MyAnimeList Scraper

Release 0.3.0

Mar 14, 2018

Contents

CHAPTER 1

Overview

docs	
tests	
package	

MyAnimeList Web Scraper (mal-scraper) is a Python library for gathering a basic set of data about anime.

It can gather information about users from MAL including what anime they have watched and their ratings. It can discover users on MAL, and retrieve some very basic information about each anime. This information can be used to analyse data.

1.1 Installation & Usage

```
pip install mal-scraper
```

Please use the online documentation for to get started.

The library follows Semantic Versioning.

1.2 Development

Please see the Contributing documentation page for full details, and especially look at the tips section.

After cloning, and creating a virtualenv, install the development dependencies:

```
pip install -e .[develop]
```

To run the all tests, skipping the python interpreters you don't have:

```
tox --skip-missing-interpreters
```

Project Notes:

- Tests will always mock requests to the internet. You can set the environment variable `LIVE_RESPONSES=1` to properly test web scraping.
- You can look at coverage results inside `htmlcov/index.html`.

Note, to combine the coverage data from all the tox environments run:

Windows	`set PYTEST_ADDOPTS=--cov-append` `tox`
Other	`PYTEST_ADDOPTS=--cov-append tox`

CHAPTER 2

Usage/Examples

To use MyAnimeList Scraper in a project, for example to retrieve anime metadata:

```
import mal_scraper
import mycode

next_id_ref = mycode.last_id_ref() + 1

try:
    meta, data = mal_scraper.get_anime(next_id_ref)
except requests.exceptions.HTTPError as err:
    code = err.response.status_code
    if code == 404:
        print('Anime #%d does not exist (404)', next_id_ref)
        mycode.ignore_id_ref(next_id_ref)
    else:
        # Retry on network/server/request errors
        print('Anime #%d HTTP error (%d)', next_id_ref, code)
        mycode.mark_for_retry(next_id_ref)
else:
    print('Adding Anime #%d', meta['id_ref'])
    mycode.add_anime(
        id_ref=meta['id_ref'],
        anime_information_dated_at=meta['when'],
        name=data['name'],
        episodes=data['episodes'],
        # Ignore other data
    )
```

CHAPTER 3

Reference

The public API is available directly by importing from *mal_scraper*, for example:

```
import mal_scraper

mal_scraper.get_anime  # Core API
mal_scraper.AiringStatus.ongoing  # Constants/Enumerations
mal_scraper.ParseError  # Exceptions
```

3.1 Core API

The library supports retrieving Anime, User Profile/Stats, and User-Anime Info.

Anime and Users are identified by id_ref (int), and user_id (str) respectively, so while you can enumerate through Anime, you must 'discover' Users.

mal_scraper.**discover_users**(*requester=<module 'requests' from '/home/docs/checkouts/readthedocs.org/user_builds/mal-scraper/envs/latest/lib/python3.5/site-packages/requests/__init__.py'>*, *use_cache=True*, *use_web=None*)

Return a set of user_ids usable by other user related library calls.

By default we will attempt to return any in our cache - clearing the cache in the process. If there are no users in the cache, we will attempt to find some on MAL but these will be biased towards recently active users.

The cache is built up by discovering users from all of the other web-pages retrieved from other API calls as you make those calls.

Parameters

- **requester** (*requests-like, optional*) – HTTP request maker. This allows us to control/limit/mock requests.
- **use_cache** (*bool, optional*) – Ignore the cache that we have built up over time? True (default): Pretend the cache is empty (and do not clear it). False: Get and clear the cache.

- **use_web** (*bool, optional*) – Control whether to fall back to scraping. None (default) to make a network call only if the cache is empty. False to never make a network call. True to always make a network call.

Returns A set of user_ids which are strings.

Raises *Network and Request Errors* – See Requests library.

Examples

Get user_ids discovered from earlier uses of the library:

```
animes = mal_scraper.get_anime()
users_probably_from_cache = mal_scraper.discover_users()
```

Get user_ids if there are any in the cache, but don't bother to make a network call just to find some:

```
users_from_cache = mal_scraper.discover_users(use_web=False)
```

Discover some users from the web, ignoring the cache:

```
users_from_web = mal_scraper.discover_users(use_cache=False)
```

mal_scraper.**get_anime**(*id_ref=1*, *requester=<module 'requests' from '/home/docs/checkouts/readthedocs.org/user_builds/mal-scraper/envs/latest/lib/python3.5/site-packages/requests/__init__.py'>*)

Return the information for a particular show.

You can simply enumerate through id_refs.

This will raise exceptions unless we properly and fully retrieve and process the web-page.

TODO: Genres https://myanimelist.net/info.php?go=genre # Broadcast? Producers? Licensors? Studios? Source? Duration?

