python-mal Documentation

Release 0.1

Charles Guo

January 25, 2016

python-mal is a Python library for interacting programmatically with resources on MyAnimeList.

Confusingly enough, the package is called `python-mal` while the module itself is named `myanimelist`. The docs will refer to `python-mal` whenever we talk about the package, e.g. installing it, but for anything else, e.g. actual code, we'll refer to `myanimelist`.

All features are tested under Python 2.7.

CHAPTER 1

Getting Started

If you've never used `myanimelist` before, you should read the Getting Started with myanimelist guide to get familiar with `myanimelist`.

CHAPTER 2

Supported MAL Resources

- Users
- Anime
- Anime Lists
- Manga
- Manga Lists
- Characters

CHAPTER 3

Additional Resources

- python-mal repository

3.1 Getting Started with myanimelist

This tutorial will walk you through installing `myanimelist`, as well how to use it. It assumes you are familiar with Python; if you're not, please look up one of the many fantastic Python tutorials out there before continuing.

3.1.1 Installing python-mal

You can use `pip` to install the latest released version of `python-mal`:

```
pip install python-mal
```

If you want to install `python-mal` from source:

```
git clone git://github.com/shaldengeki/python-mal.git
cd python-mal
python setup.py install
```

3.1.2 Connecting to MAL

`myanimelist` provides a `Session` class that manages requests to MAL. To do anything, you're first going to need to instantiate one:

```
>>> import myanimelist.session
>>> session = myanimelist.session.Session()
```

If you have a username and password that you want to log into MAL with, you can provide the `Session` object with those:

```
>>> import myanimelist.session
>>> session = myanimelist.session.Session(username="mal_username", password="mal_password")
>>> session.login()
```

Providing credentials to MAL isn't actually required for most tasks, so feel free to forego the login process if you don't need it.

3.1.3 Interacting with MAL

Once you have a `Session` connected to MAL, there are methods on that object that will return resources on MAL, like `anime` or `manga`. You're typically going to want to fetch all your resources through these convenience methods. The following code demonstrates how to fetch an anime and look up all characters for it:

```
>>> import myanimelist.session
>>> session = myanimelist.session.Session()
# Return an anime object corresponding to an ID of 1. IDs must be natural numbers.
>>> bebop = session.anime(1)
>>> print bebop.title
Cowboy Bebop
# Print each character's name and their role.
>>> for character in bebop.characters:
...   print character.name, '---', bebop.characters[character]['role']
Spike Spiegel --- Main
Faye Valentine --- Main
Jet Black --- Main
Ein --- Supporting
Rocco Bonnaro --- Supporting
Mad Pierrot --- Supporting
Dr. Londez --- Supporting
...
```

Users on MAL are slightly different; their primary ID is their username, instead of an integral ID. So, say you wanted to look up some user's recommendations. The following code would be one way to do it:

```
>>> import myanimelist.session
>>> session = myanimelist.session.Session()
# Return an user object corresponding to the given username. Usernames _must_ be unicode!
>>> shal = session.user(u'shaldengeki')
>>> print shal.website
llanim.us
# Print each recommendation.
>>> for anime in shal.recommendations:
...   print anime.title, '---', shal.recommendations[anime]['anime'].title
...   print "====================================="
...   print shal.recommendations[anime]['text']
Kanon (2006) --- Clannad: After Story
=====================================
School life, slice of life anime based on a visual novel. Male protagonist gets to know the backgrou
```

Anime and manga lists are similar, being primarily-identified through usernames instead of user IDs.

Each resource has a slightly-different set of methods. You'll want to refer to the other guides and API references in this documentation to see what's available.

3.2 myanimelist

3.2.1 myanimelist package

Submodules

myanimelist.anime module

class myanimelist.anime.**Anime**(*session*, *anime_id*)
Bases: *myanimelist.media.Media*

Primary interface to anime resources on MAL.

aired
A tuple(2) containing up to two datetime.date objects representing the start and end dates of this anime's airing.

Potential configurations:

None – Completely-unknown airing dates.

