py-trello Documentation

Release 0.2.3

Adrien Lemaire, Kyle Valade, Rick van Hattem

Jun 30, 2017

Contents

CHAPTER 1

Modules

trello package

Submodules

trello.attachments module

class trello.attachments.**Attachments**(*id*, *bytes*, *date*, *edge_color*, *idMember*, *is_upload*, *mime_type*, *name*, *previews*, *url*)

Bases: *trello.base.TrelloBase*

https://developers.trello.com/advanced-reference/card#get-1-cards-card-id-or-shortlink-attachments

static from_json(*json_obj*)

class trello.attachments.**AttachmentsPreview**(*bytes*, *url*, *width*, *height*, *is_scaled*)

Bases: object

static from_json(*json_obj*)

trello.base module

class trello.base.**TrelloBase**

Bases: object

trello.board module

class trello.board.**Board**(*client=None*, *board_id=None*, *organization=None*, *name=''*)

Bases: *trello.base.TrelloBase*

Class representing a Trello board. Board attributes are stored as normal Python attributes; access to all sub-objects, however, is always an API call (Lists, Cards).

add_label(*name*, *color*)
Add a label to this board

Name name of the label

Color the color, either green, yellow, orange red, purple, blue, sky, lime, pink, or black

Returns the label

Return type *Label*

add_list(*name*, *pos=None*)
Add a list to this board

Name name for the list

Pos position of the list: "bottom", "top" or a positive number

Returns the list

Return type *List*

add_member(*member*, *member_type='normal'*)

admin_members()
Returns all admin members on this board

Return type list of Member

all_cards()
Returns all cards on this board

Return type list of Card

all_lists()
Returns all lists on this board

Return type list of List

all_members()
Returns all members on this board

Return type list of Member

close()

closed_cards()
Returns all closed cards on this board

Return type list of Card

closed_lists()
Returns all closed lists on this board

Return type list of List

fetch()
Fetch all attributes for this board

fetch_actions(*action_filter*, *action_limit=50*, *before=None*, *since=None*)
Returns all actions that conform to the given filters.

Action_filter str of possible actions separated by comma ie. 'createCard,updateCard'

Action_limit int of max items returned

Before datetime obj

Since datetime obj

More info on action filter values: https://developers.trello.com/advanced-reference/board#get-1-boards-board-id-actions

Return type json list of past actions

classmethod **from_json**(*trello_client=None*, *organization=None*, *json_obj=None*)
Deserialize the board json object to a Board object

Trello_client the trello client

Json_obj the board json object

Alternative contrustraction:

Deserialize the board json object to a board object

Organization the organization object that the board belongs to

Json_obj the json board object

get_cards(*filters=None*, *card_filter=''*)

Filters dict containing query parameters. Eg. {'fields': 'all'}

Card_filter filters on card status ('open', 'closed', 'all')

More info on card queries: https://trello.com/docs/api/board/index.html#get-1-boards-board-id-cards

Return type list of Card

get_checklists(*cards='all'*)
Get checklists

Return type list of Checklist

get_labels(*fields='all'*, *limit=50*)
Get label

Return type list of Label

get_last_activity()
Return the date of the last action done on the board.

Return type datetime.datetime

get_list(*list_id*)
Get list

Return type *List*

get_lists(*list_filter*)
Get lists from filter

Return type list of List

get_members(*filters=None*)
Get members with filter

Filters dict containing query parameters. Eg. {'fields': 'all', 'filter': 'admins'}

More info on possible filters: https://developers.trello.com/advanced-reference/board#get-1-boards-board-id-members

Return type list of Member

list_lists(*list_filter='all'*)
Get lists from filter

Return type list of List

normal_members()
Returns all normal members on this board

Return type list of Member

open()

open_cards()
Returns all open cards on this board

Return type list of Card

open_lists()
Returns all open lists on this board

Return type list of List

owner_members()
Returns all owner members on this board

Return type list of Member

remove_member(*member*)

save()

set_description(*desc*)

set_name(*name*)

trello.card module

class trello.card.**Card**(*parent*, *card_id*, *name=''*)
Bases: *trello.base.TrelloBase*

