ttrss-python Documentation

Release 0.1.4

Markus Wiik

April 20, 2013

CONTENTS

ttrss-python is a light-weight client library for the JSON API of Tiny Tiny RSS. Handling JSON in Python can be quite a pain, so this library abstracts all that away to deliver Python object representations of categories, feeds and articles. Also, the specific calls and POST data sent to the server is handled automatically, so that you can focus on the stuff that matters to your frontend application.

Sounds good so far? Great, let's get started!

CHAPTER

ONE

INSTALLATION

This package is available at PyPI, so the easiest way to install is by doing a `pip install ttrss-python`. This will install the latest released version, as well as all dependencies. Currently, the only dependency is the awesome Python `requests` library for handling all the http requests.

If you for some reason can't or don't want to use `pip`, just download the tarball and run `python setup.py install` manually.

It's highly recommended to use `virtualenv` for this (and any other for that matter!) installation.

CHAPTER

TWO

BASIC USAGE

The first thing you need to do is instantiate a `TTRClient`. The constructor requires three arguments; URL, your username and your password:

```
>>> from ttrss.client import TTRClient
>>> client = TTRClient('http://url-to-tiny-tiny', 'username', 'super-secret-password')
>>> client.login()
```

If you want the client to login automatically, as well as automatically refresh an expired session cookie, you may supply the optional keyword argument `auto_login=True`. Note that this may affect performance in a high-traffic client application, since it uses a response hook to check every server response for a `NOT_LOGGED_IN` message:

```
>>> client = TTRClient('http://url-to-tiny-tiny', 'username', 'super-secret-password', auto_login=Tru
```

Refer to the API docs for details on how to retrieve objects from the server.

2.1 Categories

Category objects contain attributes describing the category, as well as a method to retrieve feeds contained in it. Assuming a category object called `cat`:

```
>>> cat.title
u'Example category'
>>> cat.unread
20
>>> cat.id
2
```

To retrieve a list of feeds belonging to this category, simply type:

```
>>> cat.feeds()
[<ttrss.client.Feed object at 0x103a0cfd0>, <ttrss.client.Feed object at 0x103478a50>]
```

The `feeds` method accepts parameters as well. Please refer to the API docs for details.

CHAPTER

THREE

FEEDS

Like category objects, feed objects contain metadata and a method to retrieve headlines:

```
>>> feed.title
u'MacRumors: Mac News and Rumors - All Stories'
>>> feed.last_updated
datetime.datetime(2013, 3, 24, 21, 18, 29)
>>> feed.unread
24
>>> feed.feed_url
u'http://feeds.macrumors.com/MacRumors-All'
>>> feed.id
5
>>> feed.headlines()
[<ttrss.client.Headline object at 0x103a0cfd0>, ...]
```

CHAPTER

FOUR

HEADLINES

Headlines are short versions of articles. They too include all useful metadata:

```
>>> headline.title
u'Apple Acquires Indoor Mobile Location Positioning Firm WifiSLAM for $20 Million'
>>> headline.excerpt
u'The Wall Street Journal reports that Apple has acquired indoor location company WifiSLAM, paying ar
>>> headline.link
u'http://www.macrumors.com/2013/03/23/apple-acquires-indoor-mobile-location-positioning-firm-wifislar
>>> headline.updated
datetime.datetime(2013, 3, 24, 21, 18, 29)
>>> headline.unread
True
>>> headline.tags
[u'front page']
>>> headline.published
True
>>> headline.labels
[]
>>> headline.id
1
>>> headline.feed_id
u'5'
```

To get the full article, simply type:

```
>>> headline.full_article()
<ttrss.client.Article object at 0x103a0cf90>
```

CHAPTER

FIVE

ARTICLES

Article objects include all the useful information:

```
>>> article.link
u'http://www.macrumors.com/2013/03/23/apple-acquires-indoor-mobile-location-positioning-firm-wifislar
>>> article.title
u'Apple Acquires Indoor Mobile Location Positioning Firm WifiSLAM for $20 Million'
>>> article.updated
datetime.datetime(2013, 3, 24, 21, 18, 29)
>>> article.comments
u''
>>> article.author
u'Eric Slivka'
>>> article.id
1
>>> article.unread
True
>>> article.content
u"Lots of text... "
```

Article objects also include some useful methods for interaction:

```
>>> article.publish()        # Publish to shared
>>> article.toggle_unread() # Toggle unread status
```

You may also refresh the information about an article with fresh data from the server. This is useful if you have a long-running script and interact with the server by other means while it's running:

```
>>> article.unread
True
# Mark the article as read in the web interface or some other client...
>>> article.refresh_status()
>>> article.unread
False
```

CHAPTER

SIX

DEVELOPMENT

This project is open source and MIT licensed. The source code is available at https://github.com/Vassius/ttrss-python

6.1 Contributions

Bugreports, feature requests and other feedback is very much appreciated, and best submitted in the GitHub project mentioned above. If you're a doer and want to patch something yourself - Awesome! Just fork my repo and submit a pull request.

6.2 Author

`ttrss-python` is written and maintained by Markus Wiik <vassius@gmail.com>

CHAPTER

SEVEN

API DOCUMENTATION

This section will become more detailed over time as I add docstrings to the source code.

7.1 `client` module

class `ttrss.client.`**`Article`**(*attr*, *client*)
Bases: `ttrss.client.RemoteObject`

`publish`()
Share this article to published feed

`refresh_status`()
Refresh this object with new data fetched from the server.

