ownpaste Documentation

Release 0.1

Rafael G. Martins

June 09, 2012

CONTENTS

Author Rafael Goncalves Martins

Source code https://hg.rafaelmartins.eng.br/ownpaste/

License BSD

Version 0.1

ownpaste is a pastebin web application, designed to be used as a personal/private pastebin.

This project is motivated by the difficulty of maintain a public pastebin service nowadays, without time expiration and heavy spam filters. This is the reason why services like http://paste.pocoo.org/ are shutting down.

A simple private pastebin application, that you can run by yourself, that provides a nice RESTful API, allowing the creation of cool clients, and that can highlight your files nicely, without the risk of get offline tomorrow due to spam issues looks like a nice idea in this current scenario.

ownpaste uses the Flask web framework (and some of its extensions), the Pygments syntax highlighter and a few other well-known Python libraries.

CHAPTER

ONE

USER'S GUIDE

1.1 Server setup

This section will guide you through the alternatives for setting up and configuring ownpaste in your operating system. ownpaste is currently tested on Linux, but should works in other operating systems.

ownpaste works on Python 2.7.

ownpaste is available at the *Python Package Index* (PyPI):

http://pypi.python.org/pypi/ownpaste

1.1.1 Installing ownpaste

Manually

Download the latest tarball from PyPI, extract it and run:

```
# python setup.py install
```

Using `pip`/`easy_install`

To install blohg using `pip`, type:

```
# pip install ownpaste
```

Or using `easy_install`, type:

```
# easy_install ownpaste
```

Gentoo Linux

There's a Gentoo ebuild available in the main tree. Install it using:

```
# emerge -av www-apps/ownpaste
```

Running ownpaste from the Mercurial repository

You can also run ownpaste from the Mercurial repository. Just clone it and make sure that it is added to your Python path:

```
$ hg clone https://hg.rafaelmartins.eng.br/ownpaste/
$ cd ownpaste/
```

The `ownpaste` script does not exists in the repository, but you can run it using the following command from the repository root:

```
$ python ownpaste/
```

`ownpaste/` is the directory of the main Python package, with the ownpaste implementation.

1.1.2 Configuring ownpaste

These are the steps needed to configure ownpaste properly.

Generate password hash

ownpaste is a private pastebin application, then you need an username and a password to be able to add pastes. Password is saved in the configuration file, but for security reasons you will want it hashed.

ownpaste provides an `ownpaste` script, that have some cool commands to help you when deploying ownpaste.

The following command will ask you for the desired password, and output the hash to be used in the configuration file:

```
$ ownpaste generatepw
```

Configuration parameters

These are the configuration parameters available for ownpaste.

Please read the descriptions carefully and create your configuration file. The configuration file is an usual python file, with the following variables:

Key	Default	Description
PYGMENTS_STYLE	'friendly'	Pygments style. See Pygments documentation for reference
PYGMENTS_LINENOS	True	Enable Pygments line numbering
PER_PAGE	20	Number of pastes per page, for pagination
SQLALCHEMY_DATABASE_URI	'sqlite:////tmp/ownpaste.db'	SQL-Alchemy database string
USERNAME	'ownpaste'	Username
PASSWORD	hash of 'test'	Password hash
IP_BLOCK_HITS	10	Number of login attempts before block the user IP
IP_BLOCK_TIMEOUT	60	Timeout to remove IPs from block blacklist
TIMEZONE	'UTC'	Timezone

Please don't use the default 'test' password, it is *VERY* unsecure.

Save your configuration file somewhere.

Initializing the ownpaste database

You'll need to initialize the database with the needed tables. You can use any database system supported by SQL-Alchemy.

The `ownpaste` script provides a command to initialize the database:

```
$ ownpaste initdb --config-file=/path/to/config-file.cfg
```

Running ownpaste

You can run ownpaste using the `ownpaste` script, for tests. The built-in server can't handle a big request load, then please don't use it in production.

```
$ ownpaste runserver --config-file=/path/to/config-file.cfg
```

You can also setup the configuration file path using the environment variable `OWNPASTE_SETTINGS`. This variable should contains a string with the path of the configuration file.

Deploying ownpaste

A simple wsgi file for ownpaste looks like this:

```
from ownpaste import create_app

application = create_app('/path/to/config-file.cfg')
```

ownpaste is an usual Flask application, take a look at flask deployment documentation for instructions:

http://flask.pocoo.org/docs/deploying/

Also, make sure that you deploy ownpaste using HTTPS, to improve the security.

