MishMash Documentation

Release 0.3.3

Travis Shirk

Mar 01, 2020

Contents

Contents:

CHAPTER 1

MishMash

Music database and web interface.

1.1 Features

- MishMash is a music database using Python and SQLAlchemy.
- A command-line tool for building and managing a music database.
- Web browser interface (using Pyramid) for browsing your music library.
- Uses eyeD3 for reading MP3s and ID3 metadata.
- Support and tested with Python 3.6 and Postgresql. SQLite is periodically tested with, but future features may not be supported (e.g. full text search).
- Free software: GNU GPL v3.0 license

1.2 Getting Started

```
$ mishmash info
/\/\_____  .__        .__         _____                  .__   /\/\
\(\(     \ |__| _____|  |__    /     \ _____    _____|  |__\(\(
  /  \ /  \|  |/  ___/  |  \  /  \ /  \\__  \  /  ___/  |  \
 /    Y    \  |\___ \|   Y  \/    Y    \/ __ \_\___ \|   Y  \
 \____|__  /__/_____  >___|  /\____|__  (____  /_____  >___|  /
         \/        \/     \/         \/     \/     \/     \/

Version              : 0.3
Database URL         : sqlite:////~/mishmash.db
Database version     : 0.3
```

(continues on next page)

(continued from previous page)

```
Last sync            : Never
Configuration files  : <default>


=== Music library ===
0 music tracks
0 music artists
0 music albums
0 music tags
```

Surprise, you now have an empty sqlite database in the current directory. Let's leave it here for now, it can be located elsewhere or use a different database using command line arguments and/or environment variables. Pretty useless without any music.:

```
$ mishmash sync ~/Music/Melvins
Syncing library 'Music': paths=['~/Music/Melvins/']
Syncing directory: ~/Music/Melvins/
Syncing directory: ~/Music/Melvins/1984 - Mangled Demos
Adding artist: Melvins
Syncing directory: ~/Music/Melvins/1986 - 10 Songs
Adding album: 10 Songs
Adding track: ~/Music/Melvins/1986 - 10 Songs/Melvins - 01 - Easy As It Was.mp3
Updating album: 10 Songs
...
== Library 'Music' sync'd [ 8.73s time (45.9 files/s) ] ==
401 files sync'd
401 tracks added
0 tracks modified
0 orphaned tracks deleted
0 orphaned artists deleted
0 orphaned albums deleted
```

Use your database as you wish. Browse it with *mishmash web*, or use one of its management commands.

Check out the Unsonic project for streaming capabilities.

CHAPTER 2

Installation

2.1 Using pip

At the command line:

```
$ pip install mishmash
$ pip install mishmash[web]
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv MishMash
$ pip install mishmash
```

2.2 Using a source distribution

At the command line:

```
$ tar zxf mishmash-0.3.3.tar.gz
$ cd mishmash-0.3.3
$ python setup.py install
```

2.3 From GitHub

At the command line:

```
$ git clone https://github.com/nicfit/MishMash
$ cd mishmash
$ python setup.py install
```

Additional dependencies should be installed if developing MishMash:

```
$ pip install -r requirements/dev.txt
```

2.4 Dependencies

All the required software dependencies are installed using either `requirements/default.txt` files or by `python install setup.py`.

CHAPTER 3

Usage

3.1 Configuration

MishMash ships with a default configuration that should work out of the box with no extra additional settings by using a SQLite database saved in `${HOME}/mishmash.db`. Running `mishmash info` with demonstrate this, afterwards `mishmash.db` will exist in your home directory and be initialized with the database schema.

```
$ mishmash info
$ sqlite3 /home/travis/mishmash.db
sqlite> select * from artists;
1|Various Artists|Various Artists|2014-10-11 01:12:10.246406|||
sqlite>
```

To see the current configuration use info command's `--default-config` option. You may wish to capture this output for writing custom configuration files.

```
$ mishmash --default-config
[mishmash]
sqlalchemy.url = sqlite:////home/travis/mishmash.db

[loggers]
keys = root, sqlalchemy, eyed3, mishmash

[handlers]
keys = console

[formatters]
keys = generic

[logger_root]
level = INFO
handlers = console

... more config ...
```

