HyperKitty Documentation

Release 1.1.0

Mailman Coders

May 26, 2017

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

1.1.0

(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.

1.0.3

(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

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

0.1.7

(2014-01-30)

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

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

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)

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://packages.python.org/mailman/

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 pacakge 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.

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.

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.

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. Make sure you point at the `*.txt` version of the files and not the `*.gz`.

If the previous archives aren't available locally, you need to download them from your current Mailman 2.1 installation. The `mailman2_download` management command can help you do that, its syntax is:

```
django-admin mailman2_download --pythonpath example_project --settings settings -u␣
↪URL -l LIST_NAME [-d destdir]
```

where:

- `URL` is the base URL of your current Mailman 2.1 installation, typically the part before the `/pipermail` subdirectory when you're looking at your current archives. Make sure you remember to include the 'http://' in this string.
- `LIST_NAME` is the name of the mailing-list without the domain (before the `@` sign)

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.

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.

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/hyperkitty.cfg
```

An example of the 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.

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
```

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

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.

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.

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.

License

HyperKitty is licensed under the GPL v3.0

CHAPTER 3

Development

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.

Communication channels

Hang out on IRC and ask questions on `#mailman` or join the mailing list `hyperkitty-devel@lists.fedorahosted.org`.

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 pacakge 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.

Configuration

For a development setup, you should create a `example_project/settings_local.py` file 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.

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.

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. Make sure you point at the `*.txt` version of the files and not the `*.gz`.

If the previous archives aren't available locally, you need to download them from your current Mailman 2.1 installation. The `mailman2_download` management command can help you do that, its syntax is:

```
django-admin mailman2_download --pythonpath example_project --settings settings -u␣
↪URL -l LIST_NAME [-d destdir]
```

where:

- `URL` is the base URL of your current Mailman 2.1 installation, typically the part before the `/pipermail` subdirectory when you're looking at your current archives. Make sure you remember to include the 'http://' in this string.
- `LIST_NAME` is the name of the mailing-list without the domain (before the `@` sign)

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.

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.

Testing

Use the following command:

```
django-admin test --pythonpath example_project --settings settings 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 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