Class representing a Trello card. Card attributes are stored on the object

https://developers.trello.com/advanced-reference/card

add_checklist(*title*, *items*, *itemstates=None*)
Add a checklist to this card

Title title of the checklist

Items a list of the item names

Itemstates a list of the state (True/False) of each item

Returns the checklist

add_label(*label*)

add_member(*member*)

assign(*member_id*)

attach(*name=None*, *mimeType=None*, *file=None*, *url=None*)
Add an attachment to the card. The attachment can be either a file or a url. Setting the name and/or mime type is optional. :param name: The name of the attachment :param mimeType: mime type for the attachement :param file: a file-like, binary object that supports read() :param url: a URL pointing to the resource to be attached

attachments
Lazily loads and returns the attachments

attriExp(*multiple*)
Provides the option to explore what comes from trello :multiple is one of the attributes of GET /1/cards/[card id or shortlink]/actions

board_id

card_created_date
Will return the creation date of the card.

NOTE: This will return the date the card was created, even if it was created on another board. The created_date() above actually just returns the first activity and has the issue described in the warning.

The first 8 characters of the card id is a hexadecimal number. Converted to a decimal from hexadecimal, the timestamp is an Unix timestamp (the number of seconds that have elapsed since January 1, 1970 midnight UTC. See http://help.trello.com/article/759-getting-the-time-a-card-or-board-was-created

change_board(*board_id*, *list_id=None*)

change_list(*list_id*)

change_pos(*position*)

checklists
Lazily loads and returns the checklists

comment(*comment_text*)
Add a comment to a card.

Comment_text str

comments
Lazily loads and returns the comments

created_date
Will return the creation date of the card.

WARNING: if the card was create via convertion of a checklist item it fails. attriExp('convertToCardFromCheckItem') allows to test for the condition.

date_last_activity

delete()

delete_comment(*comment*)

description

due_date

fetch(*eager=True*)
Fetch all attributes for this card

Parameters **eager** – If eager, comments, checklists and attachments will be fetched immediately, otherwise on demand

fetch_actions(*action_filter='createCard'*, *since=None*, *before=None*)
Fetch actions for this card can give more argv to action_filter, split for ',' json_obj is list

fetch_attachments(*force=False*)

fetch_checklists()

fetch_comments(*force=False*, *limit=None*)

fetch_plugin_data()

classmethod from_json(*parent*, *json_obj*)
Deserialize the card json object to a Card object

Parent the list object that the card belongs to

Json_obj json object

Return type *Card*

get_attachments()

get_comments()
Alias for fetch_comments for backward compatibility. Always contact server

get_list()

get_stats_by_list(*lists*, *list_cmp=None*, *done_list=None*, *time_unit='seconds'*, *card_movements_filter=None*)
Gets several stats about the card by each list of the board: - time: The time that the card has been in each column in seconds (minutes or hours). - forward_moves: How many times this card has been the source of a forward movement. - backward_moves: How many times this card has been the source of a backward movement.

Returns a dict where the key is list id and value is a dict with keys time, forward_moves and backward_moves.

Parameters

- **lists** – list of board lists.
- **list_cmp** – function that compares two lists a,b given id_a, id_b. If b is in a forward position returns 1 else -1.
- **time_unit** – default to seconds. Allow specifying time in "minutes" or "hours".
- **done_list** – Column that implies that the task is done. If present, time measurement will be stopped if is current task list.
- **card_movements_filter** – Pair of two dates (two strings in YYYY-MM-DD format) that will filter the movements of the card. Optional.

Returns dict of the form {list_id: {time:<time card was in that list>, forward_moves: <number>, backward_moves: <number> }}

idLabels

latestCardMove_date
Returns the date of the last card transition

listCardMove_date()
Will return the history of transitions of a card from one list to another. The lower the index the more resent the historical item.

