pymal Documentation

Release 0.5b4

tomghel

October 08, 2014

Contents

1 Tutorial **3**
1.1 Installation 3
1.2 Usage 3

2 Coding **7**
2.1 Account 7
2.2 Account Objects 9
2.3 Anime 14
2.4 Manga 15
2.5 Seasons 17
2.6 Season 17
2.7 Exceptions 18
2.8 Review 18
2.9 Recommendation 19
2.10 Guide lines 19

3 Testing **21**
3.1 Running 21
3.2 Guide lines 21

4 The idea **23**

5 Indices and tables **25**

Python Module Index **27**

Contents:

CHAPTER 1

Tutorial

1.1 Installation

If you want to checkout a repository, navigate to the directory and run **python setup.py install**. The recommended way is to run **pip install pymal**.

1.2 Usage

Most objects data can be required by not authentication mal, but all list manipulations on MAL requires authentication.

1.2.1 account.Account

To connect MAL you need an `account.Account` object.

```
>>> from pymal.account import Account
>>> account = Account('mal-username', 'mal-password')
```

Then all your mangas and animes will be like this:

```
>>> animelist = account.animes
>>> mangalist = account.mangas
```

1.2.2 anime.Anime

Right now, give him the anime id and it will generate the most of the things.

```
>>> from pymal.anime import Anime
>>> anime = Anime(1887)  # Lucky star's anime id
```

For all data that can be used look in the python. To add it its need an `account.Account` object to related on.

```
>>> my_anime = anime.add(account)
>>> assert type(my_anime) != type(anime)
```

After adding an `anime.Anime`, you will found it in your list!

```
>>> assert my_anime not in animelist
>>> animelist.reload()
>>> assert my_anime in animelist
```

1.2.3 `account_objects.my_anime.MyAnime`

A class which has more attribute like the account's number of watched episodes and so on.

It has his `anime.Anime` under:

```
>>> assert my_anime.obj == anime
```

You can update it and delete it:

```
>>> my_anime.update()
>>> my_anime.delete()
```

1.2.4 `manga.Manga`

Right now, give him the `manga.Manga` id and it will generate the most of the things.

```
>>> from pymal.manga import Manga
>>> manga = Manga(587)  # Lucky star's manga id
```

All the objects under `account.Account` are subclass of `anime.Anime` and `manga.Manga`. To add it its need an `account.Account` object to related on.

```
>>> my_manga = manga.add(account)
>>> assert type(my_manga) != type(manga)
```

After adding an `manga.Manga`, you will found it in your list!

```
>>> assert my_manga not in mangalist
>>> mangalist.reload()
>>> assert my_manga in mangalist
```

1.2.5 `account_objects.my_manga.MyManga`

A class which has more attribute like the account's number of read chapters and so on.

It has his `manga.Manga` under:

```
>>> assert my_manga.obj == manga
```

You can update it and delete it:

```
>>> my_manga.update()
>>> my_manga.delete()
```

CHAPTER 2

Coding

Contents:

2.1 Account

It should held all the data about the account. Also it's the only one who has the password and authenticated connection. All other object should use him. Objects which connected to Account are placed under *account_object* folder.

You can find there:

- `account_objects.account_animes.AccountAnimes`
- `account_objects.account_mangas.AccountMangas`
- `account_objects.account_friends.AccountFriends`
- `account_objects.my_anime.MyAnime`
- `account_objects.my_manga.MyManga`

class `Account` (*username: str*, *password: str=None*)

Object that keeps all the account data in MAL.

`animes`

Returns list of account's animes.

Return type `account_objects.account_animes.AccountAnimes`

`auth_connect` (*url: str*, *data: str=None*, *headers: dict=None*) → str

Parameters

- **url** (*str*) – The url to get.
- **data** (*str or None*) – The data to pass (will change the request to "POST")
- **headers** (*dict or None*) – Headers for the request.

Returns data after using the account authenticate to get the data.

Return type str

change_password(*password: str*) → bool
Checking if the new password is valid

Parameters **password** (*str*) – password

Returns True if password is right.

Return type bool

Raises exceptions.FailedToParseError when failed

friends

Returns list of account's friends.

Return type `account_objects.account_friends.AccountFriends`

get_image()

Returns image of the account's avatar

Return type `PIL.Image.Image`

is_auth

Returns True if the password is right (and able to authenticate).

Return type bool

mangas

Returns list of account's mangas.

