pinnwand Documentation

Release 0.0.0

supakeen

Aug 06, 2020

Contents

Welcome to the documentation of the `pinnwand` pastebin software.

CHAPTER 1

Introduction

`pinnwand` is a straightforward pastebin written for Python 3.6 and up using the Tornado web framework and the sqlalchemy ORM. It uses the awesome pygments library for its syntax highlighting.

It's no nonsense and has no features aside from pasting code temporarily.

CHAPTER 2

Usage

You can run the built in `pinnwand` HTTP server through the `pinnwand http` command. This will serve up a HTTP server listening on localhost on port 8000 and use an in-memory sqlite database to store data.

If you stop the process then the database will be gone again.

For persistence you'll need to configure `pinnwand` to use another database. See the *Configuration* page for the howto.

CHAPTER 3

Installation

You can install `pinnwand` from PyPI by running pip as follows:

    pip install pinnwand

I suggest you use a virtual environment for installation. There are extended *Installation* instructions available which explain how to do so.

CHAPTER 4

Contributing

`pinnwand` is a place that will accept your first contribution to an open source project. The preferred place to start out is to visit our GitHub page and look at the issues there. If you can solve any of them then you can send a pull request. I will make sure to review your code.

If you are thinking about contributing a new feature then keep in mind that `pinnwand` is trying to stay as small and lean of a project as possible. Open a ticket first if you have a specific feature in mind.

CHAPTER 5

Table of Contents

5.1 Installation

The common way of installing `pinnwand` is by installing from PyPI. I suggest you to use a virtual environment, these prevent accidentally updating libraries that your other projects or even your operating system depend on.

For a Debian based distribution installation would look like this:

> python3 -m venv virtual-environment virtual-environment/bin/pip install pinnwand

After this you can run `pinnwand` in the following way:

> virtual-environment/bin/pinnwand http

This will start the built-in HTTP server on localhost, port 8000, and will use an in-memory sqlite3 database. This means that your pastes will be gone when the process exits.

To supply a persistent database, see the *Configuration* bit of the documentation.

5.1.1 Running on boot

If you wish to run `pinnwand` as an actual service there's a few more things we will need to take care of. This bit of the documentation has strong ideas about how to host a website and doesn't provide commands but general guidance.

This guide only applies to Linux and BSD systems though I am sure it can be applied to Windows systems as well if you read synonymous terms where necessary.

Prerequisites

- A HTTP server or proxy such as nginx, haproxy, apache2, etc.

Setup

Start by creating a separate user for your `pinnwand` this way we can make use of the filesystem.

After you've created this user it's now time to setup an environment where this users code will live. Let's get into their homedir and then perform the initial section of this document as the user you've just created.

From now on I'll assume you have `/home/youruser/virtual-environment` with `pinnwand` installed to it.

To have persistence we need to use a persistent database so install your favourite one and then install its drivers into your virtual environment.

Make sure you create a database with a user and password for pinnwand so that we can go to the next step.

We'll create a configuration file `/home/youruser/pinnwand.toml` with the following content:

```
database_uri = "mysql+pymysql://user:password@host/database"
```

If you want to configure more then read the *Configuration* section.

Now that we have all of this setup it's time to test out `pinnwand` real quick:

```
/home/youruser/virtual-environment/bin/pinnwand --configuration-path /home/youruser/
↪pinnwand.toml http
```

This should start `pinnwand` listening on localhost port 8000. Verify that this is the case and make sure to paste some data and see if it ends up in the expected database. If it does you can stop it again and we can continue to configuring the system.

I will use systemd for this example since it comes pre-installed on most of our systems nowadays.

Take the example systemd unit file from the repository and place it in `/etc/systemd/system/pinnwand.service`. Then open the file and adjust the paths to the paths you've created.

After you've done this you can `systemctl daemon-reload` and `systemctl enable pinnwand.service`. Check its status, if it has come up verify that you can connect to localhost port 8000 as well and get served with the `pinnwand` pastebin.

Now it's time to configure our webserver to forward requests to `pinnwand`. I'll use `nginx` in this example but the ideas carry over to anything you might be using.