It returns a list of lists. The sublists are triplets of starting list, ending list and when the transition occurred.

list_id

list_labels

list_movements(*list_cmp=None*, *filter_by_date_interval=None*)
Will return the history of transitions of a card from one list to another. The lower the index the more resent the historical item.

It returns a list of dicts in date and time descending order (the first movement is the earliest). Dicts are of the form source: <listobj> destination: <listobj> datetime: <datetimeobj>

Param list_cmp Comparison function between lists. For list_cmp(a, b) returns -1 if list a is greater that list b. Returns 1 otherwise.

Param filter_by_date_interval: pair of two dates (two strings in YYYY-MM-DD format) to filter card movements by date.

member_id

plugin_data
Lazily loads and returns the plugin data

remove_attachment(*attachment_id*)
Remove attachment from card :param attachment_id: Attachment id :return: None

remove_due()
Remove the due datetime of this card.

remove_due_complete()
Remove due complete

Returns None

remove_label(*label*)

remove_member(*member*)

set_closed(*closed*)

set_description(*description*)

set_due(*due*)
Set the due time for the card

Due a datetime object

set_due_complete()
Set due complete

Returns None

set_name(*new_name*)
Update the name on the card to :new_name:

New_name str

set_pos(*pos*)
Update card position in list

Pos 'top', 'bottom' or int

short_id

short_url

subscribe()

unassign(*member_id*)

update_comment(*comment_id*, *comment_text*)
Update a comment.

trello.checklist module

class trello.checklist.**Checklist**(*client*, *checked*, *obj*, *trello_card=None*)
Bases: *trello.base.TrelloBase*

Class representing a Trello checklist.

add_checklist_item(*name*, *checked=False*)
Add a checklist item to this checklist

Name name of the checklist item

Checked True if item state should be checked, False otherwise

Returns the checklist item json object

clear()
Clear checklist by removing all checklist items

delete()
Removes this checklist

delete_checklist_item(*name*)
Delete an item on this checklist

Name name of the checklist item to delete

rename(*new_name*)
Rename this checklist

New_name new name of the checklist

rename_checklist_item(*name*, *new_name*)
Rename the item on this checklist

Name name of the checklist item

New_name new name of item

set_checklist_item(*name*, *checked*)
Set the state of an item on this checklist

Name name of the checklist item

Checked True if item state should be checked, False otherwise

trello.compat module

trello.compat.**force_str**(*s*, *encoding='utf-8'*)
Converts *s* to the *str* type, regardless of the Python version. This is useful for __repr__ return types, where a *str* (bytes) is expected in Python 2 and a *str* (unicode string) is expected in Python 3.

trello.exceptions module

exception trello.exceptions.**ResourceUnavailable**(*msg*, *http_response*)
Bases: exceptions.Exception

Exception representing a failed request to a resource

exception trello.exceptions.**TokenError**
Bases: exceptions.Exception

exception trello.exceptions.**Unauthorized**(*msg*, *http_response*)
Bases: *trello.exceptions.ResourceUnavailable*

trello.label module

class trello.label.**Label**(*client*, *label_id*, *name*, *color=''*)
Bases: *trello.base.TrelloBase*

Class representing a Trello Label.

fetch()
Fetch all attributes for this label

classmethod from_json(*board*, *json_obj*)
Deserialize the label json object to a Label object

Board the parent board the label is on

Json_obj the label json object

classmethod from_json_list(*board*, *json_objs*)

trello.member module

class trello.member.**Member**(*client*, *member_id*, *full_name=''*)
Bases: *trello.base.TrelloBase*

Class representing a Trello member.

fetch()
Fetch all attributes for this member

fetch_cards()
Fetches all the cards for this member

fetch_comments()

fetch_notifications(*filters=[]*)
Fetches all the notifications for this member

classmethod from_json(*trello_client*, *json_obj*)
Deserialize the organization json object to a member object