Parameters

- **id_ref** (*int, optional*) – Internal show identifier.
- **requester** (*requests-like, optional*) – HTTP request maker. This allows us to control/limit/mock requests.

Returns

Retrieved – with the attributes *meta* and *data*.

data:

```
{
    'name': str,
    'name_english': str,
    'format': mal_scraper.Format,
    'episodes': int, or None when MAL does not know,
    'airing_status': mal_scraper.AiringStatus,
    'airing_started': date, or None when MAL does not know,
    'airing_finished': date, or None when MAL does not know,
    'airing_premiere': tuple(Year (int), Season (mal_scraper.Season))
        or None (for films, OVAs, specials, ONAs, music, or
        if MAL does not know),
    'mal_age_rating': mal_scraper.AgeRating,
    'mal_score': float, or None when not yet aired/MAL does not know,
```

```
    'mal_scored_by': int (number of people),
    'mal_rank': int, or None when not yet aired/some R rated anime,
    'mal_popularity': int,
    'mal_members': int,
    'mal_favourites': int,
}
```

See also *Format*, *AiringStatus*, *Season*.

Raises

- *Network and Request Errors* – See Requests library.
- *ParseError* – Upon processing the web-page including anything that does not meet expectations.

Examples

Retrieve the first anime and get the next anime to retrieve:

```
next_anime = 1

try:
    meta, data = mal_scraper.get_anime(next_anime)
except mal_scraper.ParseError as err:
    logger.error('Investigate page %s with error %d', err.url, err.code)
except NetworkandRequestErrors:  # Pseudo-code (TODO: These docs)
    pass  # Retry?
else:
    mycode.save_data(data, when=meta['when'])

next_anime = meta['id_ref'] + 1
```

mal_scraper.**get_user_anime_list**(*user_id*, *requester=<module 'requests' from '/home/docs/checkouts/readthedocs.org/user_builds/mal-scraper/envs/latest/lib/python3.5/site-packages/requests/__init__.py'>*)

Return the anime listed by the user on their profile.

This will make multiple network requests (possibly > 10).

TODO: Return Meta

Parameters

- **user_id** (*str*) – The user identifier (i.e. the username).
- **requester** (*requests-like, optional*) – HTTP request maker. This allows us to control/limit/mock requests.

Returns

A list of anime-info where each anime-info is the following dict:

```
{
    'name': (string) name of the anime,
    'id_ref': (id_ref) can be used with mal_scraper.get_anime,
    'consumption_status': (mal_scraper.ConsumptionStatus),
    'is_rewatch': (bool),
    'score': (int) 0-10,
```

```
    'progress': (int) 0+ number of episodes watched,
    'tags': (set of strings) user tags,

    The following tags have been removed for now:
    'start_date': (date, or None) may be missing,
    'finish_date': (date, or None) may be missing or not finished,
}
```

See also *ConsumptionStatus*.

Raises

- *Network and Request Errors* – See Requests library.
- *RequestError* – `RequestError.Code.forbidden` if the user's info is private, or `RequestError.Code.does_not_exist` if the user_id is invalid. See *RequestError.Code*.
- *ParseError* – Upon processing the web-page including anything that does not meet expectations.

`mal_scraper.`**`get_user_stats`**(*user_id*, *requester=<module 'requests' from '/home/docs/checkouts/readthedocs.org/user_builds/mal-scraper/envs/latest/lib/python3.5/site-packages/requests/__init__.py'>*)

Return statistics about a particular user.

TODO: Return Gender Male/Female # TODO: Return Birthday "Nov", "Jan 27, 1997" # TODO: Return Location "England" # e.g. https://myanimelist.net/profile/Sakana-san

Parameters

- **user_id** (*string*) – The username identifier of the MAL user.
- **requester** (*requests-like, optional*) – HTTP request maker. This allows us to control/limit/mock requests.

Returns

Retrieved – with the attributes *meta* and *data*.

data:

```
{
    'name': (str) user_id/username,
    'last_online': (datetime),
    'joined': (datetime),
    'num_anime_watching': (int),
    'num_anime_completed': (int),
    'num_anime_on_hold': (int),
    'num_anime_dropped': (int),
    'num_anime_plan_to_watch': (int),
}
```

Raises

- *Network and Request Errors* – See Requests library.
- *RequestError* – `RequestError.Code.does_not_exist` if the user_id is invalid (i.e. the username does not exist). See *RequestError.Code*.
- *ParseError* – Upon processing the web-page including anything that does not meet expectations.

3.2 Constants

All constants/enumerations are available directly from *mal_scraper.x*

class mal_scraper.consts.**AgeRating**
The age rating of a media item.

MAL Ratings are dubious.

None == Unknown.

