opmuse Documentation

Release 0.2.2

Mattias Fliesberg <mattias@fliesberg.email>

Aug 09, 2019

Contents

opmuse is a web application to play, organize, share and make your music library social.

These pages document development of opmuse. If you want to use opmuse you should go to the opmuse website. If you actually want to develop you can start by reading *Contributing*. Then you can take a look at *Getting Started* for instructions on how to setup an environment etc.

CHAPTER 1

Other Resources

- opmuse website.
- GitHub repository.
- GitHub issue tracker.
- #opmuse at OFTC.
- opmuse on Twitter.
- opmuse on Open Hub.

CHAPTER 2

Index

2.1 Contributing

Here's some rules that should be followed and things to think about when contributing code to opmuse.

2.1.1 Code style

Coding styles for the different languages we use. One thing to keep in mind for all of these is that readability is more important than convention and convention is more important than performance.

Python

We use PEP8 but we have a max line length of 120 chars, though we try to keep the lines at around 80 chars.

Javascript

For Javascript we follow a style based on the Crockford code conventions with a few extra rules and exceptions.

- Lines shouldn't be longer than 120 chars but try to keep them at around 80 chars.
- Everything should be a requirejs module.
- Singleton requirejs modules should be lowercase.
- Class requirejs modules should start with uppercase.
- requirejs dependencies should be declared on seperate lines
- 'use strict' should be used.

Here's a simple example.

```
define([
        'module1',
        'module2',
    ], function (module1, module2) {

    'use strict';

    // code
});
```

Jinja

As there really isn't any good Jinja style checkers or even style guides out there just try to think like PEP8 when coding Jinja. Use 4 spaces for indentation, maximum line length of 120 chars but try to keep them at around 80 chars. Also, indent both HTML tags and Jinja control structures.

Here's a simple example.

```
<ul>
    {% for item in items %}
        <li>
            {{ item.name }}
        </li>
    {% endfor %}
</ul>
```

Less & CSS

Here's some guidelines to follow for Less and CSS.

- 4 spaces for indentation
- Max line length of 120 chars but try to keep them at around 80 chars
- Curly brackets on same line as selector
- Seperate selectors with comma AND newline.

Here's a simple example.

```
ul li,
dl dt {
    margin: 0;
    padding: 0;
}
```

Shell script

Use 4 spaces, max line length of 120 chars but try to keep them at around 80 chars.

2.1.2 Git

Try to follow this style guide when writing commit messages.

2.2 Getting Started

2.2.1 Docker

You can use the opmuse-dev docker image for development.

This will get a copy of the repo from the docker image and mount it inside the container.

```
$ docker export $(docker run -d inty/opmuse-dev) | tar xf - root/opmuse --strip-
↪components=1
$ cd opmuse
$ docker run -d -p 8080:8080 --name=opmuse-dev -v `pwd`:/root/opmuse inty/opmuse-dev
```

To get cherrypy's output do this

```
$ docker logs -f opmuse-dev
```

To get a shell

```
$ docker exec -it opmuse-dev /bin/bash
```

2.2.2 Requirements

You need **Python 3.5**, **ffmpeg**, **ImageMagick**, **nodejs**, **npm**, **yarn**, **rsync** and a Linux environment. This has only been tested on Debian and Exherbo but most other Linux distros should do as well as other similar *nix OSes. If you're on Windows you're on your own.

2.2.3 Install

```
$ git clone https://github.com/opmuse/opmuse.git
$ cd opmuse
$ virtualenv -p python3 ./virtualenv
$ source virtualenv/bin/activate
$ pip install -r requirements.txt
$ yarn
$ yarn build:dev
$ cp config/opmuse.dist.ini config/opmuse.ini
$ edit config/opmuse.ini # fix library.path
```

If you just want to use **SQLite**.

```
$ ./console database create
```

If you want to use **MySQL** instead of SQLite (MySQL is recommended).

```
$ pip install -r mysql-requirements.txt
$ edit config/opmuse.ini # fix database.url
$ ./console database create
```

If you want some additional dev tools and stuff (repoze.profile, colorlog), install 'em

```
$ source virtualenv/bin/activate
$ pip install -r dev-requirements.txt
$ ./console cherrypy -- -p # start with repoze.profile (access it at /__profile__)
```

You probably want fixtures for some default data (an admin account with password admin for one).

```
$ ./console database fixtures
```

Then you start the whole debacle with

```
$ ./console cherrypy
```

2.2.4 Upgrading

When you do a git pull some of these might be required.

```
$ merge config/opmuse.dist.ini config/opmuse.ini

$ source virtualenv/bin/activate
$ pip install --upgrade -r requirements.txt
$ pip install --upgrade -r mysql-requirements.txt
$ pip install --upgrade -r dev-requirements.txt
$ yarn
$ yarn build:dev
$ ./console database reset # will initiate rescan, might not be required
$ ./console database update
```

2.3 Testing

We use pytest to run and write our tests. This is how we run the tests, assuming you've setup a dev environment according to *Getting Started* first.

```
$ pip install -r dev-requirements.txt
$ pytest opmuse/test/
```

2.3.1 Regular tests

First we have regular tests for services and utilities. They're just plain test classes optionally with some setup and teardown methods for the database and such.

2.3.2 Controller tests

Second we have controller tests that utilizes cherrypy's test framework to test controllers.

2.4 Setup APT Repo

These docs show you how to setup an APT repo for opmuse with reprepro and build opmuse's .deb packages. It is done in the inty/opmuse-build docker image here but can of course be done anywhere that has the right stuff.

```
$ cd /srv/opmuse
$ git pull # or whatever to get the new code
$ source virtualenv/bin/activate
$ pip install -r requirements.txt
$ deactivate
$ yarn
```

Optionally set the lastfm key and secret.

```
$ python3 setup.py setopt --command global --option lastfm.key --set-value KEY
$ python3 setup.py setopt --command global --option lastfm.secret --set-value SECRET
```

Start the build

```
$ ./scripts/build-debs.sh /srv/repo buster --debug
```

2.4.1 Using

You can test it out like this

```
$ apt-get update --allow-insecure-repositories
$ apt-get install opmuse --allow-unauthenticated
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search