(datetime.date, None) – Anime start date is known, end date is unknown.

(datetime.date, datetime.date) – Anime start and end dates are known.

duration
The duration of an episode of this anime as a datetime.timedelta.

episodes
The number of episodes in this anime. If undetermined, is None, otherwise > 0.

parse_characters(*character_page*)
Parses the DOM and returns anime character attributes in the sidebar.

Parameters **character_page** (bs4.BeautifulSoup) – MAL anime character page's DOM

Return type dict

Returns anime character attributes

Raises *InvalidAnimeError*, *MalformedAnimePageError*

parse_sidebar(*anime_page*)
Parses the DOM and returns anime attributes in the sidebar.

Parameters **anime_page** (bs4.BeautifulSoup) – MAL anime page's DOM

Return type dict

Returns anime attributes

Raises *InvalidAnimeError*, *MalformedAnimePageError*

producers
A list of myanimelist.producer.Producer objects involved in this anime.

rating
The MPAA rating given to this anime.

staff
A staff dict with *myanimelist.person.Person* objects of the staff members as keys, and lists containing the various duties performed by staff members as values.

voice_actors
A voice actors dict with *myanimelist.person.Person* objects of the voice actors as keys, and dicts containing info about the roles played, e.g. {'role': 'Main', 'character': myanimelist.character.Character(1)} as values.

exception myanimelist.anime.**InvalidAnimeError**(*id*, *message=None*)
Bases: *myanimelist.media.InvalidMediaError*

Indicates that the anime requested does not exist on MAL.

exception myanimelist.anime.**MalformedAnimePageError**(*id*, *html*, *message=None*)
Bases: *myanimelist.media.MalformedMediaPageError*

Indicates that an anime-related page on MAL has irreparably broken markup in some way.

myanimelist.anime_list module

class myanimelist.anime_list.**AnimeList**(*session*, *user_name*)
Bases: *myanimelist.media_list.MediaList*

parse_entry(*soup*)

parse_entry_media_attributes(*soup*)

parse_section_columns(*columns*)

type

verb

myanimelist.base module

class myanimelist.base.**Base**(*session*)
Bases: object

Abstract base class for MAL resources. Provides autoloading, auto-setting functionality for other MAL objects.

load()
A callback to run before any @loadable attributes are returned.

set(*attr_dict*)
Sets attributes of this user object.

Parameters **attr_dict** (*dict*) – Parameters to set, with attribute keys.

Return type *Base*

Returns The current object.

exception myanimelist.base.**Error**(*message=None*)
Bases: exceptions.Exception

Base exception class that takes a message to display upon raising.

exception myanimelist.base.**InvalidBaseError**(*id*, *message=None*)
Bases: *myanimelist.base.Error*

Indicates that the particular resource instance requested does not exist on MAL.

exception myanimelist.base.**MalformedPageError**(*id*, *html*, *message=None*)
Bases: *myanimelist.base.Error*

Indicates that a page on MAL has broken markup in some way.

myanimelist.base.**loadable**(*func_name*)
Decorator for getters that require a load() upon first access.

Parameters **func_name** (*function*) – class method that requires that load() be called if the class's _attribute value is None

Return type function

Returns the decorated class method.

myanimelist.character module

class myanimelist.character.**Character**(*session*, *character_id*)
Bases: *myanimelist.base.Base*

Primary interface to character resources on MAL.

animeography
Anime appearance dict for this character, with *myanimelist.anime.Anime* objects as keys and the type of role as values, e.g. 'Main'

clubs
List of clubs relevant to this character.

description
Character's description.

favorites
List of users who have favourited this character.

full_name
Character's full name.

load()
Fetches the MAL character page and sets the current character's attributes.

Return type *Character*

Returns Current character object.

load_clubs()
Fetches the MAL character clubs page and sets the current character's clubs attributes.

Return type *Character*

Returns Current character object.

load_favorites()
Fetches the MAL character favorites page and sets the current character's favorites attributes.

Return type *Character*

Returns Current character object.

load_pictures()
Fetches the MAL character pictures page and sets the current character's pictures attributes.

