags to message threads

CHAPTER 2

Installation

Note: This installation guide covers HyperKitty, the web user interface to access GNU Mailman v3 Archives. To install GNU Mailman follow the instructions in the documentation: http://mailman.readthedocs.io/

2.1 Install the code

Install the HyperKitty package and its dependencies with the following commands:

```
sudo python setup.py install
```

You will also need to install the Sass CSS processor using your package manager or the project's installation documentation. You can either use the default Ruby implementation or the C/C++ version, called libsass (the binary is `sassc`). The configuration file in `example_project/settings.py` defaults to the `sassc` version, but you just have to edit the `COMPRESS_PRECOMPILERS` mapping to switch to the Ruby implementation, whoose binary is called `sass`.

Those tools are usually packaged by your distribution. On Fedora the Ruby package is named `rubygem-sass`, so you can install it with:

```
sudo yum install rubygem-sass
```

On Debian and Ubuntu, the Ruby package is available in the `ruby-sass` package, which you can install with:

```
sudo apt-get install ruby-sass
```

There is no package of libsass or sassc on either distribution today, but it is being worked on.

It is however recommended to use Virtualenv to install HyperKitty, even for a non-development setup (production). Check out *the development documentation* for details.

2.2 Setup your django project

You now have installed the necessary packages but you still need to setup the Django site (project). Example files are provided in the `example_project` subdirectory.

Change the database setting in `example_project/settings.py` to your preferred database. Edit this file to reflect the correct database credentials, the configuration variable is `DATABASES` (at the top of the file).

Or better yet, instead of changing the `settings.py` file itself, copy the values you want to change to a file called `settings_local.py` and change them there. It will override the values in `settings.py` and will make future upgrades easier.

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

2.3 Setting up the databases

The HyperKitty database is configured using the `DATABASE` setting in Django's `settings.py` file, as usual. The database can be created with the following command:

```
django-admin migrate --pythonpath example_project --settings settings
```

HyperKitty also uses a fulltext search engine. Thanks to the Django-Haystack library, the search engine backend is pluggable, refer to the Haystack documentation on how to install and configure the fulltext search engine backend.

HyperKitty's default configuration uses the Whoosh backend, so if you want to use that you just need to install the `Whoosh` Python library.

2.4 Importing the current archives

If you are currently running Mailman 2.1, you can run the `hyperkitty_import` management command to import existing archives into the mailman database. This command will import the Mbox files: if you're installing HyperKitty on the machine which hosted the previous version of Mailman, those files are available locally and you can use them directly.

The command's syntax is:

```
django-admin hyperkitty_import --pythonpath example_project --settings settings -l␣
↪ADDRESS mbox_file [mbox_file ...]
```

where:

- `ADDRESS` is the fully-qualified list name (including the `@` sign and the domain name)
- The `mbox_file` arguments are the existing archives to import (in mbox format).

The archive mbox file for a list is usually available at the following location:

```
/var/lib/mailman/archives/private/LIST_NAME.mbox/LIST_NAME.mbox
```

If the previous archives aren't available locally, you need to download them from your current Mailman 2.1 installation. The file is not web-accessible.

Before importing an archive mbox, it is a good idea to check its integrity with the hyperkitty/contrib/check_hk_import script and with Mailman 2.1's bin/cleanarch script.

After importing your existing archives, you must add them to the fulltext search engine with the following command:

```
django-admin update_index --pythonpath example_project --settings settings
```

Refer to the command's documentation for available switches.

2.5 Running HyperKitty on 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.

Edit `example_project/apache.conf` to point to your source code location.

Add following line to your apache/httpd configuration file:

```
Include "/{path-to-example_project}/apache.conf"
```

And reload Apache. We're almost ready. But you need to collect the static files from HyperKitty (which resides somewhere on your pythonpath) to be able to serve them from the site directory. All you have to do is run:

```
django-admin collectstatic --pythonpath example_project --settings settings
django-admin compress --pythonpath example_project --settings settings
```

Note: Your `django-admin` command may be called `django-admin.py` depending on your installation method.

These static files will be collected in the `example_project/static` directory and served by Apache. You should now be all set. Try accessing HyperKitty in your web browser.

2.6 Connecting to Mailman

To receive incoming emails from Mailman, you must install the mailman-hyperkitty module available on PyPI in Mailman's virtualenv, and then add the following lines to `mailman.cfg`:

```
[archiver.hyperkitty]
class: mailman_hyperkitty.Archiver
enable: yes
configuration: /path/to/example_project/mailman-hyperkitty.cfg
```

An example of the mailman-hyperkitty.cfg file is shipped with the mailman-hyperkitty package. You must set the URL to your HyperKitty installation in that file. It is also highly recommanded to change the API secret key and in the `MAILMAN_ARCHIVER_KEY` variable in `settings.py` (or `settings_local.py`).

