todoist-python Documentation

Release 1.0

Doist

Aug 16, 2018

Contents

The official Todoist Python API library.

CHAPTER 1

Modules

1.1 todoist

1.1.1 todoist.api

exception todoist.api.**SyncError**
Bases: exceptions.Exception

class todoist.api.**TodoistAPI**(*token=''*, *api_endpoint='https://todoist.com'*, *session=None*, *cache='~/.todoist-sync/'*)
Bases: object

Implements the API that makes it possible to interact with a Todoist user account and its data.

classmethod deserialize(*data*)

reset_state()

serialize()

get_api_url()

generate_uuid()
Generates a uuid.

sync(*commands=None*)
Sends to the server the changes that were made locally, and also fetches the latest updated data from the server.

commit(*raise_on_error=True*)
Commits all requests that are queued. Note that, without calling this method none of the changes that are made to the objects are actually synchronized to the server, unless one of the aforementioned Sync API calls are called directly.

query(*queries*, ***kwargs*)
DEPRECATED: query endpoint is deprecated for a long time and this method will be removed in the next major version of todoist-python

add_item(*content*, ***kwargs*)
Adds a new task.

todoist.api.**state_default**(*obj*)

todoist.api.**json_default**(*obj*)

1.1.2 todoist.models

class todoist.models.**Model**(*data*, *api*)
Bases: object

Implements a generic object.

class todoist.models.**Collaborator**(*data*, *api*)
Bases: *todoist.models.Model*

Implements a collaborator.

delete(*project_id*)
Deletes a collaborator from a shared project.

class todoist.models.**CollaboratorState**(*data*, *api*)
Bases: *todoist.models.Model*

Implements a collaborator state.

class todoist.models.**Filter**(*data*, *api*)
Bases: *todoist.models.Model*

Implements a filter.

update(***kwargs*)
Updates filter.

delete()
Deletes filter.

class todoist.models.**Item**(*data*, *api*)
Bases: *todoist.models.Model*

Implements an item.

update(***kwargs*)
Updates item.

delete()
Deletes item.

move(*to_project*)
Moves item to another project.

close()
Marks item as closed

complete(*force_history=0*)
Marks item as completed.

uncomplete(*update_item_orders=1*, *restore_state=None*)
Marks item as not completed.

update_date_complete(*new_date_utc=None*, *date_string=None*, *is_forward=None*)
Completes a recurring task.

class todoist.models.**Label**(*data*, *api*)
Bases: *todoist.models.Model*

Implements a label.

update(***kwargs*)
Updates label.

delete()
Deletes label.

class todoist.models.**LiveNotification**(*data*, *api*)
Bases: *todoist.models.Model*

Implements a live notification.

class todoist.models.**GenericNote**(*data*, *api*)
Bases: *todoist.models.Model*

Implements a note.

local_manager = None
has to be defined in subclasses

update(***kwargs*)
Updates note.

delete()
Deletes note.

class todoist.models.**Note**(*data*, *api*)
Bases: *todoist.models.GenericNote*

Implement an item note.

class todoist.models.**ProjectNote**(*data*, *api*)
Bases: *todoist.models.GenericNote*

Implement a project note.

class todoist.models.**Project**(*data*, *api*)
Bases: *todoist.models.Model*

Implements a project.

update(***kwargs*)
Updates project.

delete()
Deletes project.

archive()
Marks project as archived.

unarchive()
Marks project as not archived.

share(*email*, *message=''*)
Shares projects with a user.

take_ownership()
Takes ownership of a shared project.

class todoist.models.**Reminder**(*data*, *api*)
Bases: *todoist.models.Model*

Implements a reminder.

update(***kwargs*)
Updates reminder.

delete()
Deletes reminder.