Here's an example nginx configuration file:

```
server {
        listen 443 ssl ;
        listen [::]:443 ssl ;

        root /var/www/empty;

        server_name mypastebin.net; # managed by Certbot

        add_header X-Xss-Protection "1; mode=block" always;
        add_header X-Content-Type-Options "nosniff" always;
        add_header X-Frame-Options "SAMEORIGIN" always;
        add_header Content-Security-Policy "default-src 'self'" always;
        add_header Strict-Transport-Security "max-age=31536000; includeSubdomains;␣
↪preload" always;
        add_header Referrer-Policy "no-referrer" always;
        add_header Feature-Policy "accelerometer 'none'; camera 'none'; geolocation
↪'none'; gyroscope 'none'; magnetometer 'none'; microphone 'none'; payment 'none';␣
↪usb 'none'" always;
```

(continues on next page)

(continued from previous page)

```
        location / {
                limit_req zone=mypastebin burst=100;
                proxy_pass http://127.0.0.1:8000;
                proxy_set_header Host $host;
                proxy_set_header X-Forwarded-Proto https;
        }

        access_log /home/youruser/mypastebin.net_access.log;

        ssl_certificate /etc/letsencrypt/live/mypastebin.net/fullchain.pem; # managed␣
↪by Certbot
        ssl_certificate_key /etc/letsencrypt/live/mypastebin.net/privkey.pem; #␣
↪managed by Certbot
}
```

Place that file in `/etc/nginx/sites-enabled/mypastebin.net` and reload your nginx. It is important that you pass the Host header and protocol as `pinnwand` will use these to build its URLs.

Your pastebin is now up and running!

5.2 Configuration

`pinnwand` is configured in two ways, one is by arguments and the other is through a configuration file.

The options available are dependent on the command you're running. You can always pass the `--configuration-path` argument to `pinnwand`.

Here is a quick example:

```
pinnwand --configuration-path /tmp/foo.toml http --port 9000
```

The `http` subcommand takes a separate argument `--port` to override the default listening port (8000).

5.2.1 File

The configuration file has a bunch more properties to configure `pinnwand` with. Here's an example file:

```
# Example configuration file for `pinnwand`. Shows what you can adjust. More
# information can be found at `pinnwand`'s documentation:
# https://pinnwand.readthedocs.io/en/latest/ or you can file an issue at our
# GitHub: https://github.com/supakeen/pinnwand

# Database URI to connect to see: https://docs.sqlalchemy.org/en/13/core/engines.html
↪#database-urls
# if you want to use databases other than sqlite be sure to install the
# appropriate packages and then format this url to correspond to them.
database_uri = "sqlite:///:memory:"

# Maximum paste size you want to allow.
paste_size = 262144  # 256kB in bytes

# Preferred lexers. This list of lexers will appear on top of the dropdown
# on the website allowing you to preselect commonly used lexers. Note that the
```

(continues on next page)

(continued from previous page)

```
# names here have to be the identifiers used by pygments, not the human names.
# If empty no preferred lexers are shown.
preferred_lexers = []

# Logo path, used to render your logo. If left out the default logo will be
# used. This file must be a png file.
# logo_path = "/path/to/a/file.png"

# The page path is used to find the pages listed in the page_list
page_path = "/tmp"

# This is the whitelist of files that should exist in the `page_path`
# configuration directive. `pinnwand` will look for `$file.rst` in the
# `page_path` directory and serve it at `/$file`.
page_list = ["about", "removal", "expiry"]

# The footer in raw HTML, shown at the bottom of the page and handy to link to
# your previously configured pages.
footer = 'View <a href="//github.com/supakeen/pinnwand" target="_BLANK">source code</
↪a>, the <a href="/removal">removal</a> or <a href="/expiry">expiry</a> stories, or␣
↪read the <a href="/about">about</a> page.'

# HTML for the 'help text' that can be shown above the paste area, if left
# empty no help text will be shown.
paste_help = "<p>Welcome to pinnwand, this site is a pastebin. It allows you to share␣
↪code with others. If you write code in the text area below and press the paste␣
↪button you will be given a link you can share with others so they can view your␣
↪code as well.</p><p>People with the link can view your pasted code, only you can␣
↪remove your paste and it expires automatically. Note that anyone could guess the␣
↪URI to your paste so don't rely on it being private.</p>"
```

5.2.2 Options

database_uri

A URI as accepted by sqlalchemy for the database to use.