The first change most users will want to do is change the database that MishMash uses. The `-D/--database` option make this easy. In this example, information about ./mymusic.db SQLite database and the mymusic PostgreSQL database is displayed.

```
$ mishmash --database=sqlite:///mymusic.db info
$ mishmash -D postgresql://mishmash@localhost:5432/mymusic info
```

In you wish to make additional configuration changes, or would like to avoid needing to type the database URL all the time, a configuration is needed. The file may contain the entire configuration or only the values you wish to change (i.e. changes are applied to the default configuration). With the settings saved to a file use the `-c/--config` option to have MishMash use it. In this examples the database URL and a logging level are modified.

```
$ cat example.ini
[mishmash]
sqlalchemy.url = postgresql://mishmash@localhost:5432/mymusic

[logger_sqlalchemy]
level = DEBUG

$ mishmash -c example.ini info
```

You can avoid typing `-c/--config` option by setting the file name in the `MISHMASH_CONFIG` environment variable.

```
$ export MISHMASH_CONFIG=/etc/mishmash.ini
```

None of the options for controlling configuration are mutually exclusive, complex setups can be made by combining them. The order of precedence is show below:

```
Default <-- -c/--config <-- MISHMASH_CONFIG <-- -D/--database
```

Items to the left are lower precedence and the direction arrows (`<--`) show the order in which the options are merged. For example, local machine changes (local.ini) could be merged with the global site configuration (site.ini) and the PostgreSQL server at dbserver.example.com is used regardless then the other files set.

```
$ MISHMASH_CONFIG=local.ini mishmash -c site.ini -D postgresql://dbserver.example.
↪com:5432/music
```

3.2 Databases

The first requirement is deciding a database for MishMash to use. One of the great things about SQLAlchemy is its support for a multitude of databases, feel free to try whichever you would like but that the only back-ends that are currently tested/supported are:

```
* Postgresql
* SQLite; limited testing.
```

The default value uses a SQLite database called 'mishmash.db' in the user's home directory.:

```
sqlite:///${HOME}/mishmash.db
```

The URL in this example specifies the type of database (i.e. SQLite) and the filename of the DB file. The following sections provide more URL examples for Postgresql (where authentication credentials are required) and SQLite but see the full documentation for SQLAlchemy database URLs for a complete reference.

3.2.1 Postgresql

The pattern for Postgresql URLs is:

```
postgresql://user:passwd@host:port/db_name
```

`user` and `passwd` are the login credentials for accessing the database, while `host` and `port` (the default is 5432) determine where to connect. Lastly, the specific name of the database that contains the MishMash data is given by `db_name`. A specific example:

```
postgresql://mishmash:mishmash@localhost/mishmash_test
```

Setup of initial database and roles::

```
$ createuser --createdb mishmash
$ createdb -E utf8 -U mishmash mishmash
```

3.2.2 SQLite

The pattern for SQLite URLs is:

```
sqlite://filename
```

The slashes can be a little odd, so some examples:

```
sqlite:///relative/path/to/filename.db
sqlite:////absolute/path/to/filename.db
sqlite:///:memory:
```

The last example specifies an in-memory database that only exists as long as the application using it.

3.3 mishmash info

The `info` command displays details about the current settings and database. TODO

3.4 mishmash sync

The `sync` command imports music metadata into the database. TODO

3.5 mishmash web

The `web` command runs the web interface. TODO

3.6 mishmash merge-artists

TODO

3.7 mishmash split-artists

Since MishMash tries not to make assumption about directory structure there may be times when multiple artists with the same name are merged. The *split-artists* command can use to fix this.:

```
$ mishmash split-artists -L Music "The Giraffes"
```

The city of origin is used to distinguish between each of the artists and then albums are assigned to each.:

```
4 albums by The Giraffes:
   2005       The Giraffes
   2005       Haunted Heaven EP
   2008       Prime Motivator
   2000       The Days Are Filled With Years

Enter the number of distinct artists: 2

The Giraffes #1
   City: Brooklyn
   State: NY
   Country: US

The Giraffes #2
   City: Seattle
   State: WA
   Country: USA

Assign albums to the correct artist.
Enter 1 for The Giraffes from Brooklyn, NY, USA
Enter 2 for The Giraffes from Seattle, WA, USA

The Giraffes (Giraffes, The (Brooklyn)/2005 - The Giraffes): 1
Haunted Heaven EP (Giraffes, The (Brooklyn)/2005 - Haunted Heaven EP): 1
Prime Motivator (Giraffes, The (Brooklyn)/2008 - Prime Motivator): 1
The Days Are Filled With Years (Giraffes, The (Seattle)/2000 - The Days Are Filled␣
↪With Years): 2
```

