py420chan Documentation

Release 0.0.3

Antonizoon Overtwater

Nov 26, 2018

Contents

1 General Documentation **3**
- 1.1 Tutorial . . . 3
- 1.2 Changes from the original py4chan . . . 4

2 API Documentation **7**
- 2.1 `py420chan` – 420chan Python Library . . . 7
- 2.2 `py420chan.Board` – 420chan Boards . . . 8
- 2.3 `py420chan.Thread` – 420chan Threads . . . 10
- 2.4 `py420chan.Post` – 420chan Post . . . 11
- 2.5 `py420chan.File` – 420chan File . . . 13

Python Module Index **15**

py420chan is a Python library that gives access to the 420chan API and an object-oriented way to browse and get board and thread information quickly and easily.

py420chan is based on BASC-py4chan, a 4chan API wrapper that was adopted and extended by the Bibliotheca Anonoma.

The py420chan repository is located on Github, where pull requests and issues can be submitted.

Getting Help If you want help, or you have some trouble using this library, our primary IRC channel is #bibanon on irc.rizon.net. Simply head in there and talk to dan or antonizoon. Otherwise, you can put a issue on our Github Issue Tracker and we'll respond as soon as we can!

CHAPTER 1

General Documentation

1.1 Tutorial

When using BASC-py4chan, it can be a bit hard to find where to begin. Here, we run through how to create and use the various objects available in this module.

1.1.1 Boards

`basc_py4chan.Board` is the first thing you create when using BASC-py4chan. Everything else is created through that class. The most basic way to create a board is as below:

```
board = basc_py4chan.Board('tg')
```

This creates a `basc_py4chan.Board` object that you can then use to create `basc_py4chan.Thread` and `basc_py4chan.Post` objects.

But what sort of things does a `basc_py4chan.Board` object let you do?

Here's a short code snippet of us printing out how many threads are active on a board:

```
board = basc_py4chan.Board('tg')
thread_ids = board.get_all_thread_ids()
str_thread_ids = [str(id) for id in thread_ids]  # need to do this so str.join below␣
↪works
print('There are', len(all_ids), 'active threads on /tg/:', ', '.join(str_thread_ids))
```

1.1.2 Threads

Listing how many threads exist on a board is all well and good, but most people want to actually get threads and do things with them. Here, we'll describe how to do that.

All `basc_py4chan.Thread` objects are created by a `basc_py4chan.Board` object, using one of the `basc_py4chan.Board.get_thread()` methods.

For this example, we have a user ask us about "thread 1234", and we return information about it:

```
thread_id = 1234
board = basc_py4chan.Board('tg')

if board.thread_exists(thread_id):
    thread = board.get_thread(thread_id)

    # print thread information
    print('Thread', thread_id)
    if thread.closed:
        print('  is closed')
    if thread.sticky
        print('  is a sticky')

    # information from the OP
    topic = thread.topic
    print('  is named:', topic.subject)
    print('  and was made by:', name, email)
```

1.2 Changes from the original py4chan

Since Edgeworth has gone MIA, BASC has adopted the project and made the following improvements.

1.2.1 Changes by antonizoon

- **4chan Link Structure Update** - 4chan has heavily reformed it's link structure, finally removing the strange folder structure inherited from the Futaba Channel.
- **4chan cdn Link update** - To save money on bandwidth. 4chan has changed it's image/thumbnail/json/css servers to a domain name with fewer characters.
- **Thread Class:** new `filenames()` function that return the filenames of all files (not thumbnails) in a thread.
- **Thread Class:** new `thumbnames()` function that return the filenames of all thumbnails in a thread.
  - **Post Class:** new `image_fname` and `thumbnail_fname` properties, designed for Thread Class `filenames()` and `thumbnames()`.
- **Actual API Documentation** - Real documentation on using the py-4chan library is a must. For some people, it is rocket science.

1.2.2 Changes by Anorov

- **Anorov's underscore_function_notation** - Even I have to say that CamelCase is beginning to suck, so we've adopted Anorov's function notation for py4chan. This breaks API compatibility with the original py-4chan, but just use find/replace to change your functions.
- **Break up classes into separate files.** - Makes the code much cleaner.
- Thread Class: `expand()` function, used to display omitted posts and images. Used by all_posts().
- Thread Class: `semantic_thread_url()` function, used to obtain 4chan's new URL format, which tacks on the thread title (obtained from `slug()`).