Return type `account_objects.account_mangas.AccountMangas`

reload()
reloading account image (all the other things are already lazy load!

search(*search_line: str*, *is_anime: bool=True*) → map
Searching like regular search but switching all the object in "my" lists to the "my" objects.

Parameters

- **search_line** (*str*) – the search line.
- **is_anime** (*bool*) – True is searching for anime, False for manga.

Returns searched objects

Return type map

user_id

Returns the user id. If unknown loading it.

Return type int

username

Returns the user name.

Return type str

2.2 Account Objects

All accounts connected object are placed here.

account's lists:

2.2.1 AccountAnimes

It should lazy load accounts' list of animes. It can held future global data about accounts' animes. It shouldn't be used out side of context of account

In the package it only a package for list(MyAnime).

class **`AccountAnimes`**(*account*)

A slow loading of an account anime list.

Variables

- **watching** – `frozenset` of the watching animes.
- **completed** – `frozenset` of the completed animes.
- **on_hold** – `frozenset` of the "on hold" animes.
- **dropped** – `frozenset` of the dropped animes.
- **plan_to_watch** – `frozenset` of th "plan to watch" animes.

`reload`()

reloading data from MAL.

2.2.2 AccountMangas

It should lazy load accounts' list of mangas. It can held future global data about accounts' mangas. It shouldn't be used out side of context of account.

In the package it only a package for list(MyManga).

class **`AccountMangas`**(*account*)

A slow loading of an account anime list.

Variables

- **reading** – `frozenset`
- **completed** – `frozenset`
- **on_hold** – `frozenset`
- **dropped** – `frozenset`

- **plan_to_read** – `frozenset`

`reload``()`
reloading data from MAL.

2.2.3 AccountFriends

It should lazy load accounts' list of friends. It can held future global data about accounts' friends. It shouldn't be used out side of context of account.

In the package it only a package for list(Account).

class `AccountFriends``(`*account*`)`
A slow load of friend list.

Variables account – the account to load his friends.

`reload``()`
:exception FailedToParseError

account's owned objects:

2.2.4 MyAnime

This object and `my_manga.MyManga` should have a very close interface (except for volumes-chapters vs episodes). A basic object to obtain account specific data about an anime. Can manipulate the anime data in the account's list.

class `MyAnime``(`*mal_id: int*, *my_mal_id*, *account*`)`
Saves an account data about anime.

Variables

- **my_enable_discussion** – boolean
- **my_id** – int
- **my_status** – int. #TODO: put the dictionary here.
- **my_score** – int.
- **my_start_date** – string as mmddyyyy.
- **my_end_date** – string as mmddyyyy.
- **my_priority** – int.
- **my_storage_type** – int. #TODO: put the dictanary here.
- **my_storage_value** – float.
- **my_is_rewatching** – boolean.
- **my_completed_episodes** – int.
- **my_download_episodes** – int.

- **my_times_rewatched** – int.
- **my_rewatch_value** – int.
- **my_tags** – frozenset.
- **my_comments** – string
- **my_fan_sub_groups** – string.

`add`(*account*)
Adding the anime to an account. If its the same account as this owner returning this.

Parameters **account** (`account.Account`) – account to connect to the anime.

Returns anime connected to the account

Return type `account_objects.my_anime.MyAnime`

`delete`()
Deleteing the anime from the list.

`increase`() → bool
Increasing the watched episode. If it is completed, setting the flag of rewatching.

Returns True if succeed to set every.

Return type bool

`increase_downloaded`() → bool
Increasing the downloaded episode.

Returns True if succeed to set every.

Return type bool

`is_completed`

Returns True if the number of completed episode is equal to number of episode in anime.

Return type bool

`my_id`

Returns the id in the account.

Return type int

`my_reload`()
Reloading data from MAL.

`set_completed`() → bool
Setting the anime as completed.

Returns True if succeed

Return type bool

set_completed_download() → bool
Setting the number of downloaded episodes as completed.

Returns True if succeed

Return type bool

to_xml()

Returns the anime as an xml string.

Return type str

update()
Updating the anime data.

2.2.5 MyManga

This object and `my_anime.MyAnime` should have a very close interface (except for volumes-chapters vs episodes). A basic object to obtain account specific data about a manga. Can manipulate the manga data in the account's list.

class **MyManga** (*mal_id: int*, *my_mal_id*, *account*)
Saves an account data about manga.