Return type *Character*

Returns Current character object.

mangaography
Manga appearance dict for this character, with *myanimelist.manga.Manga* objects as keys and the type of role as values, e.g. 'Main'

name
Character name.

name_jpn
Character's Japanese name.

num_favorites
Number of users who have favourited this character.

parse(*character_page*)
Parses the DOM and returns character attributes in the main-content area.

Parameters **character_page** (bs4.BeautifulSoup) – MAL character page's DOM

Return type dict

Returns Character attributes.

parse_clubs(*clubs_page*)
Parses the DOM and returns character clubs attributes.

Parameters **clubs_page** (bs4.BeautifulSoup) – MAL character clubs page's DOM

Return type dict

Returns character clubs attributes.

parse_favorites(*favorites_page*)
Parses the DOM and returns character favorites attributes.

Parameters **favorites_page** (bs4.BeautifulSoup) – MAL character favorites page's DOM

Return type dict

Returns Character favorites attributes.

parse_pictures(*picture_page*)
Parses the DOM and returns character pictures attributes.

Parameters **picture_page** (bs4.BeautifulSoup) – MAL character pictures page's DOM

Return type dict

Returns character pictures attributes.

parse_sidebar(*character_page*)
Parses the DOM and returns character attributes in the sidebar.

Parameters **character_page** (bs4.BeautifulSoup) – MAL character page's DOM

Return type dict

Returns Character attributes

Raises *InvalidCharacterError*, *MalformedCharacterPageError*

picture
URL of primary picture for this character.

pictures
List of picture URLs for this character.

voice_actors
Voice actor dict for this character, with *myanimelist.person.Person* objects as keys and the language as values.

exception myanimelist.character.**InvalidCharacterError**(*id*, *message=None*)
Bases: *myanimelist.base.InvalidBaseError*

Indicates that the character requested does not exist on MAL.

exception myanimelist.character.**MalformedCharacterPageError**(*id*, *html*, *message=None*)
Bases: *myanimelist.base.MalformedPageError*

Indicates that a character-related page on MAL has irreparably broken markup in some way.

myanimelist.club module

class myanimelist.club.**Club**(*session*, *club_id*)
Bases: *myanimelist.base.Base*

load()

name

num_members

exception myanimelist.club.**InvalidClubError**(*id*, *message=None*)
Bases: *myanimelist.base.InvalidBaseError*

exception myanimelist.club.**MalformedClubPageError**(*id*, *html*, *message=None*)
Bases: *myanimelist.base.MalformedPageError*

myanimelist.genre module

class myanimelist.genre.**Genre**(*session*, *genre_id*)
Bases: *myanimelist.base.Base*

load()

name

exception myanimelist.genre.**InvalidGenreError**(*id*, *message=None*)
Bases: *myanimelist.base.InvalidBaseError*

exception myanimelist.genre.**MalformedGenrePageError**(*id*, *html*, *message=None*)
Bases: *myanimelist.base.MalformedPageError*

myanimelist.manga module

exception myanimelist.manga.**InvalidMangaError**(*id*, *message=None*)
Bases: *myanimelist.media.InvalidMediaError*

Indicates that the manga requested does not exist on MAL.

exception myanimelist.manga.**MalformedMangaPageError**(*id*, *html*, *message=None*)
Bases: *myanimelist.media.MalformedMediaPageError*

Indicates that a manga-related page on MAL has irreparably broken markup in some way.

class myanimelist.manga.**Manga**(*session*, *manga_id*)
Bases: *myanimelist.media.Media*

Primary interface to manga resources on MAL.

authors
An author dict with *myanimelist.person.Person* objects of the authors as keys, and strings describing the duties of these authors as values.

chapters
The number of chapters in this manga.

parse_sidebar(*manga_page*)
Parses the DOM and returns manga attributes in the sidebar.

Parameters **manga_page** (bs4.BeautifulSoup) – MAL manga page's DOM

Return type dict

Returns manga attributes

Raises *InvalidMangaError*, *MalformedMangaPageError*

published
A tuple(2) containing up to two datetime.date objects representing the start and end dates of this manga's publishing.