- Post Class: `comment()` has been modified to use `clean_comment_body()` when returning a comment. The raw text from the 4chan API can still be obtained from `orig_comment()`.
    - Util Class: `clean_comment_body()` function, which converts all HTML tags and entities within 4chan comments into human-readable text equivalents.(e.g. `<br>` to a newline, `<a href>` into a raw link)
- Board Class: `_get_json()` function, which dumps the raw JSON from the 4chan API.
- A whole host of new Catalog parsing functions:
    - Board Class: `refresh_cache()` and `clear_cache()` - Get the latest Catalog of all threads in the board, or clear the current cache.
    - Board Class: `get_threads(page)` - Get a list of all threads on a certain page. (Pages are now indexed starting from 1).
    - Board Class: `get_all_thread_ids()` - Get a list of all thread IDs on the board.
    - Board Class: `get_all_threads()` - Return all threads on all pages in the board.

1.2.3 Changes by Daniel Oaks

- **ReadTheDocs Documentation** - Splitting the documentation out to ReadTheDocs, using Sphinx to generate nice, useful docs!

CHAPTER 2

API Documentation

2.1 `py420chan` – 420chan Python Library

`py420chan` gives access to 420chan from a clean Python interface.

2.1.1 Basic Usage

420chan Python Library.

py420chan is a Python library that gives access to the 420chan API and an object-oriented way to browse and get board and thread information quickly and easily.

2.1.2 Methods

`py420chan.`**`get_boards`**`(`*board_name_list*, **args*, ***kwargs*`)`

Given a list of boards, return `basc_py4chan.Board` objects.

Parameters **`board_name_list`** (*`list`*) – List of board names to get, eg: [‘b’, ‘tg’]

Returns Requested boards.

Return type dict of `basc_py4chan.Board`

`py420chan.`**`get_all_boards`**`(`**args*, ***kwargs*`)`

Returns every board on 4chan.

Returns All boards.

Return type dict of `basc_py4chan.Board`

2.2 `py420chan.Board` – 420chan Boards

py420chan.Board provides access to a 420chan board including checking if threads exist, retrieving appropriate *py420chan.Thread* objects, and returning lists of all the threads that exist on the given board.

2.2.1 Example

Here is a sample application that grabs and uses Board information:

```
from __future__ import print_function
import py420chan

board = py420chan.Board('tg')
thread_ids = board.get_all_thread_ids()
str_thread_ids = [str(id) for id in thread_ids]  # need to do this so str.join below␣
↪works
print('There are', len(all_ids), 'active threads on /tg/:', ', '.join(str_thread_ids))
```

2.2.2 Basic Usage

class py420chan.**Board**(*board_name*, *https=False*, *session=None*)
Represents a Board.

name
Name of this board, such as `tg` or `k`.

Type str

name
Name of the board, such as "tg" or "etc".

Type string

title
Board title, such as "Animu and Mango".

Type string

is_worksafe
Whether this board is worksafe.

Type bool

page_count
How many pages this board has.

Type int

threads_per_page
How many threads there are on each page.

Type int

2.2.3 Methods

Board.**__init__**(*board_name*, *https=False*, *session=None*)
Creates a `basc_py4chan.Board` object.

Parameters

- **`board_name`** (*`string`*) – Name of the board, such as “tg” or “etc”.
- **`https`** (*`bool`*) – Whether to use a secure connection to 4chan.
- **`session`** – Existing requests.session object to use instead of our current one.

`Board.`**`thread_exists`**(*thread_id*)

Check if a thread exists or has 404’d.

Parameters **`thread_id`** (*`int`*) – Thread ID

Returns Whether the given thread exists on this board.

Return type bool

`Board.`**`get_thread`**(*thread_id*, *update_if_cached=True*, *raise_404=False*)

Get a thread from 4chan via 4chan API.

Parameters

- **`thread_id`** (*`int`*) – Thread ID
- **`update_if_cached`** (*`bool`*) – Whether the thread should be updated if it’s already in our cache
- **`raise_404`** (*`bool`*) – Raise an Exception if thread has 404’d

Returns Thread object

Return type `basc_py4chan.Thread`

`Board.`**`get_threads`**(*page=0*)

Returns all threads on a certain page.

Gets a list of Thread objects for every thread on the given page. If a thread is already in our cache, the cached version is returned and thread.want_update is set to True on the specific thread object.

Pages on 420chan are indexed from 0 onwards.

Parameters **`page`** (*`int`*) – Page to request threads for. Defaults to the first page.

Returns List of Thread objects representing the threads on the given page.

Return type list of `basc_py4chan.Thread`

`Board.`**`get_all_threads`**(*expand=False*)

Return every thread on this board.

If not expanded, result is same as get_threads run across all board pages, with last 3-5 replies included.