CHAPTER 4

mishmash

4.1 mishmash package

4.1.1 Subpackages

mishmash.commands package

Submodules

mishmash.commands.command module

mishmash.commands.info module

class mishmash.commands.info.**DisplayList**
Bases: object

add(*key*, *val*)

clear()

print(*_format*, *clear=False*, ***kwargs*)

class mishmash.commands.info.**Info**(*subparsers=None*, ***kwargs*)
Bases: *mishmash.core.Command*

Construct a command. Any *kwargs* are added to the class object using `setattr`. All commands have an ArgumentParser, either constructed here or when `subparsers` is given a new parser is created using its `add_parser` method.

HELP = 'Show information about the database and configuration.'

NAME = 'info'

lib_query(*OrgType*, *lib*)

mishmash.commands.mgmt module

class mishmash.commands.mgmt.**Images**(*subparsers=None*, ***kwargs*)
Bases: *mishmash.core.Command*

Construct a command. Any *kwargs* are added to the class object using `setattr`. All commands have an ArgumentParser, either constructed here or when `subparsers` is given a new parser is created using its `add_parser` method.

HELP = 'Image mgmt.'

NAME = 'image'

class mishmash.commands.mgmt.**MergeArtists**(*subparsers=None*, ***kwargs*)
Bases: *mishmash.core.Command*

Construct a command. Any *kwargs* are added to the class object using `setattr`. All commands have an ArgumentParser, either constructed here or when `subparsers` is given a new parser is created using its `add_parser` method.

HELP = 'Merge two or more artists into a single artist.'

NAME = 'merge-artists'

class mishmash.commands.mgmt.**SplitArtists**(*subparsers=None*, ***kwargs*)
Bases: *mishmash.core.Command*

Construct a command. Any *kwargs* are added to the class object using `setattr`. All commands have an ArgumentParser, either constructed here or when `subparsers` is given a new parser is created using its `add_parser` method.

HELP = 'Split a single artist name into N distinct artists.'

NAME = 'split-artists'

mishmash.commands.sync module

mishmash.commands.web module

class mishmash.commands.web.**Web**(*subparsers=None*, ***kwargs*)
Bases: *mishmash.core.Command*

Construct a command. Any *kwargs* are added to the class object using `setattr`. All commands have an ArgumentParser, either constructed here or when `subparsers` is given a new parser is created using its `add_parser` method.

HELP = 'MishMash web interface.'

NAME = 'web'

Module contents

mishmash.orm module

mishmash.web package

Submodules

mishmash.web.layouts module

class mishmash.web.layouts.**AppLayout**(*context*, *request*)
Bases: object

add_heading(*name*, **args*, ***kw*)

page_title

mishmash.web.models module

mishmash.web.models.**DBSession = None**
The type for making sessions.

mishmash.web.panels module

mishmash.web.panels.**album_cover**(*context*, *request*, *album*, *size=None*, *link=False*)

mishmash.web.panels.**artist_image**(*context*, *request*, *artist*, *scale_percent=None*, *link=False*)

mishmash.web.panels.**footer**(*context*, *request*)

mishmash.web.panels.**navbar**(*context*, *request*)

mishmash.web.views module

class mishmash.web.views.**ResponseDict**(**args*, ***kwargs*)
Bases: dict

mishmash.web.views.**albumView**(*request*)

mishmash.web.views.**allAlbumsView**(*request*)

mishmash.web.views.**allArtistsView**(*request*)

mishmash.web.views.**artistView**(*request*)

mishmash.web.views.**artist_images**(*request*)

mishmash.web.views.**covers**(*request*)

mishmash.web.views.**home_view**(*request*)

mishmash.web.views.**newMusicView**(*request*)

mishmash.web.views.**searchView**(*request*)

Module contents

mishmash.web.**main**(*global_config*, ***main_settings*)