Trello_client the trello client

Json_obj the member json object

trello.organization module

class trello.organization.**Organization**(*client*, *organization_id*, *name=''*)
Bases: *trello.base.TrelloBase*

TIMEZONE = None
Class representing an organization

all_boards()
Returns all boards on this organization

fetch()
Fetch all attributes for this organization

classmethod **from_json**(*trello_client*, *json_obj*)
Deserialize the board json object to a Organization object

Trello_client the trello client

Json_obj the board json object

get_board(*field_name*)
Get board

Return type list of Board

get_boards(*list_filter*)
Get boards using filter

Return type list of Board

get_members()

trello.trelloclient module

class trello.trelloclient.**TrelloClient**(*api_key*, *api_secret=None*, *token=None*, *token_secret=None*)
Bases: object

Base class for Trello API access

add_board(*board_name*, *source_board=None*, *organization_id=None*, *permission_level='private'*)
Create board :param board_name: Name of the board to create :param source_board: Optional Board to copy :param permission_level: Permission level, defaults to private :rtype: Board

create_hook(*callback_url*, *id_model*, *desc=None*, *token=None*)
Creates a new webhook. Returns the WebHook object created.

There seems to be some sort of bug that makes you unable to create a hook using httplib2, so I'm using urllib2 for that instead.

fetch_json(*uri_path*, *http_method='GET'*, *headers=None*, *query_params=None*, *post_args=None*, *files=None*)
Fetch some JSON from Trello

get_board(*board_id*)
Get board

Return type *Board*

get_card(*card_id*)
Get card

Return type *Card*

get_label(*label_id*, *board_id*)
Get Label

Requires the parent board id the label is on

Return type *Label*

get_list(*list_id*)
Get list

Return type *List*

get_member(*member_id*)
Get member

Return type *Member*

get_organization(*organization_id*)
Get organization

Return type *Organization*

info_for_all_boards(*actions*)
Use this if you want to retrieve info for all your boards in one swoop

list_boards(*board_filter='all'*)
Returns all boards for your Trello user

Returns a list of Python objects representing the Trello boards.

Return type list of Board

Each board has the following noteworthy attributes:

- id: the board's identifier
- name: Name of the board
- **desc: Description of the board (optional - may be missing from the** returned JSON)
- closed: Boolean representing whether this board is closed or not
- url: URL to the board

list_hooks(*token=None*)
Returns a list of all hooks associated with a specific token. If you don't pass in a token, it tries to use the token associated with the TrelloClient object (if it exists)

list_organizations()
Returns all organizations for your Trello user

Returns a list of Python objects representing the Trello organizations.

Return type list of Organization

Each organization has the following noteworthy attributes:

- id: the organization's identifier
- name: Name of the organization
- **desc: Description of the organization (optional - may be missing from the** returned JSON)
- closed: Boolean representing whether this organization is closed or not
- url: URL to the organization

logout()
Log out of Trello.

search(*query*, *partial_match=False*, *models=[]*, *board_ids=[]*, *org_ids=[]*, *card_ids=[]*)
Search trello given a query string.

Parameters

- **query** (*str*) – A query string up to 16K characters

- **partial_match** (*bool*) – True means that trello will look for content that starts with any of the words in your query.
- **models** (*list*) – Comma-separated list of types of objects to search. This can be 'actions', 'boards', 'cards', 'members', or 'organizations'. The default is 'all' models.
- **board_ids** (*list*) – Comma-separated list of boards to limit search
- **org_ids** – Comma-separated list of organizations to limit search
- **card_ids** – Comma-separated list of cards to limit search

Returns All objects matching the search criterial. These can be Cards, Boards, Organizations, and Members. The attributes of the objects in the results are minimal; the user must call the fetch method on the resulting objects to get a full set of attributes populated.

