fedora-happiness-packets documentation

Fedora Project contributors

Jun 10, 2023

Contents

Welcome to the documentation for fedora-happiness-packets. This is a fork of the happinesspackets.io website. This fork adds Fedora Account System authentication support and fedora-messaging integration to Happiness Packets.

For more detailed information, see the specific pages below:

CHAPTER 1

Model

1.1 Data Fields

- **`status`: Indicates the status of the message:**
  - 'pending_sender_confirmation': The sender has yet to click on the confirmation email.
  - 'sent': The sender has confirmed the email, and it has been sent to the recipient.
  - 'read': The recipient has read the email.
- `sender_name`: Name of the sender of the packet.
- `sender_email`: Email address of sender.
- `sender_email_stripped`: Email address of sender without aliases, to prevent circumvention of the blacklist.
- `sender_email_token`: Prevents CSRF by ensuring the link has been accessed from the email.
- `sender_ip`: IP address of sender: rate-limiting takes place in order to prevent resource abuse.
- `recipient_email`: Name of the recipient of the packet.
- `recipient_email`: Email address of recipient.
- `recipient_email_stripped`: Email address of recipient without aliases, to prevent circumvention of the blacklist.
- `recipient_email_token`: Prevents CSRF by ensuring the link has been accessed from the email.
- `message`: The text content of the message.
- `sender_named`: Boolean indicating whether the sender wishes to reveal their identity to the packet recipient. If false, the recipient receives an anonymous packet.
- `sender_approved_public`: Boolean indicating whether the sender wishes to make the contents of the packet public. (Both the sender and the recipient must approve making the packet public in order for it to be made available publically).

- `sender_approved_public_named`: Boolean indicating whether the sender wishes to make their identity as the packet sender public.
- `recipient_approved_public`: Boolean indicating whether the packet recipient wishes to make its contents public.
- `recipient_approved_public_named`: Boolean indicating whether the recipient wishes to make their identity as the packet recipient public.
- `admin_approved_public`: Boolean field indicating whether this message is allowed to be publically shared.

1.2 Model Functions

- `save()`: Strips sender and recipient emails, generates a message ID< and saves the message in the database.
- `send_sender_confirmation()`: Sends a confirmation email to the sender of the packet.
- `send_to_recipient()`: Sends the packet to the recipient.

1.3 Helper Functions

- `strip_email()`: Removes email aliases and all non-alphanumeric characters, with the exception of the @ symbol.

CHAPTER 2

Frequently asked questions

This page contains frequently asked questions about fedora-happiness-packets. It includes troubleshooting steps and other project details.

ERROR: Couldn't connect to Docker deamon at http+docker://localhost-is it running? Verify the logged in user is member of the docker group. To verify logged in user run:

```
sudo usermod -aG docker ${USER}
```

If logged in user is not member of the docker group add it using:

```
sudo gpasswd -a${USER}
```

No sample messages on the main page and no messages in Archives, even though both sender and receiver approved to display message publicily.

This is due to `admin_approved_public` not being set. This has to be done manually using shell. Steps to resolve:

1. Access the shell of the container web using:

   ```
   docker-compose exec web sh
   ```

2. Acess the Django shell using:

   ```
   python manage.py shell
   ```

3. Import the `Message` model and query the necessary message. [Assuming the variable `message` points to the queried object]

4. Set the `admin_approved_public` attribute to True and save the modified object:

   ```
   message.admin_approved_public = True
   message.save()
   ```

Now when you access Archives, the message can be seen.

ERROR: for fedora-happiness-packets_redis_1 Cannot start service redis: driver failed programming external connectivity on

A redis service is already running on your machine. To see what process is running use:

```
sudo netstat -lnp
```

End the running process run the web server again using `docker-compose up`. To see processes running on a particular port (eg: 0.0.0.0:6379) Use `grep` to filter for that specific port:

```
sudo netstat -tulnp | grep 6379
```

CHAPTER 3

Contributing to fedora-happiness-packets

These guidelines explain how to submit changes to fedora-happiness-packets. Following these guidelines help maintainers respond to new tickets and pull requests. Not following these guidelines may make it harder or take longer to review your change. If you have questions about any of these guidelines, please ask in the Community Operations team channels.