Default: `sqlite:///:memory``

paste_size

Maximum size of a formatted paste. This includes the HTML as generated by pygments. The size should be supplied in bytes.

Default: `262144` (256 kB).

preferred_lexers

The lexers that are shown on the homepage above all other lexers. This allows you to customize your homepage to the most-used lexers for your users.

Leaving this list empty will not show any preferred lexers. The lexer names in this list must be supported by pygments.

Default: `[]`.

logo_path

Path to a custom logo file. Needs to be readable by the user `pinnwand` runs as. Leave out of the configuration file if you want to use the default logo.

Default: `unset`.

page_path

A filesystem path where pages listed in `page_path` are looked up in. If unset the default `pinnwand` path will be used.

Default: `unset`.

page_list

List of static text pages. If set these pages will be looked up in the `page_path` variable. These files should exist in `page_path` with a `.rst` suffix.

Default: `["about", "removal", "expiry"]`

footer

HTML to render in the footer.

Default: `bunch of html`

paste_help

HTML to render above the new paste page to help users on how to use your instance.

Default: `bunch of html`

5.3 Usage

5.3.1 APIs

The `pinnwand` project has several 'APIs' and I use the word loosely here as some of these were never meant to be used as an API but they are being used as such.

Currently the only officially supported APIs are the `v1` and `curl`, the others are deprecated. That doesn't mean they'll disappear anytime soon but is an indication to users that their tooling is using endpoints that could be using the newer API which has more features such as multiple files.

Each API has several endpoints and you can find their usecases here.

v1

The `v1` API supports all current features of `pinnwand` including multi file pastes. It currently has three endpoints. All of which take JSON bodies as their inputs. Examples are provided with the requests library.

/api/v1/paste

This is the main endpoint for creating new pastes. It takes a JSON body as its input for a `POST` request, the JSON body must contain the following fields:

expiry The expiry for this paste, you can list the expiries that are valid on the `/api/v1/expiry` endpoint.

files A list of file objects.

A file object needs the following fields:

lexer The lexer to use for this file, you can retrieve a list of valid lexers from the `/api/v1/lexer` endpoint.

content Content of the file, the max filesize depends on the configuration of the `pinnwand` instance you are talking to.

name (optional) If applicable add a `name` field to set the filename of the file.

Here's an example with the `requests` library that ticks all the above boxes:

```
>>> requests.post(
...     "http://localhost:8000/api/v1/paste",
...     json={
...             "expiry": "1day",
...             "files": [
...                     {"name": "spam", "lexer": "python", "content": "eggs"},
...             ],
...     }
... ).json()
{'link': 'http://localhost:8000/74', 'removal': 'http://localhost:8000/remove/
↪KYXQLPZQEWV2L4YZM7NYGTR7TY'}a
```

To remove a paste a `GET` request to the `removal` URL returned is sufficient.

/api/v1/lexer

An endpoint to list all lexers available in the `pinnwand` instance whose API you're using. The keys returned are valid for the `lexer` field:

```
>>> requests.get("http://localhost:8000/api/v1/lexer")
{'abap': 'ABAP', 'apl': 'APL', 'abnf': 'ABNF', ...}
```

/api/v1/expiry

Used to list all valid expiries to be used in the `expiry` field. These expiries are dependent on the configuration of the `pinnwand` instance that you're talking to:

```
>>> requests.get("http://localhost:8000/api/v1/expiry")
{'1day': '1 day, 0:00:00', '1week': '7 days, 0:00:00'}
```

curl

The `curl` API provides a handy one-stop for using `curl` to submit a file to `pinnwand` quickly. It doesn't support multi file but it does give you a quick way to create a shell alias for pasting to a `pinnwand` instance.

See *Tricks* for such a shell alias.

/curl

This is an endpoint that only takes `POST` requests, the body should be formencoded, `curl` will handle this for you. The following other parameters are available:

expiry The expiry for this paste, if not supplied `1day` is selected. Valid expiries depend on the configuration of the `pinnwand` instance.

lexer The lexer to use for this paste. If not supplied `text` is selected. Valid lexers depend on the configuration of the `pinnwand` instance but are generally those provided by `pygments`.