Potential configurations:

None – Completely-unknown publishing dates.

(datetime.date, None) – Manga start date is known, end date is unknown.

(datetime.date, datetime.date) – Manga start and end dates are known.

serialization
The *myanimelist.publication.Publication* involved in the first serialization of this manga.

volumes
The number of volumes in this manga.

myanimelist.manga_list module

class myanimelist.manga_list.**MangaList**(*session*, *user_name*)
Bases: *myanimelist.media_list.MediaList*

parse_entry(*soup*)

parse_entry_media_attributes(*soup*)

type

verb

myanimelist.media module

exception myanimelist.media.**InvalidMediaError**(*id*, *message=None*)
Bases: *myanimelist.base.InvalidBaseError*

Indicates that the media requested does not exist on MAL.

exception myanimelist.media.**MalformedMediaPageError**(*id*, *html*, *message=None*)
Bases: *myanimelist.base.MalformedPageError*

Indicates that a media-related page on MAL has broken markup in some way.

class myanimelist.media.**Media**(*session*, *id*)
Bases: *myanimelist.base.Base*

Abstract base class for all media resources on MAL.

To subclass, create a class that inherits from Media, implementing status_terms and consuming_verb at the bare minimum.

alternative_titles
Alternative titles dict, with types of titles, e.g. 'Japanese', 'English', or 'Synonyms' as keys, and lists of said alternative titles as values.

characters
Character dict, with *myanimelist.character.Character* objects as keys, and a dict with attributes of this role, e.g. 'role': 'Main' as values.

favorites
Number of users who favourited this media.

genres
A list of *myanimelist.genre.Genre* objects associated with this media.

load()
Fetches the MAL media page and sets the current media's attributes.

Return type *Media*

Returns current media object.

load_characters()
Fetches the MAL media characters page and sets the current media's character attributes.

Return type *Media*

Returns current media object.

load_stats()
Fetches the MAL media statistics page and sets the current media's statistics attributes.

Return type *Media*

Returns current media object.

members
Number of members.

classmethod newest(*session*)
Fetches the latest media added to MAL.

Parameters **session** (*myanimelist.session.Session*) – A valid MAL session

Return type *Media*

Returns the newest media on MAL

Raises *MalformedMediaPageError*

parse(*media_page*)
Parses the DOM and returns media attributes in the main-content area.

Parameters **media_page** (bs4.BeautifulSoup) – MAL media page's DOM

Return type dict

Returns media attributes.

parse_characters(*character_page*)
Parses the DOM and returns media character attributes in the sidebar.

Parameters **character_page** (bs4.BeautifulSoup) – MAL character page's DOM

Return type dict

Returns character attributes.

parse_sidebar(*media_page*)
Parses the DOM and returns media attributes in the sidebar.

Parameters **media_page** (bs4.BeautifulSoup) – MAL media page's DOM

Return type dict

Returns media attributes.

Raises InvalidMediaError, MalformedMediaPageError

parse_stats(*media_page*)
Parses the DOM and returns media statistics attributes.

Parameters **media_page** (bs4.BeautifulSoup) – MAL media stats page's DOM

Return type dict

Returns media stats attributes.

picture
URL of media's primary pictures.

popular_tags
Tags dict with *myanimelist.tag.Tag* objects as keys, and the number of tags as values.

popularity
Popularity rank.

rank
Score rank.

related
Related media dict, with strings of relation types, e.g. 'Sequel' as keys, and lists containing instances of *Media* subclasses as values.

score
A tuple(2) containing an instance of decimal.Decimal storing the aggregate score, weighted or non-weighted, and an int storing the number of ratings

score_stats
Score statistics dict, with int scores from 1-10 as keys, and an int number of users as values.

status
Publication status, e.g. 'Finished Airing'

status_stats
Status statistics dict, with strings of statuses, e.g. 'on_hold' as keys, and an int number of users as values.

synopsis
Media synopsis.

title
Media's title.