`toggle_unread`()
Toggle unread status of this article

class `ttrss.client.`**`Category`**(*attr*, *client=None*)
Bases: `ttrss.client.RemoteObject`

`feeds`(***kwargs*)
Get a list of feeds for this category.

Parameters

- **unread_only** – *Optional* Include only feeds containing unread articles. Default is `False`.
- **limit** – *Optional* Limit number of included feeds to `limit`. Default is 0 (unlimited).
- **offset** – *Optional* Skip this number of feeds. Useful for pagination. Default is 0.
- **include_nested** – *Optional* Include child categories. Default is `False`.

class `ttrss.client.`**`Feed`**(*attr*, *client*)
Bases: `ttrss.client.RemoteObject`

`catchup`()
Mark this feed as read

`headlines`(***kwargs*)
Get a list of headlines from a this feed.

Parameters

- **limit** – Return no more than this number of headlines. Default is `0` (unlimited, though the server limits to 60).

- **skip** – Skip this number of headlines. Useful for pagination. Default is `0`.
- **show_excerpt** – Include a short excerpt of the article. Defaults to `True`.
- **show_content** – Include full article content. Defaults to `False`.
- **view_mode** – (string = all_articles, unread, adaptive, marked, updated)
- **include_attachments** – include article attachments. Defaults to `False`.
- **since_id** – Only include headlines newer than `since_id`.
- **include_nested** – Include articles from child categories. Defaults to `True`.

class `ttrss.client.`**`Headline`**(*attr*, *client*)
Bases: `ttrss.client.RemoteObject`

This class represents Headline objects. A headline is a short version of an article.

`full_article`()
Get the full article corresponding to this headline

class `ttrss.client.`**`RemoteObject`**(*attr*, *client=None*)
Bases: `object`

This is the base class for representing remote resources as Python objects.

class `ttrss.client.`**`TTRClient`**(*url*, *user=None*, *password=None*, *auto_login=False*)
Bases: `object`

This is the actual client interface to Tiny Tiny RSS.

This object retains a http session with the needed session cookies. From the client you can fetch categories, feeds, headlines and articles, all represented by Python objects. You can also update modify articles and feeds on the server.

`catchup_feed`(*feed_id*, *is_cat=False*)
Attempt to mark all articles in specified feed as read.

Parameters

- **feed_id** – id of the feed to catchup.
- **is_cat** – Specified feed is a category. Default is False.

`get_articles`(*article_id*)
Get a list of articles from article ids.

Parameters **article_id** – A comma separated string of article ids to fetch.

`get_categories`(*unread_only=False*, *enable_nested=False*, *include_empty=False*)
Get a list of all available categories

Parameters

- **unread_only** – Only return categories containing unread articles. Defaults to `False`.
- **enable_nested** – When enabled, traverse through sub-categories and return only the **top-most** categories in a flat list. Defaults to `False`.
- **include_empty** – Include categories not containing any feeds. Defaults to `False`. *Requires server version 1.7.6*

`get_feed_count`()
Get total number of subscribed feeds.

get_feeds(*cat_id=-1*, *unread_only=False*, *limit=0*, *offset=0*, *include_nested=False*)
Get a list of feeds in a category.

Parameters

- **cat_id** – Category id. This is available as the `id` property of a Category object.
- **unread_only** – *Optional* Include only feeds containing unread articles. Default is `False`.
- **limit** – *Optional* Limit number of included feeds to `limit`. Default is 0 (unlimited).
- **offset** – *Optional* Skip this number of feeds. Useful for pagination. Default is 0.
- **include_nested** – *Optional* Include child categories. Default is `False`.

get_headlines(*feed_id=-4*, *limit=0*, *skip=0*, *is_cat=False*, *show_excerpt=True*, *show_content=False*, *view_mode=None*, *include_attachments=False*, *since_id=None*, *include_nested=True*)
Get a list of headlines from a specified feed.

Parameters

- **feed_id** – Feed id. This is available as the `id` property of a Feed object. Default is `-4` (all feeds).
- **limit** – Return no more than this number of headlines. Default is `0` (unlimited, though the server limits to 60).
- **skip** – Skip this number of headlines. Useful for pagination. Default is `0`.
- **is_cat** – The feed_id is a category. Defaults to `False`.
- **show_excerpt** – Include a short excerpt of the article. Defaults to `True`.
- **show_content** – Include full article content. Defaults to `False`.
- **view_mode** – (string = all_articles, unread, adaptive, marked, updated)
- **include_attachments** – include article attachments. Defaults to `False`.
- **since_id** – Only include headlines newer than `since_id`.
- **include_nested** – Include articles from child categories. Defaults to `True`.

get_unread_count()
Get total number of unread articles

logged_in()

login()
Manually log in (i.e. request a session cookie)

This method must be used if the client was not instantiated with `auto_login=True`

logout()
Log out.

After logging out, `login()` must be used to gain a valid session again. Please note that logging out invalidates any automatic re-login even after logging back in.

mark_read(*article_id*)
Mark an article as read.

Parameters **article_id** – ID of article to mark as read.

mark_unread(*article_id*)
Mark an article as unread.

Parameters **article_id** – ID of article to mark as unread.

refresh_article(*article*)
Update all properties of an article object with fresh information from the server.

Please note that this method alters the original object and does not return a new one.

Parameters **article** – The article to refresh.

share_to_published(*title*, *url*, *content*)
Share an article to the *published* feed.

Parameters

- **title** – Article title.
- **url** – Article url.
- **content** – Article content.

toggle_unread(*article_id*)
Toggle the unread status of an article.

Parameters **article_id** – ID of the article to toggle.

CHAPTER

EIGHT

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

t