An example of where the `/curl` endpoint comes in handy:

```
€ echo "foo" | curl -X POST http://localhost:8000/curl -F 'raw=<-'
Paste URL:   http://localhost:8000/OE
Raw URL:     http://localhost:8000/raw/GU
Removal URL: http://localhost:8000/remove/GQBHGJYKRWIS34D6FNU6CJ3B5M
€ curl http://localhost:8000/raw/GU
foo%
```

deprecated-web

In the beginning there was only the `/` endpoint so people started posting to it directly. This endpoint is really the worst one to use as it doesn't give you any useful information back in an easily readable format. You'll have to parse the data out of the response and form your own URLs for for example the `removal` URL.

/

When you throw a `POST` request at this endpoint it requires the following parameters as form encoded data:

code The code to paste.

lexer The lexer to use. Valid lexers depend on the configuration of the `pinnwand` instance but are generally those provided by `pygments`.

expiry The expiry for this paste. Valid expiries depend on the configuration of the `pinnwand` instance.

The response of this endpoint is a redirect to the URL at which the newly created paste can be viewed. The removal ID is in the `Set-Cookie` header on this response, you'll have to format it into a URL `/remove/{id}` yourself.

Here's an example using `curl` to send data to this endpoint:

```
€ curl -v http://localhost:8000/ -d 'code=foo' -d 'lexer=c' -d 'expiry=1day'
*   Trying ::1...
* TCP_NODELAY set
* Connected to localhost (::1) port 8000 (#0)
> POST / HTTP/1.1
> Host: localhost:8000
> User-Agent: curl/7.58.0
> Accept: */*
> Content-Length: 28
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 28 out of 28 bytes
< HTTP/1.1 302 Found
< Server: TornadoServer/6.0.3
< Content-Type: text/html; charset=UTF-8
```

(continues on next page)

(continued from previous page)

```
< Date: Sun, 01 Mar 2020 13:03:24 GMT
< Location: /SA
< Content-Length: 0
< Set-Cookie: removal=U35UORIU6SEEGRICOJFNIAGZBM; Path=/SA
<
* Connection #0 to host localhost left intact
```

deprecated-api

`pinnwand` provided a json based API for the bpython project early on, this API does not support multi file pastes but is in common use.

Of special note is that these endpoints do not serve json in their error responses so you should not blindly try to parse their results.

/json/new

A `POST` to this endpoint requires the following formencoded fields to be present:

lexer The lexer to use for this paste, you can retrieve a valid list of lexers on the `/json/lexers` endpoint.

code The code to paste.

expiry Expiry to use for this paste, you can retrieve a valid list of expiries on the `/json/expiries` endpoint.

filename (optional) Filename to use for the pasted file.

An example of posting to this endpoint to show its return values:

```
>>> requests.post("http://localhost:8000/json/new", data={"lexer": "python", "code":
↪"spam", "expiry": "1day"}).json()
{'paste_id': 'OI', 'removal_id': 'OQTL5MSDDKHSTHCBE7WXPRHY3Q', 'paste_url': 'http://
↪localhost:8000/OI', 'raw_url': 'http://localhost:8000/raw/OI'}
```

The returned valued are the raw ID of the paste and the raw removal ID in case you want to make your own URLs. There's also some full URLs provided to visit the paste directly, note that a removal_url is missing.

/json/remove

This endpoint can be `POST`-ed to with a removal ID you've received previously and stored. It takes one parameter:

removal_id A removal ID for a paste.

This is how you'd use it:

```
>>> requests.post("http://localhost:8000/json/remove", data={"removal_id":
↪"OQTL5MSDDKHSTHCBE7WXPRHY3Q"}).json()
[{'paste_id': 'OI', 'status': 'removed'}]
```

The return value is a bit weird here as it gives you a list.

/json/show/([A-Z2-7]+)(?:#.+)?

