GMusicProcurator Documentation

Release 0.5.0

Mark Lee

Sep 27, 2017

Contents

A proxy/HTML5 frontend for Google Music streaming. This mini webapp is inspired by GMusicProxy.

CHAPTER 1

Features

- XSPF playlists
- Option to ID3v2-tag MP3 streams
  - Option to embed album art in MP3 streams
- Browse playlists via an HTML5 frontend
- Stream playlists in the browser without using Flash or other browser plugins, in browsers that support HTML5 Audio

CHAPTER 2

Table of Contents

Installation

Requirements

Backend

- Python 2.7 (only tested with CPython)
- virtualenv (optional, but recommended)
- git (some packages are only installable via version control)

Frontend

- Either libsass-python or the reference implementation of Sass (which requires Ruby)
- Node.js + NPM

Browser

Any web browser which supports the HTML5 audio element is supported, except IE9, due to the layout CSS.

Instructions

Create a config file. The location depends on the OS where you're installing this app:

OS X: `~/Library/Application Support/gmusicapi/gmusicprocurator.cfg`

Linux: `~/.config/gmusicapi/gmusicprocurator.cfg`

Windows: *If you want to use this on Windows, let me know. I have no idea whether it will work correctly.*

The contents of the file will look like this:

```
GACCOUNT_EMAIL = 'my-google-account@gmail.com'
```

Then run the following (lines that start with # are comments, not commands):

```
# Get the code
user@host:Code$ git clone https://github.com/malept/gmusicprocurator.git
user@host:Code$ cd gmusicprocurator
# Create a new virtual environment
user@host:gmusicprocurator$ virtualenv venv
user@host:gmusicprocurator$ source venv/bin/activate
(venv)user@host:gmusicprocurator$ pip install -r requirements.txt
# Only run the next line if you wish to use libsass-python instead of the
# Ruby version of Sass:
(venv)user@host:gmusicprocurator$ pip install libsass
(venv)user@host:gmusicprocurator$ python -m gmusicprocurator set_password
```

The last command will activate an interactive prompt that will store your Google account password (or, if your account has two-factor authentication enabled, your application-specific password) into the operating system's password storage service.

Once your password is set, you will need to associate GMusicProcurator with one of your mobile devices. Run the following command to list the devices:

```
(venv)user@host:gmusicprocurator$ python -m gmusicprocurator list_devices --no-desktop
```

Select one of them and add the following to the config file from above (substituting `REPLACE_ME` with the ID, which is after the colon in the device ID printout):

```
GACCOUNT_DEVICE_ID = 'REPLACE_ME'
```

If you do not want to run the frontend, add the following to the config file:

```
GMP_FRONTEND_ENABLED = False
```

Once the config file is saved, the server can be started.

```
(venv)user@host:gmusicprocurator$ python -m gmusicprocurator runserver
```

By default, it runs at `localhost:5000`. For assistance on how to change these settings, run `python -m gmusicprocurator runserver --help`.

Currently, the proxy assumes that you know the playlist ID. You can access the (XSPF) playlist in the media player of your choice via the URL `http://localhost:5000/playlists/$PLAYLIST_ID`, replacing `$PLAYLIST_ID` with the proper playlist ID.

Frontend-specific

If you want to run the frontend as well, run the following before starting the server:

```
(venv)user@host:gmusicprocurator$ npm install --production
(venv)user@host:gmusicprocurator$ node_modules/.bin/bower install -p
```

Via Saltstack

If you use the Saltstack systems management software, a formula has been written so that most of GMusicProcurator's dependencies can be installed automatically. If using the frontend, the bower dependencies still need to be installed "manually".

App Configuration

App-specific Flask settings.

Standard

`gmusicprocurator.default_settings.`**`GMP_FRONTEND_ENABLED`**

Defaults to `True`. If disabled, the code and views related to the frontend are not loaded. For example, `GET / HTTP/1.1` will return a `404`.

`gmusicprocurator.default_settings.`**`GMP_NODE_MODULES_DIR`**

Defaults to the `node_modules` directory in the top-level directory of the repository. This is the path where all of the Node-based asset utilities are installed.

`gmusicprocurator.default_settings.`**`GMP_SONG_FILTERS`**

A tuple of callable filters used on streaming MP3 data. By default, it looks like:

```
GMP_SONG_FILTERS = (
    'add_id3_tags_to_mp3',
)
```

Tuple items can be either strings (built-in to the app) or callables. Callables have the following signature:

```
def (str song_id, io.BytesIO data) -> io.BytesIO
```

`gmusicprocurator.default_settings.`**`GMP_EMBED_ALBUM_ART`**

Embed album art in the songs' ID3 tags (assuming that ID3 tags are being embedded in the MP3s). Defaults to `False`.

Development

Settings that should only be configured if you are developing GMusicProcurator and/or you know what you're doing.