1.2 RESTful API documentation

ownpaste provides a neat RESTful API, that can returns HTML and JSON on demand, as required by the client, and that can receive JSON data from the client as well.

The returned content (`application/json` or `text/html`) is controlled by the `Accept:` HTTP header.

Our current API version is `1`.

This section of the documentation will explain the API endpoints and methods.

The tables with returned/received objects are related to the JSON API and usually (but not always) to the variables that can be used by Jinja templates.

The tables with query string parameters are related to any API format.

All the methods and endpoints will be able to return a JSON object, some of them can return HTML, and some of them can receive a JSON object in the body of the request.

Some methods will require basic authentication. Use the credetials created during the server setup phase.

1.2.1 Base JSON response object

All the methods, when returning JSON data, will have a common base format:

Key	Type	Description
status	String	Status of the request. `ok` or `fail`

If `status` is equals to `fail` another key will be added:

Key	Type	Description
error	String	Description of the error that happened

1.2.2 / endpoint

This endpoint returns some basic information about the ownpaste instance.

GET /

This method returns HTML or JSON.

Returned object:

Key	Type	Description
version	String	ownpaste version
api_version	String	API version
language	Object	Languages available on the ownpaste instance. Keys are the language aliases and values are the language names

1.2.3 /paste/ endpoint

This endpoint deals with the pastes itself, being able to list, add, delete, change, etc.

GET /paste/

This method returns HTML or JSON. It lists the pastes available (public or public+private) with pagination.

Query string parameters:

Key	Type	Description
page	Integer	Page index for pagination. Defaults to 1
private	Integer	If `1` will list private pastes as well. Requires authentication

Returned object:

Key	Type	Description
page	Integer	Current page, for pagination
pages	Integer	Total number of pages, for pagination
per_page	Integer	Number of pastes per page, for pagination
total	Integer	Total number of pastes in the database
pastes	List	List of objects with specific data of each paste

The `pastes` list will have objects with the following format:

Key	Type	Description
paste_id	Integer	Numeric unique ID of the paste
language	String	Language alias of the paste language
file_name	String	File name of the paste, or None
pub_timestamp	Integer	UTC Unix timestamp of the creation date
private	Boolean	Paste is private?
private_id	String	If paste is private, the paste unique ID, otherwise None
file_content_preview	String	First 5 lines of the paste file content

GET `/paste/<paste_id>/`

This method returns HTML or JSON. It returns details of a paste, by the paste public or private ID. It will requires authentication if you want to retrieve data of a private post using the public (numeric) ID.

Returned object:

Key	Type	Description
paste_id	Integer	Numeric unique ID of the paste
language	String	Language alias of the paste language
file_name	String	File name of the paste, or None
pub_timestamp	Integer	UTC Unix timestamp of the creation date
private	Boolean	Paste is private?
private_id	String	If paste is private, the paste unique ID, otherwise None
file_content	String	The full paste file content

POST `/paste/`

This method just returns JSON. It will add a new paste to the database. It requires authentication.

Received object:

Key	Type	Description
language	String	Language alias of the paste language. Optional, language will be guessed if not provided or None
file_name	String	File name of the paste. Optional, defaults to None
private	Boolean	Paste is private? Optional, defaults to False
file_content	String	The full paste file content

Returned object:

Key	Type	Description
paste_id	Integer	Numeric unique ID of the paste
language	String	Language alias of the paste language
file_name	String	File name of the paste, or None
pub_timestamp	Integer	UTC Unix timestamp of the creation date
private	Boolean	Paste is private?
private_id	String	If paste is private, the paste unique ID, otherwise None
file_content_preview	String	First 5 lines of the paste file content

PATCH `/paste/<paste_id>/`

This method just returns JSON. It will change an existing paste. It requires authentication.

Received object (all parameters are optional, and will be changed if provided):

Key	Type	Description
language	String	Language alias of the paste language
file_name	String	File name of the paste
private	Boolean	Paste is private?
file_content	String	The full paste file content

Returned object:

Key	Type	Description
paste_id	Integer	Numeric unique ID of the paste
language	String	Language alias of the paste language
file_name	String	File name of the paste, or None
pub_timestamp	Integer	UTC Unix timestamp of the creation date
private	Boolean	Paste is private?
private_id	String	If paste is private, the paste unique ID, otherwise None
file_content_preview	String	First 5 lines of the paste file content

DELETE `/paste/<paste_id>/`

This method just returns JSON. It will remove a paste from the database.

Use the `status` key from the base JSON object to know if the delete request was successful.