Variables

- **my_enable_discussion** – boolean
- **my_id** – int
- **my_status** – int. #TODO: put the dictanary here.
- **my_score** – int.
- **my_start_date** – string as mmddyyyy.
- **my_end_date** – string as mmddyyyy.
- **my_priority** – int.
- **my_storage_type** – int. #TODO: put the dictnary here.
- **my_is_rereading** – boolean.
- **my_completed_chapters** – int.
- **my_completed_volumes** – int.
- **my_downloaded_chapters** – int.
- **my_times_reread** – int.
- **my_reread_value** – int.
- **my_tags** – frozenset.
- **my_comments** – string
- **my_fan_sub_groups** – string.

- **my_retail_volumes** – int.

add(*account*)

Adding the anime to an account. If its the same account as this owner returning this.

Parameters **account** (`account.Account`) – account to connect to the anime.

Returns anime connected to the account

Return type `accounts_objects.MyManga.MyManga`

delete()

Deleteing the anime from the list.

increase() → bool

Increasing the read chapters. If it is completed, setting the flag of rereading.

Returns True if succeed to set every.

Return type bool

increase_downloaded() → bool

Increasing the downloaded chapters.

Returns True if succeed to set every.

Return type bool

increase_volume() → bool

Increasing the read volumes. If it is completed, setting the flag of rereading.

Returns True if succeed to set every.

Return type bool

is_completed

Returns True if the number of completed chapters is equal to number of chapters in manga.

Return type bool

my_id

Returns the id in the account.

Return type int

my_reload()

Reloading data from MAL.

set_completed() → bool

Setting the anime as completed.

Returns True if succeed

Return type bool

set_completed_download() → bool
Setting the number of downloaded chapters as completed.

Returns True if succeed

Return type bool

to_xml()

Returns the anime as an xml string.

Return type str

update()
Updating the anime data.

2.3 Anime

This object and `manga.Manga` should have a very close interface (except for volumes-chapters vs episodes). A very basic object to obtain generic data about an anime.

class Anime(*mal_id: int*)
Object that keeps all the anime data in MAL.

Variables

- **image_url** – `str`
- **title** – `str`
- **english** – `str`
- **synonyms** – `str`
- **japanese** – `str`
- **type** – `str`
- **status** – `int`
- **start_time** – `int`
- **end_time** – `int`
- **creators** – `dict`
- **genres** – `dict`
- **duration** – `int`
- **score** – `float`
- **rank** – `int`
- **popularity** – `int`
- **rating** – `str`
- **episodes** – `int`

- **synopsis** – `str`
- **adaptations** – `frozenset`
- **characters** – `frozenset`
- **sequels** – `frozenset`
- **prequels** – `frozenset`
- **spin_offs** – `frozenset`
- **alternative_versions** – `frozenset`
- **side_stories** – `frozenset`
- **summaries** – `frozenset`
- **others** – `frozenset`
- **parent_stories** – `frozenset`
- **alternative_settings** – `frozenset`
- **full_stories** – `frozenset`

`add`(*account*)

Parameters account (`account.Account`) – the account to add him self anime.

Return type `account_objects.my_anime.MyAnime`

Raises exceptions.MyAnimeListApiAddError when failed.

`get_image`()

Returns The image of the anime

Return type `PIL.Image.Image`

`id`

Returns The id of the anime.

Return type `int`

`reload`()

Raises exceptions.FailedToReloadError when failed.

2.4 Manga

This object and `anime.Anime` should have a very close interface (except for volumes-chapters vs episodes). A very basic object to obtain generic data about a manga.

class `Manga`(*mal_id: int*)

Object that keeps all the anime data in MAL.

Variables

- **title** – `str`
- **image_url** – `str`
- **english** – `str`
- **synonyms** – `str`
- **japanese** – `str`
- **type** – `str`
- **status** – `int`
- **start_time** – `int`
- **end_time** – `int`
- **creators** – `dict`
- **genres** – `dict`
- **duration** – `int`
- **score** – `float`
- **rank** – `int`
- **popularity** – `int`
- **rating** – `str`
- **chapters** – `int`
- **volumes** – `int`
- **synopsis** – `str`
- **adaptations** – `frozenset`
- **characters** – `frozenset`
- **sequels** – `frozenset`
- **prequels** – `frozenset`
- **spin_offs** – `frozenset`
- **alternative_versions** – `frozenset`
- **side_stories** – `frozenset`
- **summaries** – `frozenset`
- **others** – `frozenset`
- **parent_stories** – `frozenset`
- **alternative_settings** – `frozenset`

add(*account*)

Parameters **account** (`account.Account`) – the account to add him self manga.