Uses the catalog when not expanding, and uses the flat thread ID listing at /{board}/threads.json when expanding for more efficient resource usage.

If expanded, all data of all threads is returned with no omitted posts.

Parameters **`expand`** (*`bool`*) – Whether to download every single post of every thread. If enabled, this option can be very slow and bandwidth-intensive.

Returns List of Thread objects representing every thread on this board.

Return type list of `basc_py4chan.Thread`

`Board.`**`get_all_thread_ids`**()

Return the ID of every thread on this board.

Returns List of IDs of every thread on this board.

Return type list of ints

`Board.`**`refresh_cache`**`(`*if_want_update=False*`)`
Update all threads currently stored in our cache.

`Board.`**`clear_cache`**`()`
Remove everything currently stored in our cache.

2.3 `py420chan.Thread` – 420chan Threads

`py420chan.Thread` allows for standard access to a 420chan thread, including listing all the posts in the thread, information such as whether the thread is locked and stickied, and lists of attached file URLs or thumbnails.

2.3.1 Basic Usage

`class` `py420chan.`**`Thread`**`(`*board*, *id*`)`
Represents a thread.

`closed`
Whether the thread has been closed.

Type bool

`sticky`
Whether this thread is a 'sticky'.

Type bool

`topic`
Topic post of the thread, the OP.

Type `basc_py4chan.Post`

`posts`
List of all posts in the thread, including the OP.

Type list of `basc_py4chan.Post`

`all_posts`
List of all posts in the thread, including the OP and any omitted posts.

Type list of `basc_py4chan.Post`

`url`
URL of the thread, not including semantic slug.

Type string

`semantic_url`
URL of the thread, with the semantic slug.

Type string

`semantic_slug`
The 'pretty URL slug' assigned to this thread by 4chan.

Type string

2.3.2 Methods

Thread objects are not instantiated directly, but instead through the appropriate *py420chan.Board* methods such as *py420chan.Board.get_thread()*.

`Thread.`**`files`**`()`
Returns the URLs of all files attached to posts in the thread.

`Thread.`**`thumbs`**`()`
Returns the URLs of all thumbnails in the thread.

`Thread.`**`filenames`**`()`
Returns the filenames of all files attached to posts in the thread.

`Thread.`**`thumbnames`**`()`
Returns the filenames of all thumbnails in the thread.

`Thread.`**`update`**`(`*force=False*`)`
Fetch new posts from the server.

Parameters **`force`** (*bool*) – Force a thread update, even if thread has 404'd.

Returns How many new posts have been fetched.

Return type int

2.4 `py420chan.Post` – 420chan Post

py420chan.Post allows for standard access to a 420chan post.

2.4.1 Example

Here is a sample application that grabs and prints *py420chan.Thread* and *py420chan.Post* information:

```python
# credits to Anarov for improved example
from __future__ import print_function
import py420chan

# get the board we want
board = py420chan.Board('v')

# select the first thread on the board
all_thread_ids = board.get_all_thread_ids()
first_thread_id = all_thread_ids[0]
thread = board.get_thread(first_thread_id)

# print thread information
print(thread)
print('Sticky?', thread.sticky)
print('Closed?', thread.closed)
print('Replies:', len(thread.replies))

# print topic post information
topic = thread.topic
print('Topic Repr', topic)
print('Postnumber', topic.post_number)
print('Timestamp', topic.timestamp)
```

(continues on next page)

(continued from previous page)

```
print('Datetime', repr(topic.datetime))
print('Subject', topic.subject)
print('Comment', topic.comment)

# file information
for f in first_thread.file_objects():
    print('Filename', f.filename)
    print('  Filemd5hex', f.file_md5_hex)
    print('  Fileurl', f.file_url)
    print('  Thumbnailurl', f.thumbnail_url)
    print()
```

2.4.2 Basic Usage

class py420chan.**Post**(*thread*, *data*)

Represents a post.

post_id

ID of this post. Eg: `123123123`, `456456456`.

Type int

poster_id

Poster ID.

Type int

name

Poster's name.

Type string

email

Poster's email.

Type string

tripcode

Poster's tripcode.

Type string

subject

Subject of this post.

Type string

comment

This comment, with the <wbr> tag removed.

Type string

html_comment

Original, direct HTML of this comment.

Type string

text_comment

Plaintext version of this comment.

Type string

is_op
Whether this is the OP (first post of the thread)

Type bool

timestamp
Unix timestamp for this post.

Type int

datetime
Datetime time of this post.

Type `datetime.datetime`

first_file
The File object associated with this post.

Type `py8chan.File`

has_file
Whether this post has a file attached to it.

