dropafile Documentation

Release 0.1.1

Uli Fouquet

March 30, 2015

Contents

While *dropafile* is not designed as a library, you might want to use single components for building bigger and better webservices. For instance you can use the `dropafile.DropAFileApplication` as WSGI app in other environments. dropafile - Drop a file on a webpage.

`dropafile.`**ALLOWED_PWD_CHARS = 'ABCDEFGHJKLMNPQRSTUVWXYZ23456789abcdefghjkmnpqrstuvwxyz'**
Chars allowed in passwords. We allow plain ASCII chars and numbers, with some entitites removed, that can be easily mixed up: letter *l* and number one, for instance.

class `dropafile.`**DropAFileApplication**(*password=None*, *upload_dir=None*)
Drop-A-File application.

A *werkzeug* based WSGI application providing a basic-auth protected web interface for file uploads.

password is required to access the application's service. If none is provided, we generate one for you.

upload_dir is the directory, where we store files uploaded by users. If none is given we create a temporary directory on start-up. Please note: the directory is not removed on shutdown.

authenticate()
Send 401 requesting basic auth from client.

Send back 401 response to client. Contains some HTML to display an 'Unauhorized' page. Should make browsers ask users for username and password.

check_auth(*request*)
Check basic auth against local password.

We accept any username, but only *the* one password. Returns `True` in case of success, `False` otherwise.

request must contain basic-auth authorization headers (as set by browsers) to succeed.

handle_uploaded_files(*request*)
Look for an upload file in *request*.

If one is found, it is saved to *self.upload_dir*.

password = None
the password we require (no username neccessary)

upload_dir = None
a path where we store files uploaded by users.

`dropafile.`**create_ssl_cert**(*path=None*, *bits=4096*, *days=2*, *cn='localhost'*, *country='US'*, *state=''*, *location=''*)
Create an SSL RSA cert and key in directory *path*.

Returns a tuple *(certificate_path, key_path)*.

path A directory, where certificate and key can be stored. If none is given, we create a temporary one.

Default attribute values of the certificate are read from a package-local SSL configuration file `openssl.conf`.

Some attribute values can be overridden:

bits number of bits of the key.

days number of days of validity of the generated certificate.

cn Common Name. Put the domain under which the app will be served in here.

state* and *location will be empty by default.

`dropafile.`**execute_cmd**(*cmd_list*)
Excute the command *cmd_list*.

Returns stdout and stderr output.

dropafile.**get_random_password**()
Get a password generated from *ALLOWED_PWD_CHARS*.

The password entropy should be >= 128 bits. We use *SystemRandom()*, which should provide enough randomness to work properly.

dropafile.**get_ssl_context**(*cert_path=None*, *key_path=None*)
Get an SSL context to serve HTTP.

If *cert_path* or *key_path* are `None`, we create some. Then we add some modifiers (avail. with Python >= 2.7.9) to disable unsafe ciphers etc.

The returned SSL context can be used with Werkzeug *run_simple*.

dropafile.**get_store_path**(*directory*, *filename*)
Get a path where we can safely store a file.

The file should be stored in *directory* under name *filename*. If *filename* already exists in *directory*, we construct new names by appending '-<NUM>' to the original filename, where `<NUM>` is a number counting up.

dropafile.**handle_options**(*args*)
Handle commandline options.

Expects the arguments passed to dropafile as a list of arguments. *args* is expected to represent `sys.argv[1:]`, i.e. the arguments when called, without the programme name.

Returns parsed options as provided by `argparse`.

dropafile.**run_server**(*args=None*)
Run a *werkzeug* server, serving a `DropAFileApplication`.

Called when running *dropafile* from commandline. Serves a `DropAFileApplication` instance until aborted.

Options *argv* are taken from commandline if not specified.

Generates a password and temporary SSL certificate/key on startup unless otherwise requested in options/args.

CHAPTER 1

License & Copyright

1.1 Copyright

dropafile and this documentation is:

`static/dropzone.js` and `static/dropzone.css` are :

1.2 License

All parts of *dropafile*, except files *static/dropzone.js*, *static/dropzone.css*, and files in *docs/* are covered by the GPL 3.0. Full license text available in *LICENSE*

static/dropzone.js and *static/dropzone.css* are covered by the MIT (Expat) license.

If that license does not fit your needs and you need a copy of the software licensed differently, please contact *uli <at> gnufix <dot> de*.

This *documentation* is licensed under a Creative Commons Attribution 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

CHAPTER 2

Changes

2.1 0.1.1 (2015-03-30)

- Turned former *dropafile* module into a Python package. This is to fix installation behvior where data files are installed in different locations depending on install tool. See diceware bug #1.

  The problem was revealed by conorsch for the diceware package.

2.2 0.1 (2015-03-22)

- Initial release.

CHAPTER 3

dropafile

Drop me a file, securely.

| documentation | sources | issues

dropafile provides an HTTPS-secured webapp where users can drop files.

It is meant as a channel to deliver documents in a not-too-unsecure manner. For instance as a quickly installable workaround if people are not able or willing to use GnuPG or similar, although they have sensible documents to send.

dropafile is written in Python (server parts) and uses the dropzonejs JavaScript library (client parts). The builtin server is based on Werkzeug.

3.1 Install

As a user, run:

```
$ pip install dropafile
```

then, start the local server:

```
$ dropafile
Creating temporary self-signed SSL certificate...
Done.
Certificate in: /tmp/tmp1y2bgh/cert.pem
Key in:         /tmp/tmp1y2bgh/cert.key
Password is: H93rqnsrdEXD2ad3rQwdWqZ
 * Running on https://localhost:8443/ (Press CTRL+C to quit)
```

The server will provide SSL. Users can access *dropafile* sevice pointing their browsers to the location given. The page is protected by basic auth. Users will have to provide an arbitrary user name and the password displayed on the commandline at startup (which changes with restart).

The *–help* option will display all available options:

```
$ dropfile --help
usage: dropafile [-h] [--host HOST] [-p PORT] [-s PASSWORD]

Start dropafile app.

optional arguments:
  -h, --help            show this help message and exit
  --host HOST           Host we bind to. An IP address or DNS name.
                        `localhost` by default.
```

```
  -p PORT, --port PORT  Port we listen at. An integer. 8443 by default.
  -s PASSWORD, --secret PASSWORD
                        Password to access dropafile. If none is given we
                        generate one.
```

Whenever a user sends a file, the path is displayed on the commandline.

3.2 Developer Install

Developers should install a virtualenv first:

```
$ virtualenv -p /usr/bin/python2.7 py27  # for Python2.7
```

See *tox.ini* for all Python versions supported.

Activate the virtualenv:

```
$ source py27/bin/activate
(py27) $
```

Now build the devel environment:

```
(py27) $ python setup.py dev
```

You can run tests like this:

```
(py27) $ py.test
```

Tests for all supported (and locally available) Python vesions can be run by:

```
(py27) $ pip install tox  # neccessary only once per virtualenv
(py27) $ tox
```

CHAPTER 4

Indices and tables

- *genindex*
- *modindex*
- *search*

d

A

C

D

E

G

H

P

R

U