`gmusicprocurator.default_settings.`**`GMP_OFFLINE_MODE`**

If set to `True`, the proxy views will only return the HTTP status code `503` (Service Unavailable). It is on by default only when Read the Docs is building the documentation.

`gmusicprocurator.default_settings.`**`GMP_MEMORY_PROFILER`**

Uses `heapy` to examine what objects are using the most memory in the app. Requires installing guppy (via `pip install guppy`). When the server is running, send the `SIGUSR1` signal to the main process, and it will print out a frequency table of allocated objects, and shut down the server.

Google

`gmusicprocurator.default_settings.`**`GACCOUNT_EMAIL`**

The Google Account email address that has access to Google Music.

`gmusicprocurator.default_settings.`**`GACCOUNT_DEVICE_ID`**
The mobile device ID to use to access Google Music. See *the installation docs* for details.

API Documentation

Proxy HTTP API

`GET /playlists/all_songs`
Retrieve a special playlist that contains all songs owned by the user.

By default, it returns an XSPF-formatted playlist. If `application/json` is preferred in the `Accept` HTTP header, it will pass through the JSON representation that is returned by GMusic.

`GET /favicon.ico`
favicon.ico route.

From: http://flask.pocoo.org/docs/patterns/favicon/

`GET /playlists`
Retrieve all of the logged in user's playlists.

By default, it returns an XSPF-formatted playlist. If `application/json` is preferred in the `Accept` HTTP header, it will return the JSON representation that is returned by GMusic.

`POST /search`
Search All Access for artists/albums/tracks.

Requires a JSON payload with one key: `query`. Returns the JSON results directly from the API.

`GET /`
Main page of the frontend.

`GET /playlists/`(*playlist_id*)
Retrieve the metadata for a given playlist.

By default, it returns an XSPF-formatted playlist. If `application/json` is preferred in the `Accept` HTTP header, it will return the JSON representation that is returned by GMusic.

`GET /artists/`(*artist_id*)
Retrieve the artist metadata from the Google Music API in JSON.

`GET /albums/`(*album_id*)
Retrieve the album metadata from the Google Music API.

By default, it returns an XSPF-formatted playlist. If `application/json` is preferred in the `Accept` HTTP header, it will return the JSON representation that is returned by GMusic.

`GET /songs/`(*song_id*)
Retrieve the song data for a given store ID.

By default, it returns the MP3. If `application/json` is preferred in the `Accept` HTTP header, it will pass through the JSON representation that is returned by GMusic.

Contributing to GMusicProcurator

This project is hosted at GitHub. I gladly accept both issues and pull requests.

Filing Issues

Issues include bugs, feedback, and feature requests. Before you file a new issue, please make sure that your issue has not already been filed by someone else. In addition to the GitHub issues UI, a kanban board is available to provide a way to show the relative priority and status of open issues.

When filing a bug, please include the following information:

- Operating system. If on Linux, please also include the distribution name and version.
- Python version that is running GMusicProcurator, by running `python -V`.
- Installed Python packages, by running `pip freeze`.
- Any relevant app settings.
- A detailed list of steps to reproduce the bug.
- If the bug is a Python exception, the traceback will be very helpful.
- If the bug is related to the frontend, a screenshot will be helpful, along with the browser name and version that is being used.

Pull Requests

Please make sure your pull requests pass the continuous integration suite, by running `tox` before creating your submission. (Run `pip install tox` if it's not already installed.) The CI suite is also automatically run for every pull request, but at this time it's faster to run it locally. Additionally, it would probably be in your best interests to add yourself to the `AUTHORS.rst` file if you have not done so already.

When you submit your PR, if you have changed CoffeeScript files, Hound CI will make comments about its conformity to the code style guide as described in the *.coffeelint.json* file in the top level of the repository.

Development Environment

A Vagrant environment is available for developing `gmusicprocurator`. Run the following command in the top-level source directory (once Vagrant is installed):

```
user@host:gmusicprocurator$ vagrant up
```

...and it will install all of the Python dependencies in a virtualenv, and the other dependencies (e.g., the node.js-based ones) globally. You can then log into the virtual machine and install the package in develop mode:

```
user@host:gmusicprocurator$ vagrant ssh
# ...
vagrant@vagrant:~$ source .virtualenv/bin/activate
(.virtualenv)vagrant@vagrant:~$ pip install -e /vagrant
```

CHAPTER 3

Legal

This web application is licensed under the terms of the GNU General Public License (GNU GPL), version 3 or later, unless otherwise noted in the source files.

The documentation is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0).

The favicon is the "Tango Music Charts" icon. According to the source page, it is licensed under a Creative Commons Attribution-Share Alike 3.0 License.

This project is not affiliated in any way to Google or any of Google's music apps.

HTTP Routing Table

/

/albums

/artists

/favicon.ico

/playlists

/search

/songs

Python Module Index

Index