Type bool

url
URL of this post.

Type string

semantic_url
URL of this post, with the thread's 'semantic' component.

Type string

semantic_slug
This post's 'semantic slug'.

Type string

Post objects are not instantiated directly, but through a *py420chan.Thread* object with an attribute like *py420chan.Thread.all_posts*.

2.5 py420chan.File – 420chan File

py420chan.Post allows for standard access to a 420chan file. This provides programs with a complete File object that contains all metadata about the 420chan file, and makes migration easy if 420chan ever makes multiple files in one Post possible (as 8chan does).

2.5.1 Basic Usage

class py420chan.**File**(*post*, *data*)
Represents File objects and their thumbnails.

Constructor: post (py4chan.Post) - parent Post object. data (dict) - The post or extra_files dict from the 8chan API.

filename
Original name of the file attached to this post.

Type string

`file_url`
URL of the file attached to this post.

Type string

`file_extension`
Extension of the file attached to this post. Eg: `png`, `webm`, etc.

Type string

`file_size`
Size of the file attached to this post.

Type int

`file_width`
Width of the file attached to this post.

Type int

`file_height`
Height of the file attached to this post.

Type int

`file_deleted`
Whether the file attached to this post was deleted after being posted.

Type bool

`thumbnail_width`
Width of the thumbnail attached to this post.

Type int

`thumbnail_height`
Height of the thumbnail attached to this post.

Type int

`thumbnail_fname`
Filename of the thumbnail attached to this post.

Type string

`thumbnail_url`
URL of the thumbnail attached to this post.

Type string

File objects are not instantiated directly, but through a *`py420chan.File`* object with an attribute like *`py420chan.Post.first_file`*.

Python Module Index

p

py420chan, 7

Index

Symbols

__init__() (py420chan.Board method), 8

A

all_posts (py420chan.Thread attribute), 10

B

Board (class in py420chan), 8

C

clear_cache() (py420chan.Board method), 10
closed (py420chan.Thread attribute), 10
comment (py420chan.Post attribute), 12

D

datetime (py420chan.Post attribute), 13

E

email (py420chan.Post attribute), 12

F

File (class in py420chan), 13
file_deleted (py420chan.File attribute), 14
file_extension (py420chan.File attribute), 14
file_height (py420chan.File attribute), 14
file_size (py420chan.File attribute), 14
file_url (py420chan.File attribute), 13
file_width (py420chan.File attribute), 14
filename (py420chan.File attribute), 13
filenames() (py420chan.Thread method), 11
files() (py420chan.Thread method), 11
first_file (py420chan.Post attribute), 13

G

get_all_boards() (in module py420chan), 7
get_all_thread_ids() (py420chan.Board method), 9
get_all_threads() (py420chan.Board method), 9
get_boards() (in module py420chan), 7
get_thread() (py420chan.Board method), 9
get_threads() (py420chan.Board method), 9

H

has_file (py420chan.Post attribute), 13
html_comment (py420chan.Post attribute), 12

I

is_op (py420chan.Post attribute), 12
is_worksafe (py420chan.Board attribute), 8

N

name (py420chan.Board attribute), 8
name (py420chan.Post attribute), 12

P

page_count (py420chan.Board attribute), 8
Post (class in py420chan), 12
post_id (py420chan.Post attribute), 12
poster_id (py420chan.Post attribute), 12
posts (py420chan.Thread attribute), 10
py420chan (module), 7

R

refresh_cache() (py420chan.Board method), 10

S

semantic_slug (py420chan.Post attribute), 13
semantic_slug (py420chan.Thread attribute), 10
semantic_url (py420chan.Post attribute), 13
semantic_url (py420chan.Thread attribute), 10
sticky (py420chan.Thread attribute), 10
subject (py420chan.Post attribute), 12

T

text_comment (py420chan.Post attribute), 12
Thread (class in py420chan), 10
thread_exists() (py420chan.Board method), 9
threads_per_page (py420chan.Board attribute), 8

thumbnail_fname (py420chan.File attribute), 14
thumbnail_height (py420chan.File attribute), 14
thumbnail_url (py420chan.File attribute), 14
thumbnail_width (py420chan.File attribute), 14
thumbnames() (py420chan.Thread method), 11
thumbs() (py420chan.Thread method), 11
timestamp (py420chan.Post attribute), 13
title (py420chan.Board attribute), 8
topic (py420chan.Thread attribute), 10
tripcode (py420chan.Post attribute), 12

U

update() (py420chan.Thread method), 11
url (py420chan.Post attribute), 13
url (py420chan.Thread attribute), 10