Raises **exceptions.MyAnimeListApiAddError** when failed.

Return type `account_objects.my_manga.MyManga`

get_image()

Returns The manga's image.

Return type `PIL.Image.Image`

id

Returns the mangas id.

Return type `int`

reload()

Raises **exceptions.FailedToReloadError** when failed.

2.5 Seasons

This object is loaded from a different db.

class **Seasons**

Lazy making of Season from online db.

Variables **seasons** – `frozenset` of `inner_objects..season.Season`.

reload()

reloading all the known seasons.

2.6 Season

An inner object of Seasons. Hold frozenset of animes in the specific season (by name and year).

class **Season**(*season_name: str*, *year: int*)

Lazy load of season data.

Attributes: animes - a frozenset of animes. year - the season year. season_name - The season name. Can be 'Winter', 'Spring', 'Summer' or 'Fall'.

reload()

fetching data.

2.7 Exceptions

All the exceptions in pymal are placed here. Create a new file for each large exception - if it got inheritance or some exception that walking together.

2.7.1 FailedToParseError

class **FailedToParseError**
    Bases: `builtins.RuntimeError`

class **FailedToReloadError**
    Bases: `exceptions.failed_to_parse_error.FailedToParseError`

class **FailedToAddError**
    Bases: `exceptions.failed_to_parse_error.FailedToParseError`

2.7.2 MyAnimeListApiError

class **MyAnimeListApiError**
    Bases: `builtins.RuntimeError`

class **MyAnimeListApiUpdateError**
    Bases: `exceptions.my_anime_list_api_error.MyAnimeListApiError`

class **MyAnimeListApiDeleteError**
    Bases: `exceptions.my_anime_list_api_error.MyAnimeListApiError`

class **MyAnimeListApiAddError**
    Bases: `exceptions.my_anime_list_api_error.MyAnimeListApiError`

2.7.3 Others

class **UnauthenticatedAccountError**
    Bases: `builtins.ValueError`

class **NotASeasonError**(*tried_season_name*)
    Bases: `builtins.ValueError`

class **GotRobotError**
    Bases: `builtins.RuntimeError`

YOU SHOULD NEVER CREATE ANY OF THOSE BY YOUR-SELF

2.8 Review

Holds data about a Review of `anime.Anime` or `manga.Manga`.

class `Review`(*div*)

Review holds all the data from a review in MAL about an anime.

Variables

- **date** – `string`
- **account** – `account.Account`
- **helpful** – `int`
- **watched** – `int`
- **when_written** – `string`
- **rating** – `int`
- **data** – `string`

2.9 Recommendation

holds data from an `anime.Anime` or `manga.Manga`.

class `Recommendation`(*div*)

Recommendation holds all the data from a recommendation in MAL about an anime.

Variables

- **recommended_anime** – `anime.Anime`
- **recommends** – `dict`

2.10 Guide lines

Some basic rules:

1. Tests are in their own directory tree that is equals to the real tree. Never put tests in here! That's why we have a folder for them :)
2. Remember that we are reading from anther server (myanimelist.net). Make everything as lazy as possible and use all the information from each data you receive.
3. Objects with same interface should inheritance from the same interface.

2.10.1 Q: Where do i put a new object?

A: Each object should be placed in his own file.

2.10.2 Q: Do you have any global files?

A: yes we do! * *global_function.py* - Here you should find and place all the globals functions. Any function that most of other code will use and its not better any where. Some people will call it 'junk file'. * *consts.py* - Here you should find and place all the constants of the project. More constants out here means better sharing and finding even for the test! * *decorators.py* - Here you should find and place all the global decorators. I don't recommend to make a lot of them, only if necessary or looks better.

CHAPTER 3

Testing

If you read this that mean that you want to do something with the code (read or add). This README will try to explain how all the tests are organized and how it should look like.

3.1 Running

To run the tests that come with pyMAL:

1. Install nose (A package for running tests - **pip install nose**).
2. Navigate to the pymal directory
3. Run **nosetests**.

3.2 Guide lines

Questions about our testing and their answers.

3.2.1 Q: What framework we are using?

A: We are using the basic unittest that python gives us. Please don't add anything else like **py.test** and his friends.

3.2.2 Q: What do we check?

A: Every object in pymal should be tested with all his function and flows. Check the returned types and their data. Tests with mocks and with connection to MAL.