Use this endpoint to retrieve a previously pasted paste with an ID you have:

```
>>> requests.get("http://localhost:8000/json/show/RQ").json()
{'paste_id': 'RQ', 'raw': 'spam', 'fmt': '<table class="sourcetable"><tr><td class=
↪"linenos"><div class="linenodiv"><pre>1</pre></div></td><td class="code"><div class=
↪"source"><pre><span></span><span class="n">spam</span>\n</pre></div>\n</td></tr></
↪table>', 'lexer': 'python', 'expiry': '2020-03-02T13:56:10.622397', 'filename':␣
↪None}
```

/json/lexers

List valid lexers for this `pinnwand` instance:

```
>>> requests.get("http://localhost:8000/json/lexers").json()
{"lexer": "Lexer Name", ...}
```

/json/expiries

List valid expiries for this `pinnwand` instance:

```
>>> requests.get("http://localhost:8000/json/expiries").json()
{'1day': '1 day, 0:00:00', '1week': '7 days, 0:00:00'}
```

5.4 Tricks

Some tricks to make talking to `pinnwand` easier.

5.4.1 steck

`steck` is an in-development command line tool to talk to `pinnwand` instances. You can find it on its homepage or github:

```
pip install steck
```

5.4.2 bash alias

You can add the following bash alias to your `.bashrc`:

```
function paste-to-pinnwand() {
    cat $1 | curl -X POST http://localhost:8000/curl -F 'raw=<-'
}
```

Make sure to adjust the URL to your favourite pinnwand instance. After this you can paste files with:

```
$ paste-to-pinnwand configuration.html
Paste URL:   http://localhost:8000/OA
Raw URL:     http://localhost:8000/raw/CY
Removal URL: http://localhost:8000/remove/Z4SAXX5Y7QU2NQUT7KCQX4ZGQU
```

5.5 Autodoc

5.5.1 pinnwand.command

Collection of pinnwand's command line entry points that allow you to start a HTTP server, add and remove paste, initialize the database and reap expired pastes.

5.5.2 pinnwand.http

5.5.3 pinnwand.database

class `pinnwand.database.`**`File`**(*slug*, *raw*, *lexer='text'*, *filename=None*)

class `pinnwand.database.`**`Paste`**(*slug*, *expiry=datetime.timedelta(days=7)*, *src=None*)

The Paste model represents a single Paste.

5.5.4 pinnwand.error

exception `pinnwand.error.`**`ValidationError`**

This exception is used to indicate that a certain requst is lacking or has unacceptable data aboard.

5.5.5 pinnwand.path

Shorthands for paths for templates, static assets, etc.

5.5.6 pinnwand.configuration

5.6 Changelog

`pinnwand` is Python pastebin software that tried to keep it simple but got a little more complex.

5.6.1 v1.1.3 (20200620)

An older bug that occurs rarely resurfaced. This time a bunch of code has been written to eradicate the problem.

- Race condition in slug_create (#34)
- Fix the millibyte notation.

5.6.2 v1.1.2 (20200608)

More bugfixes to some things that were either introduced in 1.1.1 or were lower priority.

- Update our dependencies.
- Use the /static URLs directly for logo/favicon (#85)

5.6.3 v1.1.1 (20200602)

The traditional bugfix release for the previous release. No real bugs here but something to prevent CSS changes from not being loaded.

- Prevent browsers from aggressively caching (#74)

5.6.4 v1.1.0 (20200524)

The 1.1.0 release is focused on new features to improve ease of use.

- Provide a button to toggle line wrapping, contributed by Kwpolska. (#51)
- Auto-delete pastes on view when they've expired. (#63)
- Include original filename if given for paste downloads (#26)
- Provide a button to toggle opposite colorschemes. (#69)
- For pastes the first file will now have the same slug as the paste itself, this allows for users to replace part of the URL to get to raw and download links. (#64)
- Allow access to raw and download handlers through /:id/(raw|download) to let people more easily change the URL by hand when linked to a paste (#72)
- Consolidate separate pygments and pinnwand stylesheets into one.

5.6.5 v1.0.2 (20200408)

Bugfix release to deal with spaces at the front of pastes being eaten leading to wonky things when people paste pre-indented code.

- something eats spaces at the start of a paste (#68)

5.6.6 v1.0.1 (20200326)

A quick bugfix release to depend on a newer version of `pygments-better-html`.

- Empty lines don't survive copy/paste. (#67)

5.6.7 v1.0.0 (20200323)

After a period of darkness (changelog-wise) version 1.0.0 was released and this changelog created.

Python Module Index

Index