1.1.3 todoist.managers

class todoist.managers.projects.**ProjectsManager**(*api*)
Bases: *todoist.managers.generic.Manager*, *todoist.managers.generic.AllMixin*, *todoist.managers.generic.GetByIdMixin*, *todoist.managers.generic.SyncMixin*

state_name = 'projects'

object_type = 'project'

add(*name*, ***kwargs*)
Creates a local project object.

update(*project_id*, ***kwargs*)
Updates a project remotely.

delete(*project_ids*)
Deletes a project remotely.

archive(*project_id*)
Marks project as archived remotely.

unarchive(*project_id*)
Marks project as not archived remotely.

update_orders_indents(*ids_to_orders_indents*)
Updates the orders and indents of multiple projects remotely.

share(*project_id*, *email*, *message=''*)
Shares a project with a user.

get_archived()
Returns archived projects.

get_data(*project_id*)
Returns a project's uncompleted items.

get(*project_id*)
Gets an existing project.

class todoist.managers.items.**ItemsManager**(*api*)
Bases: *todoist.managers.generic.Manager*, *todoist.managers.generic.AllMixin*, *todoist.managers.generic.GetByIdMixin*, *todoist.managers.generic.SyncMixin*

state_name = 'items'

object_type = 'item'

add(*content*, *project_id*, ***kwargs*)
Creates a local item object.

update(*item_id*, ***kwargs*)
Updates an item remotely.

delete(*item_ids*)
Deletes items remotely.

move(*project_items*, *to_project*)
Moves items to another project remotely.

close(*item_id*)
Marks item as done

complete(*item_ids*, *force_history=0*)
Marks items as completed remotely.

uncomplete(*item_ids*, *update_item_orders=1*, *restore_state=None*)
Marks items as not completed remotely.

update_date_complete(*item_id*, *new_date_utc=None*, *date_string=None*, *is_forward=None*)
Completes a recurring task remotely.

update_orders_indents(*ids_to_orders_indents*)
Updates the order and indents of multiple items remotely.

update_day_orders(*ids_to_orders*)
Updates in the local state the day orders of multiple items remotely.

get_completed(*project_id*, ***kwargs*)
Returns a project's completed items.

get(*item_id*)
Gets an existing item.

class todoist.managers.labels.**LabelsManager**(*api*)
Bases: *todoist.managers.generic.Manager*, *todoist.managers.generic.AllMixin*, *todoist.managers.generic.GetByIdMixin*, *todoist.managers.generic.SyncMixin*

state_name = 'labels'

object_type = 'label'

add(*name*, ***kwargs*)
Creates a local label object.

update(*label_id*, ***kwargs*)
Updates a label remotely.

delete(*label_id*)
Deletes a label remotely.

update_orders(*id_order_mapping*)
Updates the orders of multiple labels remotely.

get(*label_id*)
Gets an existing label.

class todoist.managers.notes.**GenericNotesManager**(*api*)
Bases: *todoist.managers.generic.Manager*, *todoist.managers.generic.AllMixin*, *todoist.managers.generic.GetByIdMixin*, *todoist.managers.generic.SyncMixin*

object_type = 'note'

update(*note_id*, ***kwargs*)
Updates an note remotely.

delete(*note_id*)
Deletes an note remotely.

class todoist.managers.notes.**NotesManager**(*api*)
Bases: *todoist.managers.notes.GenericNotesManager*

state_name = 'notes'

add(*item_id*, *content*, ***kwargs*)
Creates a local item note object.

get(*note_id*)
Gets an existing note.

class todoist.managers.notes.**ProjectNotesManager**(*api*)
Bases: *todoist.managers.notes.GenericNotesManager*

state_name = 'project_notes'

add(*project_id*, *content*, ***kwargs*)
Creates a local project note object.

class todoist.managers.filters.**FiltersManager**(*api*)
Bases: *todoist.managers.generic.Manager*, *todoist.managers.generic.AllMixin*, *todoist.managers.generic.GetByIdMixin*, *todoist.managers.generic.SyncMixin*

state_name = 'filters'

object_type = 'filter'

add(*name*, *query*, ***kwargs*)
Creates a local filter object.

update(*filter_id*, ***kwargs*)
Updates a filter remotely.

delete(*filter_id*)
Deletes a filter remotely.

update_orders(*id_order_mapping*)
Updates the orders of multiple filters remotely.

get(*filter_id*)
Gets an existing filter.