After having made these changes, you must restart Mailman. Check its log files to make sure the emails are correctly archived. You should not see "`Broken archiver: hyperkitty`" messages.

Note: Make sure you installed the `mailman_hyperkitty` module in the same environment in which Mailman's daemon is executed otherwise it will raise an exception.

2.7 Initial setup

After installing HyperKitty for the first time, you can populate the database with some data that may be useful, for example a set of thread categories to assign to your mailing-list threads. This can be done by running the following command:

```
django-admin loaddata --pythonpath example_project --settings settings first_start
```

Thread categories can be edited and added from the Django administration interface (append `/admin` to your base URL).

You must also make sure that Mailman has generated the databases files that Postfix (or another MTA) will use to lookup the lists. Otherwise SMTP delivery will fail, and that will also impact HyperKitty when it will try to validate email addresses on registration. You can force Mailman to generate those database files with the following command:

```
mailman aliases
```

2.8 Customization

You can add HTML snippets to every HyperKitty page by using Django's TEMPLATE DIRS facility and overriding the following templates:

- `hyperkitty/headers.html`: the content will appear before the `</head>` tag
- `hyperkitty/top.html`: the content will appear before the `<body>` tag
- `hyperkitty/bottom.html`: the content will appear before the `</body>` tag

By default, HyperKitty stores the email attachments in the database. If you would rather have them stored on the filesystem, you can set the `HYPERKITTY_ATTACHMENT_FOLDER` configuration value to a directory.

Make sure that the user running the Django process (for example, `apache` or `www-data`) has the permissions to write in this directory.

If you want to disable support for gravatars in Hyperkitty, you can set `GRAVATAR_SECURE_URL = ''`. This will prevent Hyperkitty to go out to gravatar to load images for email senders.

2.9 Upgrading

To upgrade an existing installation of HyperKitty, you need to update the code base and run the commands that will update the database schemas. Before updating any of those databases, it is recommended to shut down the webserver which serves HyperKitty (Apache HTTPd for example).

To update the HyperKitty database, run:

```
django-admin migrate --pythonpath example_project --settings settings
```

After this command complete, your database will be updated, you can start your webserver again.

2.10 Asynchronous tasks

There are a few tasks in HyperKitty that need to be run at regular intervals. The `example_project` directory contains an example `crontab` file that you can put in your `/etc/cron.d` directory.

To improve performance, HyperKitty uses a distributed task queue that offloads long operations to separate processes called "workers". Those workers must be started with the following command:

```
django-admin qcluster --pythonpath example_project --settings settings
```

An example service file for systemd is provided in the `example_project` directory to start the workers at boot time, and manage them like an ordinary system service. The service file is called `qcluster.service`, make sure you edit the path to the project on the `ExecStart` line.

2.11 RPMs

Some preliminary RPMs for Fedora 21 are available from the repository described in this repo file:

```
http://repos.fedorapeople.org/repos/abompard/hyperkitty/hyperkitty.repo
```

There are also RPMs for RHEL 7 available using this repo file:

```
https://repos.fedorapeople.org/repos/abompard/hyperkitty/hyperkitty-el.repo
```

The long-term plan is to submit HyperKitty and Mailman3 for inclusion into Fedora. At the moment, the packages are rather stable, but the dependencies can change. These packages have been tested on Fedora 21 and RHEL7 with the targeted SELinux policy set to enforcing.

2.12 License

HyperKitty is licensed under the GPL v3.0

CHAPTER 3

Development

3.1 Building documentation

The full documentation is located in the "doc" subfolder. It can be generated in various formats once you have installed Sphinx. To generate the HTML documentation, run the following command:

```
make -C doc html
```

The HTML files will be available in the `doc/_build/html` directory.

The documentation can also be browsed online at: https://hyperkitty.readthedocs.org.

3.2 Communication channels

Hang out on IRC and ask questions on `#mailman` or join the mailing list `mailman-users@mailman3.org`.

3.3 Setting up HyperKitty for development

The recommended way to develop on HyperKitty is to use VirtualEnv. It will create an isolated Python environment where you can add HyperKitty and its dependencies without messing up your system Python install.

First, create the virtualenv and activate it:

```
virtualenv venv_hk
source venv_hk/bin/activate
```

Then download the components of HyperKitty:

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

You will also need to install the Sass CSS processor using your package manager or the project's installation documentation. You can either use the default Ruby implementation or the C/C++ version, called libsass (the binary is `sassc`). The configuration file in `example_project/settings.py` defaults to the `sassc` version, but you just have to edit the `COMPRESS_PRECOMPILERS` mapping to switch to the Ruby implementation, whoose binary is called `sass`.