Reference: https://myanimelist.net/forum/?topicid=16816

mal_g = 'ALL'

mal_none = 'NONE'

mal_pg = 'CHILDREN'

mal_r1 = 'RESTRICTEDONE'

mal_r2 = 'RESTRICTEDTWO'

mal_r3 = 'RESTRICTEDTHREE'

mal_t = 'TEEN'

classmethod mal_to_enum(*text*)
Return the enum from the MAL string, or None.

class mal_scraper.consts.**AiringStatus**
The airing status of a media item.

finished = 'FINISHED'

classmethod mal_to_enum(*text*)

ongoing = 'ONGOING'

pre_air = 'PREAIR'

class mal_scraper.consts.**ConsumptionStatus**
A person's status on a media item, e.g. are they currently watching it?

backlog = 'BACKLOG'

completed = 'COMPLETED'

consuming = 'CONSUMING'

dropped = 'DROPPED'

classmethod mal_code_to_enum(*code*)
Return the enum from the MAL code, or None.

on_hold = 'ONHOLD'

class mal_scraper.consts.**Format**
The media format of a media item.

film = 'FILM'

classmethod mal_to_enum(*text*)
Return the enum from the MAL string, or None.

movie = 'FILM'

music = 'MUSIC'

ona = 'ONA'

ova = 'OVA'

special = 'SPECIAL'

tv = 'TV'

unknown = 'UNKNOWN'

class mal_scraper.consts.**Retrieved**(*meta*, *data*)
When successfully retrieving from a web-page

meta
A dict of metadata:

```
{
    'id_ref': (object) ID of the media depending on the context,
    'when': (datetime) Our best guess on the date of this information,
}
```

data
A dict of data varying on the media.

data
Alias for field number 1

meta
Alias for field number 0

class mal_scraper.consts.**Season**
The season in a year ordered as Winter, Spring, Summer, Autumn.

autumn = 'AUTUMN'

fall = 'AUTUMN'

classmethod mal_to_enum(*text*)
Return the enum from the MAL string, or None.

spring = 'SPRING'

summer = 'SUMMER'

winter = 'WINTER'

3.3 Exceptions

All exceptions are available directly from *mal_scraper.x*

exception mal_scraper.exceptions.**MalScraperError**
Parent to all exceptions raised by this library.

exception mal_scraper.exceptions.**ParseError**(*message*, *tag=None*)
A component of the HTML could not be parsed/processed.

The tag is the "component" under consideration to help determine where the error comes from.

Parameters

- **message** (*str*) – Human readable string describing the problem.
- **tag** (*str, optional*) – Which part of the page does this pertain to.

Variables

- **message** (*str*) – Human readable string describing the problem.
- **tag** (*str*) – Which part of the page does this pertain to.

exception mal_scraper.exceptions.**RequestError**(*code*, *message*)

An error making the request.

Parameters

- **code** (*RequestError.Code*) – Error code
- **message** (*str*) – Human readable string describing the problem.

Variables

- **code** (RequestError.Code) – Error code
- **message** (*str*) – Human readable string describing the problem.

class Code

An enumeration.

does_not_exist = 'NOEXIST'

forbidden = 'FORBIDDEN'

CHAPTER 4

Contributing/Development

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

4.1 Bug reports

When 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.

4.2 Documentation improvements

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

4.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/QasimK/mal-scraper/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 code contributions are welcome :)

4.4 Development

We follow (and our tests check):

- editorconfig
- isort

To set up *mal-scraper* for local development:

1. Fork mal-scraper (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/mal-scraper.git
```

3. Ensure Pipenv is installed on your computer and install the development packages:

```
pipenv install --dev
```

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, run all the checks, doc builder and spell checker with tox one command:

```
pipenv run tox
```

   The newly built HTML docs can be found within the dist folder in the repo.

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.

4.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

4.4.2 Tips

To run the test-suite quickly:

```
pytest
```

To run a subset of tests:

```
tox -e envname -- py.test -k test_myfeature
```

To skip Python environments that you do not have installed:

```
tox --skip-missing-interpreters
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

4.5 PyPI Submission

1. Bump Version `bumpversion minor`
2. Upload to Pypi `python setup.py sdist bdist_wheel upload -r pypi`

CHAPTER 5

Authors

- Qasim K - https://github.com/qasimk/mal-scraper

CHAPTER 6

Changelog

6.1 0.3.0 (2017-05-02)

- Fix various issues on anime pages
- Rename *retrieve_anime* to *get_anime* for consistency (backwards-incompatible)

6.2 0.2.1 (2017-05-01)

- Add Season as an Enum rather than a simple string (backwards-incompatible)
- Fix failing tests due to version number

6.3 0.2.0 (2017-05-01)

- Alter anime retrieval API to use exceptions (backwards-incompatible)
- Improve documentation (mainly around the anime API)

6.4 0.1.0 (2016-05-15)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index

S

T

U

W