Rtype list

trello.trellolist module

class trello.trellolist.**List**(*board*, *list_id*, *name=''*)
Bases: *trello.base.TrelloBase*

Class representing a Trello list. List attributes are stored on the object, but access to sub-objects (Cards) require an API call

add_card(*name*, *desc=None*, *labels=None*, *due='null'*, *source=None*, *position=None*, *assign=None*)
Add a card to this list

Name name for the card

Desc the description of the card

Labels a list of label IDs to be added

Due due date for the card

Source card ID from which to clone from

Position position of the card in the list. Must be "top", "bottom" or a positive number.

Returns the card

archive_all_cards()

cardsCnt()

close()

fetch()
Fetch all attributes for this list

fetch_actions(*action_filter*)
Fetch actions for this list can give more argv to action_filter, split for ',' json_obj is list

classmethod from_json(*board*, *json_obj*)
Deserialize the list json object to a List object

Board the board object that the list belongs to

Json_obj the json list object

list_cards(*card_filter='open'*, *actions=None*)
Lists all cards in this list

move(*position*)

move_all_cards(*destination_list*)
Move all cards of this list to another list. The list can be in the same board (or not).

open()

set_name(*name*)

set_pos(*position*)

trello.util module

trello.util.**create_oauth_token**(*expiration=None*, *scope=None*, *key=None*, *secret=None*, *name=None*, *output=True*)
Script to obtain an OAuth token from Trello.

Must have TRELLO_API_KEY and TRELLO_API_SECRET set in your environment To set the token's expiration, set TRELLO_EXPIRATION as a string in your environment settings (eg. 'never'), otherwise it will default to 30 days.

More info on token scope here: https://trello.com/docs/gettingstarted/#getting-a-token-from-a-user

trello.webhook module

class trello.webhook.**WebHook**(*client*, *token*, *hook_id=None*, *desc=None*, *id_model=None*, *callback_url=None*, *active=False*)
Bases: object

Class representing a Trello webhook.

delete()
Removes this webhook from Trello

Module contents

CHAPTER 2

Readme

A wrapper around the Trello API written in Python. Each Trello object is represented by a corresponding Python object. The attributes of these objects are cached, but the child objects are not. This can possibly be improved when the API allows for notification subscriptions; this would allow caching (assuming a connection was available to invalidate the cache as appropriate).

I've created a Trello Board for feature requests, discussion and some development tracking.

CHAPTER 3

Install

```
pip install py-trello
```

CHAPTER 4

Usage

```
from trello import TrelloClient

client = TrelloClient(
    api_key='your-key',
    api_secret='your-secret',
    token='your-oauth-token-key',
    token_secret='your-oauth-token-secret'
)
```

Where `token` and `token_secret` come from the 3-legged OAuth process and `api_key` and `api_secret` are your Trello API credentials that are (generated here).

CHAPTER 5

Getting your Trello OAuth Token

Make sure the following environment variables are set:

- TRELLO_API_KEY
- TRELLO_API_SECRET

These are obtained from the link mentioned above.

TRELLO_EXPIRATION is optional. Set it to a string such as 'never' or '1day'. Trello's default OAuth Token expiration is 30 days.

Default permissions are read/write.

More info on setting the expiration here: https://trello.com/docs/gettingstarted/#getting-a-token-from-a-user

Run

```
python ./trello/util.py
```

CHAPTER 6

Required Python modules

Found in `requirements.txt`

CHAPTER 7

Tests

To run the tests, run `python -m unittest discover`. Four environment variables must be set:

- `TRELLO_API_KEY`: your Trello API key
- `TRELLO_TOKEN`: your Trello OAuth token
- `TRELLO_TEST_BOARD_COUNT`: the number of boards in your Trello account
- `TRELLO_TEST_BOARD_NAME`: name of the board to test card manipulation on. Must be unique, or the first match will be used

To run tests across various Python versions, tox is supported. Install it and simply run `tox` from the `py-trello` directory.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index

G

I

L

M

N

O

P

R

S

T

U

W