type
Type of this media, e.g. 'TV' or 'Manga' or 'Movie'

myanimelist.media_list module

exception myanimelist.media_list.**InvalidMediaListError**(*id*, *message=None*)
Bases: *myanimelist.base.InvalidBaseError*

exception myanimelist.media_list.**MalformedMediaListPageError**(*id*, *html*, *message=None*)
Bases: *myanimelist.base.MalformedPageError*

class myanimelist.media_list.**MediaList**(*session*, *user_name*)
Bases: *myanimelist.base.Base*, _abcoll.Mapping

list

load()

parse(*xml*)

parse_entry(*soup*)

Given: soup: a bs4 element containing a row from the current media list

Return a tuple: (media object, dict of this row's parseable attributes)

parse_entry_media_attributes(*soup*)

Args: soup: a bs4 element containing a row from the current media list

Return a dict of attributes of the media the row is about.

parse_stats(*soup*)

Given: soup: a bs4 element containing the current media list's stats

Return a dict of this media list's stats.

section(*status*)

stats

type

user_status_terms

verb

myanimelist.myanimelist module

myanimelist.myanimelist.**patch_http_response_read**(*func*)

myanimelist.person module

exception myanimelist.person.**InvalidPersonError**(*id*, *message=None*)
Bases: *myanimelist.base.InvalidBaseError*

exception myanimelist.person.**MalformedPersonPageError**(*id*, *html*, *message=None*)
Bases: *myanimelist.base.MalformedPageError*

class myanimelist.person.**Person**(*session*, *person_id*)
Bases: *myanimelist.base.Base*

load()

name

myanimelist.publication module

exception myanimelist.publication.**InvalidPublicationError**(*id*, *message=None*)
Bases: *myanimelist.base.InvalidBaseError*

exception myanimelist.publication.**MalformedPublicationPageError**(*id*, *html*, *message=None*)
Bases: *myanimelist.base.MalformedPageError*

class myanimelist.publication.**Publication**(*session*, *publication_id*)
Bases: *myanimelist.base.Base*

load()

name

myanimelist.session module

class myanimelist.session.**Session**(*username=None*, *password=None*, *user_agent='iMAL-iOS'*)
Bases: object

Class to handle requests to MAL. Handles login, setting HTTP headers, etc.

anime(*anime_id*)
Creates an instance of myanimelist.Anime with the given ID.

Parameters **anime_id** (*int*) – The desired anime's ID.

Return type *myanimelist.anime.Anime*

Returns A new Anime instance with the given ID.

anime_list(*username*)
Creates an instance of myanimelist.AnimeList belonging to the given username.

Parameters **username** (*str*) – The username to whom the desired anime list belongs.

Return type *myanimelist.anime_list.AnimeList*

Returns A new AnimeList instance belonging to the given username.

character(*character_id*)
Creates an instance of myanimelist.Character with the given ID.

Parameters **character_id** (*int*) – The desired character's ID.

Return type *myanimelist.character.Character*

Returns A new Character instance with the given ID.

club(*club_id*)
Creates an instance of myanimelist.Club with the given ID.

Parameters **club_id** (*int*) – The desired club's ID.

Return type *myanimelist.club.Club*

Returns A new Club instance with the given ID.

genre(*genre_id*)
Creates an instance of myanimelist.Genre with the given ID.

Parameters **genre_id** (*int*) – The desired genre's ID.

Return type *myanimelist.genre.Genre*

Returns A new Genre instance with the given ID.

logged_in()
Checks the logged-in status of the current session. Expensive (requests a page), so use sparingly! Best practice is to try a request and catch an UnauthorizedError.

Return type bool

Returns Whether or not the current session is logged-in.

login()
Logs into MAL and sets cookies appropriately.

Return type *Session*

Returns The current session.

manga(*manga_id*)
Creates an instance of myanimelist.Manga with the given ID.

Parameters **manga_id** (*int*) – The desired manga's ID.

Return type *myanimelist.manga.Manga*

Returns A new Manga instance with the given ID.

manga_list(*username*)
Creates an instance of myanimelist.MangaList belonging to the given username.