3.1 Goals

Our goals are the purpose of our development. All changes should align to project goals. This helps focus our development efforts and be a considerate downstream (a.k.a. forked) project. Changes that do not align to these goals may not be accepted by maintainers.

More goals may be added or changed in the future.

3.1.1 1. fedora-happiness-packets is a fork.

fedora-happiness-packets is a fork of mxsasha/happinesspackets. The upstream project is also active and still in use. As a considerate downstream, if a change could also help upstream, we should direct changes there.

If a change to our project is also useful to upstream, always file a new issue to see if upstream wants to accept a change. A positive relationship with our upstream project is important.

3.1.2 2. fedora-happiness-packets supports changes required for deployment in Fedora community.

Changes to fedora-happiness-packets should generally be Fedora-specific. This includes fedora-messaging support, Fedora-related design changes, or integrating into other parts of the Fedora community. Sometimes there are exceptions. When deciding exceptions, always consider Goal 1.

3.1.3 3. Good code is tested code.

Introducing new code means adding unit tests. Fedora Happiness Packets uses the Django test suite and pytest for testing. As a rule of thumb, consider this wisdom when writing tests:

- Use test functions and plain-old `assert` statements.
- Ideally, test functions should be *around* five lines and assert one thing.
- If you need multiple test classes, inheritance, or OOP, you might be doing it wrong.
- If you need to make imports from other test directories, you might be doing it wrong.
- If you need more than two levels of indentation, you might be doing it wrong.
- "There's a plugin for that!" Pytest has a rich plugin community with almost anything you can think of. Take advantage of them!
- Don't use `@mock.patch` or `with mock.patch`. Use pytest fixtures.[1]

3.2 Create a development environment

See *How to set up a development environment*.

3.3 Start working on a ticket

Want to work on a new ticket? Follow these three steps:

1. Check if ticket has 

label
2. Check if ticket is already assigned
3. Leave a comment in the ticket to work on it

Issues with the 

label passed maintainer review. This means they are accepted as tasks to work on. Working on a ticket without the 

label means your work may not be accepted.

If someone else is already assigned, it means the task is **already in progress**. An assigned ticket is not available to start new work. If a ticket has no updates for longer than seven days, you may follow up and ask if the assignee is still working on the ticket.

Finally, **leave a comment** in the ticket you want to work on. A maintainer will reply asking for more information or they will assign the ticket to you. When you are assigned a ticket, this means you are approved to work on it.

3.3.1 Inactive tickets

Sometimes, an assignee of a ticket may no longer have time to work on a ticket. **After five days of no updates, a ticket can be reassigned by a project maintainer.** This *DOES NOT* mean all tickets must be solved in five days. It *DOES* mean if an assignee does not respond to new comments in a ticket after five days of their last comment, it can be re-assigned. This helps to keep tickets open and available for those who have time to work on them.

[1] You can still use `mock.patch`. It has more features than pytest's monkeypatch fixture. But don't use it with a decorator, which has a problem of applying at import time nor using it in `with` statements. Instead, you would use it with `pytest-mock`, which exports the Mock API via a `mocker` fixture. The fixture ensures the enter/exit context happens at the usual time in pytest setup/teardown.

If you are working on a ticket and more than five days have passed since your last comment, please give an update when possible.

3.4 Submit a pull request

These guidelines help maintainers review new pull requests. Stick to the guidelines for faster pull request reviews.

1. Prefer gradual small changes than sudden big changes
2. Write a helpful title for your pull request (if someone reads only one sentence, will they understand your change?)
3. Address the following questions in your pull request:
    1. What is a summary of your change?
    2. Why is this change helpful?
    3. Any specific details to consider?
    4. What do you think is the outcome of this change?
4. Include screenshots of before/after if your change is a front-end change.

3.4.1 Maintainer response time

Project maintainers / mentors are committed to **no more than three days for a reply** during Fedora Summer Coding project cycles. Current maintainers are volunteers working on the project, so we try to keep up with the project as best we can. If more than three days have passed and you have not received a reply, follow up with the Community Operations team channels team. Someone may have missed your comment – no one is intentionally ignored.