3.2.3 Q: Where to put my consts

A: If they are only for your test put them in your TestCase class. Otherwise, it should be place in `constants_for_testing.py`.

3.2.4 Q: And that's all?

A: Pretty much yes, but remember to make asserts informative by code or by message.

CHAPTER 4

The idea

Provides programmatic access to MyAnimeList data with python. Objects in pymal are lazy-loading: they won't go out and fetch MAL info until you first-request it.

CHAPTER 5

Indices and tables

- *genindex*
- *modindex*

a

account, 7
account_objects.account_animes, 9
account_objects.account_friends, 10
account_objects.account_mangas, 9
account_objects.my_anime, 10
account_objects.my_manga, 12
anime, 14

e

exceptions, 18
exceptions.failed_to_parse_error, 18
exceptions.my_anime_list_api_error, 18

i

inner_objects.recommendation, 19
inner_objects.review, 18
inner_objects.season, 17

m

manga, 15

s

seasons, 17

A

Account (class in account), 7
account (module), 7
account_objects.account_animes (module), 9
account_objects.account_friends (module), 10
account_objects.account_mangas (module), 9
account_objects.my_anime (module), 10
account_objects.my_manga (module), 12
AccountAnimes (class in account_objects.account_animes), 9
AccountFriends (class in account_objects.account_friends), 10
AccountMangas (class in account_objects.account_mangas), 9
add() (Anime method), 15
add() (Manga method), 16
add() (MyAnime method), 11
add() (MyManga method), 13
Anime (class in anime), 14
anime (module), 14
animes (Account attribute), 7
auth_connect() (Account method), 7

C

change_password() (Account method), 8

D

delete() (MyAnime method), 11
delete() (MyManga method), 13

E

exceptions (module), 18
exceptions.failed_to_parse_error (module), 18
exceptions.my_anime_list_api_error (module), 18

F

FailedToAddError (class in exceptions.failed_to_parse_error), 18
FailedToParseError (class in exceptions.failed_to_parse_error), 18
FailedToReloadError (class in exceptions.failed_to_parse_error), 18
friends (Account attribute), 8

G

get_image() (Account method), 8
get_image() (Anime method), 15
get_image() (Manga method), 17
GotRobotError (class in exceptions), 18

I

id (Anime attribute), 15
id (Manga attribute), 17
increase() (MyAnime method), 11
increase() (MyManga method), 13
increase_downloaded() (MyAnime method), 11
increase_downloaded() (MyManga method), 13
increase_volume() (MyManga method), 13
inner_objects.recommendation (module), 19
inner_objects.review (module), 18
inner_objects.season (module), 17
is_auth (Account attribute), 8
is_completed (MyAnime attribute), 11
is_completed (MyManga attribute), 13

M

Manga (class in manga), 15

manga (module), 15
mangas (Account attribute), 8
my_id (MyAnime attribute), 11
my_id (MyManga attribute), 13
my_reload() (MyAnime method), 11
my_reload() (MyManga method), 13
MyAnime (class in account_objects.my_anime), 10
MyAnimeListApiAddError (class in exceptions.my_anime_list_api_error), 18
MyAnimeListApiDeleteError (class in exceptions.my_anime_list_api_error), 18
MyAnimeListApiError (class in exceptions.my_anime_list_api_error), 18
MyAnimeListApiUpdateError (class in exceptions.my_anime_list_api_error), 18
MyManga (class in account_objects.my_manga), 12

N

NotASeasonError (class in exceptions), 18

R

Recommendation (class in inner_objects.recommendation), 19
reload() (Account method), 8
reload() (AccountAnimes method), 9
reload() (AccountFriends method), 10
reload() (AccountMangas method), 10
reload() (Anime method), 15
reload() (Manga method), 17
reload() (Season method), 17
reload() (Seasons method), 17
Review (class in inner_objects.review), 18

S

search() (Account method), 8
Season (class in inner_objects.season), 17
Seasons (class in seasons), 17
seasons (module), 17
set_completed() (MyAnime method), 11
set_completed() (MyManga method), 13
set_completed_download() (MyAnime method), 11
set_completed_download() (MyManga method), 13

T

to_xml() (MyAnime method), 12
to_xml() (MyManga method), 14

U

UnauthenticatedAccountError (class in exceptions), 18
update() (MyAnime method), 12
update() (MyManga method), 14
user_id (Account attribute), 8
username (Account attribute), 8