Parameters **username** (*str*) – The username to whom the desired manga list belongs.

Return type *myanimelist.manga_list.MangaList*

Returns A new MangaList instance belonging to the given username.

person(*person_id*)
Creates an instance of myanimelist.Person with the given ID.

Parameters **person_id** (*int*) – The desired person's ID.

Return type *myanimelist.person.Person*

Returns A new Person instance with the given ID.

producer(*producer_id*)
Creates an instance of myanimelist.Producer with the given ID.

Parameters **producer_id** (*int*) – The desired producer's ID.

Return type myanimelist.producer.Producer

Returns A new Producer instance with the given ID.

publication(*publication_id*)
Creates an instance of myanimelist.Publication with the given ID.

Parameters **publication_id** (*int*) – The desired publication's ID.

Return type *myanimelist.publication.Publication*

Returns A new Publication instance with the given ID.

tag(*tag_id*)
Creates an instance of myanimelist.Tag with the given ID.

Parameters **tag_id** (*int*) – The desired tag's ID.

Return type *myanimelist.tag.Tag*

Returns A new Tag instance with the given ID.

user(*username*)
Creates an instance of myanimelist.User with the given username

Parameters **username** (*str*) – The desired user's username.

Return type *myanimelist.user.User*

Returns A new User instance with the given username.

exception myanimelist.session.**UnauthorizedError**(*session*, *url*, *result*)
Bases: *myanimelist.base.Error*

Indicates that the current session is unauthorized to make the given request.

myanimelist.tag module

exception myanimelist.tag.**InvalidTagError**(*id*, *message=None*)
Bases: *myanimelist.base.InvalidBaseError*

exception myanimelist.tag.**MalformedTagPageError**(*id*, *html*, *message=None*)
Bases: *myanimelist.base.MalformedPageError*

class myanimelist.tag.**Tag**(*session*, *name*)
Bases: *myanimelist.base.Base*

load()

myanimelist.user module

exception myanimelist.user.**InvalidUserError**(*id*, *message=None*)
Bases: *myanimelist.base.InvalidBaseError*

Indicates that the user requested does not exist on MAL.

exception myanimelist.user.**MalformedUserPageError**(*id*, *html*, *message=None*)
Bases: *myanimelist.base.MalformedPageError*

Indicates that a user-related page on MAL has irreparably broken markup in some way.

class myanimelist.user.**User**(*session*, *username*)
Bases: *myanimelist.base.Base*

Primary interface to user resources on MAL.

about
This user's self-bio.

access_rank
This user's access rank on MAL.

anime_list()
This user's anime list.

Return type *myanimelist.anime_list.AnimeList*

Returns The desired anime list.

anime_list_views
The number of times this user's anime list has been viewed.

anime_stats
A dict of this user's anime stats, with keys as strings, and values as numerics.

birthday
A `datetime.datetime` object marking this user's birthday.

clubs
A list of *myanimelist.club.Club* objects containing this user's club memberships.

favorite_anime
A list of *myanimelist.anime.Anime* objects containing this user's favorite anime.

favorite_characters
A dict with *myanimelist.character.Character* objects as keys and *myanimelist.media.Media* as values.

favorite_manga
A list of *myanimelist.manga.Manga* objects containing this user's favorite manga.

favorite_people
A list of *myanimelist.person.Person* objects containing this user's favorite people.

static find_username_from_user_id(*session*, *user_id*)
Look up a MAL username's user ID.

Parameters

- **session** (*myanimelist.session.Session*) – A valid MAL session.
- **user_id** (*int*) – The user ID for which we want to look up a username.

Raises *InvalidUserError*

Return type str

Returns The given user's username.

friends
A dict of this user's friends, with keys as *myanimelist.user.User* objects, and values as dicts of attributes, e.g.

{

'last_active': `datetime.datetime`,

'since': `datetime.datetime`

}

gender
This user's gender.

id
User ID.

join_date
A `datetime.datetime` object marking when this user joined MAL.

last_list_updates
A dict of this user's last list updates, with keys as *myanimelist.media.Media* objects, and values as dicts of attributes, e.g. {'status': str, 'episodes': int, 'total_episodes': int, 'time': datetime.datetime}

last_online
A datetime.datetime object marking when this user was active on MAL.

load()
Fetches the MAL user page and sets the current user's attributes.