Those tools are usually packaged by your distribution. On Fedora the Ruby package is named `rubygem-sass`, so you can install it with:

```
sudo yum install rubygem-sass
```

On Debian and Ubuntu, the Ruby package is available in the `ruby-sass` package, which you can install with:

```
sudo apt-get install ruby-sass
```

There is no package of libsass or sassc on either distribution today, but it is being worked on.

3.4 Configuration

For a development setup, you should create a file `example_project/settings_local.py` with at least the following content:

```
DEBUG = True
TEMPLATE_DEBUG = DEBUG
USE_SSL = False
```

It's also recommended to change the database access paths in the `DATABASES` and `HAYSTACK_CONNECTIONS` variables. Absolute paths are required.

If you ever want to turn the `DEBUG` variable to `False` (by removing it from `settings_local.py`), you'll have to run two additional commands then and each time you change the static files:

```
django-admin collectstatic --pythonpath example_project --settings settings
django-admin compress --pythonpath example_project --settings settings
```

Normally, to generate compressor content, you'll need to set `COMPRESS_ENABLED` to `TRUE` and `COMPRESS_OFFLINE` to `TRUE` in `settings_local.py`. However, you can force the generation of compressor content by adding the `--force` switch to the `django-admin compress` command, which will run the compressor even if the `COMPRESS` settings are not `TRUE`.

But for development purposes, it's better to keep `DEBUG = True`.

Note: Your `django-admin` command may be called `django-admin.py` depending on your installation method.

3.5 Setting up the databases

The HyperKitty database is configured using the `DATABASE` setting in Django's `settings.py` file, as usual. The database can be created with the following command:

```
django-admin migrate --pythonpath example_project --settings settings
```

HyperKitty also uses a fulltext search engine. Thanks to the Django-Haystack library, the search engine backend is pluggable, refer to the Haystack documentation on how to install and configure the fulltext search engine backend.

HyperKitty's default configuration uses the Whoosh backend, so if you want to use that you just need to install the `Whoosh` Python library.

3.6 Importing the current archives

If you are currently running Mailman 2.1, you can run the `hyperkitty_import` management command to import existing archives into the mailman database. This command will import the Mbox files: if you're installing HyperKitty on the machine which hosted the previous version of Mailman, those files are available locally and you can use them directly.

The command's syntax is:

```
django-admin hyperkitty_import --pythonpath example_project --settings settings -l␣
↪ADDRESS mbox_file [mbox_file ...]
```

where:

- `ADDRESS` is the fully-qualified list name (including the `@` sign and the domain name)
- The `mbox_file` arguments are the existing archives to import (in mbox format).

The archive mbox file for a list is usually available at the following location:

```
/var/lib/mailman/archives/private/LIST_NAME.mbox/LIST_NAME.mbox
```

If the previous archives aren't available locally, you need to download them from your current Mailman 2.1 installation. The file is not web-accessible.

Before importing an archive mbox, it is a good idea to check its integrity with the hyperkitty/contrib/check_hk_import script and with Mailman 2.1's bin/cleanarch script.

After importing your existing archives, you must add them to the fulltext search engine with the following command:

```
django-admin update_index --pythonpath example_project --settings settings
```

Refer to the command's documentation for available switches.

3.7 Running HyperKitty

If you're coding on HyperKitty, you can use Django's integrated web server. It can be run with the following command:

```
django-admin runserver --pythonpath example_project --settings settings
```

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.

3.8 Testing

Use the following command:

```
django-admin test --settings hyperkitty.tests.settings_test hyperkitty
```

All test modules reside in the `hyperkitty/tests` directory and this is where you should put your own tests, too. To make the django test runner find your tests, make sure to add them to the folder's `__init__.py`:

CHAPTER 4

Why HyperKitty?

Mailman is in need for replacement of its default Pipermail archiver. It is over 10 years old, users' expectations have changed and their requirements are more sophisticated than the current archiver can deliver on. Mailman3 is the currently under active development and it offers a pluggable architecture where multiple archivers can be plugged to the core without too much pain.

Some of the drawbacks of Pipermail :

- It does not support stable URLs.
- It has scalability issues (it was not suitable for organizations working with hundred of thousand of messages per day, e.g, Launchpad)
- The web interface is dated and does not output standards-compliant HTML nor does it take advantage of new technologies such as AJAX.

The HyperKitty archiver addresses most of the drawbacks of Pipermail.

CHAPTER 5

Copyright