Remember, using issue templates and answering the above questions in pull requests reduces response time from a maintainer to your ticket / PR.

CHAPTER 4

Guidelines for maintainers

These guidelines are intended for maintainers of fedora-happiness-packets (including Fedora Summer Coding mentors). Following these guidelines helps us keep things running smoothly. It also helps us when we are spread across different schedules and time zones. Stick to these whenever possible.

4.1 Use PASSED label to accept tickets for future work.

The PASSED label indicates a ticket passed review by a maintainer. This means it is accepted as a valid task to work on. This label is useful during a summer coding application period. Several people may look for tasks to work on. Some new tickets may need consideration or triage before they are accepted. This helps us avoid scope creep and doing too many things at once.

Remember our documentation states a person should only work on a ticket with the PASSED label. Not using this label could exclude a ticket from consideration.

4.2 Use tags to triage tickets for type of work.

There are several tags for different types of tickets (e.g. `type - docs`, `type - frontend`, `type-backend`, etc.). Use these to triage types of work. It is easier for contributors and maintainers:

- Contributors can choose tasks in their interest areas
- Maintainers can review changes in their skill area

CHAPTER 5

Authentication

The Fedora Happiness Packets project uses Open ID Connect in order to access the Fedora Account System in order to retrieve the user's email and name. This means that in order for packet sending functionality to work, the server that this project is running on must have a client ID and secret from the Fedora Infrastructure OIDC provider.

5.1 Development

In order to obtain a Client ID and Secret for a development server, simply run the `generate_client_secrets.sh` located at the project root. The script will not run if it finds a client_secrets.json file, preventing API abuse.

Alternatively, you may use the oidc-register tool to generate a client ID. However,the project expects to have a file as returned by the `generate_client_secrets.sh` script, so if you have any issues with `oidc-register`, use the script.

5.2 Staging/Production

File a ticket with the Fedora Infrastructure team: https://pagure.io/fedora-infrastructure/issues

More information on the authentication structure can be found in the Fedora wiki: https://fedoraproject.org/wiki/Infrastructure/Authentication

CHAPTER 6

How to set up a development environment

This guide includes instructions for Linux / macOS and Windows.

6.1 Pre-requisites

You need the following programs installed before proceeding (on all operating systems).

- Git
- Python 3.6.x

6.2 How to use Docker

No matter what operating system you use, you need Docker. Docker is a tool to run applications inside what are known as "containers". Containers are similar to lightweight virtual machines (VMs). They make it easier to develop, test, and deploy a web application. Fedora Happiness Packets uses Docker for local development and to deploy to the production website.

The project comes with a **Dockerfile**. A Dockerfile is the instructions for a container build tool (like Docker) to build a container. Look at the Fedora Happiness Packets Dockerfile for an example.

6.2.1 How to install Docker

Install Docker as described in the installation docs. You also need to install Docker Compose. Docker Compose is used to run multiple containers side-by-side. While developing Fedora Happiness Packets, there are a few different services that run in multiple containers, like the Django web app, the Postgres database, and more.

See below for platform specific installation guidelines:

- Docker Desktop for Mac (Docker Compose included)
- Docker Desktop for Windows (Docker Compose included)

- CentOS
- Debian
- Fedora
- Ubuntu

6.3 Run initial set-up

This section explains how to get started developing for the first time.

6.3.1 Fork and clone

First, you need to **fork** the Fedora Happiness Packets repo on Pagure.io to your Pagure account. Then, clone your fork to your workstation using **git**. For extra help, see the Pagure first steps help page. Once you clone your fork, you need to run a script to generate authentication tokens (more on this later).

```
git clone "ssh://git@pagure.io/forks/<username>/fedora-commops/fedora-happiness-
↪packets.git"
cd fedora-happiness-packets
./generate_client_secrets.sh
```

Although Docker runs this script during container build-time, please generate a local copy first. This way, new client keys are not being generated each time the container is rebuilt. This avoids rate-limiting by the authentication service.