Return type *User*

Returns Current user object.

load_clubs()
Fetches the MAL user clubs page and sets the current user's clubs attributes.

Return type *User*

Returns Current user object.

load_friends()
Fetches the MAL user friends page and sets the current user's friends attributes.

Return type *User*

Returns Current user object.

load_recommendations()
Fetches the MAL user recommendations page and sets the current user's recommendations attributes.

Return type *User*

Returns Current user object.

load_reviews()
Fetches the MAL user reviews page and sets the current user's reviews attributes.

Return type *User*

Returns Current user object.

location
This user's location.

manga_list()
This user's manga list.

Return type *myanimelist.manga_list.MangaList*

Returns The desired manga list.

manga_list_views
The number of times this user's manga list has been viewed.

manga_stats
A dict of this user's manga stats, with keys as strings, and values as numerics.

num_comments
The number of comments this user has made.

num_forum_posts
The number of forum posts this user has made.

parse(*user_page*)
Parses the DOM and returns user attributes in the main-content area.

Parameters **user_page** (bs4.BeautifulSoup) – MAL user page's DOM

Return type dict

Returns User attributes.

parse_clubs(*clubs_page*)
Parses the DOM and returns user clubs attributes.

Parameters **clubs_page** (bs4.BeautifulSoup) – MAL user clubs page's DOM

Return type dict

Returns User clubs attributes.

parse_friends(*friends_page*)
Parses the DOM and returns user friends attributes.

Parameters **friends_page** (bs4.BeautifulSoup) – MAL user friends page's DOM

Return type dict

Returns User friends attributes.

parse_recommendations(*recommendations_page*)
Parses the DOM and returns user recommendations attributes.

Parameters **recommendations_page** (bs4.BeautifulSoup) – MAL user recommendations page's DOM

Return type dict

Returns User recommendations attributes.

parse_reviews(*reviews_page*)
Parses the DOM and returns user reviews attributes.

Parameters **reviews_page** (bs4.BeautifulSoup) – MAL user reviews page's DOM

Return type dict

Returns User reviews attributes.

parse_sidebar(*user_page*)
Parses the DOM and returns user attributes in the sidebar.

Parameters **user_page** (bs4.BeautifulSoup) – MAL user page's DOM

Return type dict

Returns User attributes

Raises *InvalidUserError*, *MalformedUserPageError*

picture
User's picture.

recommendations
A dict of this user's recommendations, with keys as *myanimelist.media.Media* objects, and values as dicts of attributes, e.g.

{

'anime|media': *myanimelist.media.Media*,

'text': str,

'date': datetime.datetime

}

reviews
A dict of this user's reviews, with keys as *myanimelist.media.Media* objects, and values as dicts of attributes, e.g.

{

'people_helped': int,

'people_total': int,

'media_consumed': int,

'media_total': int,

'rating': int,

'text': str,

'date': datetime.datetime

}

website
This user's website.

myanimelist.utilities module

myanimelist.utilities.**extract_tags**(*tags*)

myanimelist.utilities.**fix_bad_html**(*html*)
Fixes for various DOM errors that MAL commits. Yes, I know this is a cardinal sin, but there's really no elegant way to fix this.

myanimelist.utilities.**get_clean_dom**(*html*)
Given raw HTML from a MAL page, return a BeautifulSoup object with cleaned HTML.

myanimelist.utilities.**parse_profile_date**(*text*, *suppress=False*)
Parses a MAL date on a profile page. May raise ValueError if a malformed date is found. If text is "Unknown" or "?" or "Not available" then returns None. Otherwise, returns a datetime.date object.

myanimelist.utilities.**urlencode**(*url*)
Given a string, return a string that can be used safely in a MAL url.

Module contents

3.3 Indices and tables

- genindex
- modindex
- search

m

M

N

P

R

S

T

U

V

W