class todoist.managers.reminders.**RemindersManager**(*api*)
Bases: *todoist.managers.generic.Manager*, *todoist.managers.generic.AllMixin*, *todoist.managers.generic.GetByIdMixin*, *todoist.managers.generic.SyncMixin*

state_name = 'reminders'

object_type = 'reminder'

add(*item_id*, ***kwargs*)
Creates a local reminder object.

update(*reminder_id*, ***kwargs*)
Updates a reminder remotely.

delete(*reminder_id*)
Deletes a reminder remotely.

get(*reminder_id*)
Gets an existing reminder.

class todoist.managers.locations.**LocationsManager**(*api*)
Bases: *todoist.managers.generic.Manager*, *todoist.managers.generic.AllMixin*, *todoist.managers.generic.SyncMixin*

state_name = 'locations'

object_type = None

clear()
Clears the locations.

class todoist.managers.user.**UserManager**(*api*)
Bases: *todoist.managers.generic.Manager*

update(***kwargs*)
Updates the user data.

update_goals(***kwargs*)
Updates the user's karma goals.

sync()

get(*key=None*, *default=None*)

get_id()

login(*email*, *password*)
Logins user, and returns the response received by the server.

login_with_google(*email*, *oauth2_token*, ***kwargs*)
Logins user with Google account, and returns the response received by the server.

register(*email*, *full_name*, *password*, ***kwargs*)
Registers a new user.

delete(*current_password*, ***kwargs*)
Deletes an existing user.

update_notification_setting(*notification_type*, *service*, *dont_notify*)
Updates the user's notification settings.

class todoist.managers.invitations.**InvitationsManager**(*api*)
Bases: *todoist.managers.generic.Manager*, *todoist.managers.generic.SyncMixin*

state_name = None

object_type = 'share_invitation'

accept(*invitation_id*, *invitation_secret*)
Accepts an invitation to share a project.

reject(*invitation_id*, *invitation_secret*)
Rejets an invitation to share a project.

delete(*invitation_id*)
Delete an invitation to share a project.

class todoist.managers.collaborators.**CollaboratorsManager**(*api*)
Bases: *todoist.managers.generic.Manager*, *todoist.managers.generic.GetByIdMixin*, *todoist.managers.generic.SyncMixin*

state_name = 'collaborators'

object_type = None

delete(*project_id*, *email*)
Deletes a collaborator from a shared project.

class todoist.managers.collaborator_states.**CollaboratorStatesManager**(*api*)
Bases: *todoist.managers.generic.Manager*, *todoist.managers.generic.SyncMixin*

state_name = 'collaborator_states'

object_type = None

get_by_ids(*project_id*, *user_id*)
Finds and returns the collaborator state based on the project and user ids.

class todoist.managers.biz_invitations.**BizInvitationsManager**(*api*)
Bases: *todoist.managers.generic.Manager*

state_name = None

object_type = None

accept(*invitation_id*, *invitation_secret*)
Accepts a business invitation to share a project.

reject(*invitation_id*, *invitation_secret*)
Rejects a business invitation to share a project.

class todoist.managers.live_notifications.**LiveNotificationsManager**(*api*)
Bases: *todoist.managers.generic.Manager*, *todoist.managers.generic.GetByIdMixin*, *todoist.managers.generic.AllMixin*, *todoist.managers.generic.SyncMixin*

state_name = 'live_notifications'

object_type = None

set_last_read(*id*)
Sets the last known notification.

mark_read(*id*)
Marks notification as read.

mark_read_all()
Marks all notifications as read.

mark_unread(*id*)
Marks notification as unread.

class todoist.managers.generic.**Manager**(*api*)
Bases: object

state_name = None

object_type = None

state

queue

token

class todoist.managers.generic.**AllMixin**
Bases: object

all(*filt=None*)

class todoist.managers.generic.**GetByIdMixin**
Bases: object

get_by_id(*obj_id*, *only_local=False*)
Finds and returns the object based on its id.

class todoist.managers.generic.**SyncMixin**
Bases: object

Syncs this specific type of objects.

sync()

CHAPTER 2

Indices and tables

- genindex
- modindex

Python Module Index

t

Index

I

J

L

M

N

O

P

Q

R

S

T

U