4.1.2 Submodules

4.1.3 mishmash.config module

class mishmash.config.**Config**(*filename*, ***kwargs*)
Bases: nicfit.config.Config

db_url

music_libs

class mishmash.config.**MusicLibrary**(*name*, *paths=None*, *excludes=None*, *sync=True*)
Bases: object

static fromConfig(*config*)

4.1.4 mishmash.console module

mishmash.console.**promptArtist**(*text*, *name=None*, *default_name=None*, *default_city=None*, *default_state=None*, *default_country=None*, *artist=None*)

mishmash.console.**selectArtist**(*heading*, *choices=None*, *multiselect=False*, *allow_create=True*)

4.1.5 mishmash.core module

class mishmash.core.**Command**(*subparsers=None*, ***kwargs*)
Bases: nicfit.command.Command

Base class for MishMash commands.

Construct a command. Any *kwargs* are added to the class object using `setattr`. All commands have an ArgumentParser, either constructed here or when `subparsers` is given a new parser is created using its `add_parser` method.

config = None

db_conn = None

db_engine = None

db_session = None

run(*args*)

exception mishmash.core.**CommandError**(*msg*, *exit_status=1*)
Bases: Exception

Base error type for nicfit.command.Command errors.

4.1.6 mishmash.database module

class mishmash.database.**DatabaseInfo**(*engine*, *SessionMaker*, *connection*)
Bases: tuple

Create new instance of DatabaseInfo(engine, SessionMaker, connection)

SessionMaker
Alias for field number 1

connection
Alias for field number 2

engine
Alias for field number 0

mishmash.database.**dropAll**(*url*)

mishmash.database.**getTag**(*t*, *session*, *lid*, *add=False*)

mishmash.database.**init**(*db_url*, *engine_args=None*, *session_args=None*, *trans_mgr=None*, *scoped=False*)

mishmash.database.**search**(*session*, *query*)
Naive search of the database for *query*.

Returns A dict with keys 'artists', 'albums', and 'tracks'. Each containing a list of the respective ORM type.

4.1.7 mishmash.util module

mishmash.util.**addLibraryArguments**(*cli: argparse.ArgumentParser*, *nargs*)
Add library options (-L/–library) with specific *nargs*.

mishmash.util.**commonDirectoryPrefix**(**args*)

mishmash.util.**mostCommonItem**(*lst*)
Choose the most common item from the list, or the first item if all items are unique.

mishmash.util.**normalizeCountry**(*country_str*, *target='iso3c'*)
Return a normalized name/code for country in *country_str*. The input can be a code or name, the *target* determines output value. 3 character ISO code is the default (iso3c), or 'iso2c'; otherwise then formal name is returned.

Raises `ValueError` if the country is unrecognized.

mishmash.util.**safeDbUrl**(*db_url*)
Obfuscates password from a database URL.

mishmash.util.**sortByDate**(*things*, *prefer_recording_date=False*)

mishmash.util.**splitNameByPrefix**(*s*)

4.1.8 Module contents

mishmash.**getLogger**(*name=None*)

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/nicfit/MishMash/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

5.1.4 Write Documentation

MishMash could always use more documentation, whether as part of the official MishMash docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/nicfit/MishMash/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up MishMash for local development.

1. Fork the *MishMash* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/MishMash.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv mishmash
$ cd mishmash/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ make lint
$ make test
$ make test-all    # Optional, requires multiple versions of Python
```

To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub.:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, and 3.3, 3.4, 3.5, and for PyPy. Check https://travis-ci.org/nicfit/MishMash/pulls and make sure that the tests pass for all supported Python versions.

CHAPTER 6

Authors

- Travis Shirk <travis@pobox.com>
- Chris Newton <redshodan@users.noreply.github.com>
- Ben Schumacher <me@benschumacher.com>

CHAPTER 7

Release History

7.1 v0.3.3 (2020-02-29) : Telltale

7.1.1 Changes

- Replaced *countrycode* package *iso3166*
- Gunicorn upgrade.

7.2 v0.3.2 (2020-02-26) : Gareth Brown Says

- Pin gunicorn<20.0.0.
- Skip on sync errors.
- Remove User orm.

7.3 v0.3.1 (2020-02-22)

7.3.1 Fix

- Replacement of ZopeTransactionExtension.

7.4 v0.3 (2020-02-22) : Drunken Baby

- Initial release

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index

N

P

R

S

W