6.3.2 Add FAS account login info

Next, you need to configure Fedora Happiness Packets with your Fedora Account System (FAS) username and password. This is used to authenticate with Fedora APIs for username search. Copy the example file `fas-admin-details.json.example` as a new file named `fas-admin-details.json`. Add your username and password into the quotes.

6.3.3 Create a project config file

Next, create a configuration file to add admin users to Fedora Happiness Packets. Like before, copy `config.yml.example` to a new file named `config.yml`. Add your name and `@fedoraproject.org` email address for ADMINS. For superuser privileges, add your FAS username to the list.

6.3.4 How to test sending emails

In the development environment, sending emails is tested in one of two ways:

- Using console
- Using third-party mail provider (e.g. Gmail

Using console

The default setup is to send emails on the console. The full content of emails will appear in the `docker-compose` console output (explained below). To see this in action, no changes are needed.

Using Gmail

Sending real, actual emails can be tested with a third-party mail provider, like Gmail. There are other mail services you can use, but this guide explains using Gmail. To test this functionality:

1. In `settings/dev.py`, un-comment the setting for `Configurations to test sending emails using Gmail SMTP`. Comment out the setting under `Configurations for sending email on console`. In `docker-compose.yml`, un-comment the ports setting in `celery` service.
2. Enable less secure apps in the Gmail account which you want to use as the host email. (It is strongly recommended to not allow less secure apps in your primary Gmail account. A separate account for testing is recommended with this setting enabled.)
3. Replace `<HOST@EMAIL.COM>` and `<HOST_EMAIL_PASSWORD>` with the email address of the above account and its password.

6.4 Start Fedora Happiness Packets with Docker Compose

Now you are ready to start Fedora Happiness Packets! You will use Docker Compose to start all the containers used at the same time (like Redis, Celery, and others). Run this command to start up the project:

```
docker-compose up
```

Once it finishes starting up, open localhost:8000 in your browser You should see the Fedora Happiness Packets home page.

Thanks to PR #235 from @ShraddhaAg, changes to Django code, HTML templates, and CSS/JavaScript are automatically reloaded while `docker-compose` is running. You should not need to rebuild the containers every time you make a change. However, sometimes you will need to rebuild the containers (e.g. adding a new dependency). This can be done with the following command:

```
docker-compose up --build
```

6.4.1 Run integration tests

Integration tests are tests that ensure an application works fully from beginning to end. Fedora Happiness Packets is not fully tested, but there are some integration tests. To run integration tests, you need to enter the container while it is running and run the test suite. Open a new window and run this command to open a shell prompt inside the Django web app container:

```
docker-compose exec web bash
```

Once inside the container, run this command:

```
docker-compose exec web ./manage.py test -v 2 -p integration_test*.py --
↪settings=happinesspackets.settings.tsting
```

6.4.2 Test `fedora-messaging` integration

To test if messages are being sent to the RabbitMQ broker, open a new terminal while `docker-compose` is running. Run the following commands:

```
docker-compose exec web bash
fedora-messaging consume --callback=fedora_messaging.example:printer
```

The messages sent to the RabbitMQ broker, when a sender confirms sending a happiness packet, will be printed in this terminal.

6.5 Alternatives to Docker

There are other ways to run Fedora Happiness Packets without containers or Docker. However, this is discouraged as current maintainers use containers to test changes and deploy Fedora Happiness Packets to production. If you choose to not use Docker and set up everything manually, you may run into unexpected issues. Project maintainers cannot easily help you if you choose this route (and may not be able to help you)! Therefore, please consider very carefully if you wish to run everything locally without containers.

6.6 Troubleshooting

6.6.1 Windows: `alpinelinux.org error ERROR: unsatisfiable constraints`

On Windows, you might get the above error when running `docker-compose`. This can be resolved by following these steps:

1. Open Docker settings.
2. Click on Network.
3. Look for "DNS Server" section.
4. It is set to *Automatic* by default. Change it to *Fixed*.
5. The IP address should now be editable. Try changing it to `1.1.1.1`.
6. Save settings.
7. Restart Docker.