trolly Documentation

Release 0.2.1

Luke Rigby (plish) and Rick van Hattem (wolph)

March 10, 2016

Contents

Contents:

CHAPTER 1

trolly package

1.1 Module contents

1.2 Submodules

1.3 trolly.authorise module

1.4 trolly.trelloobject module

CHAPTER 2

Trolly

A Python wrapper around the Trello API. Provides a group of Python classes to represent Trello Objects. None of the classes cache values as they are designed to be inherited and extended to suit the needs of each user. Each class includes a basic set of methods based on general use cases. This library was based on work done by sarumont. Very little was kept from this code, but still props on the initial work.

CHAPTER 3

Changes

3.1 0.2

Trolly has recently been updated to start the deprecation of non-pythonic conventions and improve PEP8 compliance. All the old pascal cased methods are now deprecated, they have been left in and point to the new method names. There are no breaking changes in this release but it is recommended you change your code to use the pythonic naming conventions.

3.2 Getting Started

3.2.1 Dependencies

This library requires python 2.5+ or 3+.

Before getting stated with this library you will need a few extra things: - httplib2 - An API key for your Trello user - User authorisation token (see below for how to obtain)

3.2.2 Installation

To install for python 2 or 3 you can either download the source and run:

```
sudo python setup.py install
```

Alternatively you can use:

```
sudo pip install Trolly
```

3.2.3 Authorisation

User Authorisation Token

A user authorisation token isn't too hard to get hold of. There are instruction on how to get one on the Trello Documentation. For those too lazy there is a python class in the library called Authorise(). To use this class simply navigate to authorise.py in a terminal and type:

```
python authorise.py -a API_KEY APPLICATION_NAME WHEN_TO_EXPIRE
```

The API key and application names are required but the "WHEN_TO_EXPIRE" will default to 1day if not specified. Running this file will return a URL. Copy and paste it into your browser and away you go. You might want to store this somewhere for future use, especially if you have set it to never expire.

Oauth

This library (currently) has no Oauth support however the code this was based on includes Oauth support. So for inspiration on how to extend the Client class to include this check out the link above.

3.3 Overview

There are a number of methods for each of the Trello objects. Some accept query parameters, these are for API methods that will accept a wide range of values and return a lot of information. For example GET Boards will take a lot of query parameters to allow you to whittle down the information to the bar minimum. This is extremely useful for extending the classes without much extra programming.

3.3.1 Trello Client

This class holds the bulk of all the methods for communicating with the trello API and returning the Trello objects. A client instance is required by every Trello object, because of this it makes extending and overiding methods in this class very effective. This is where you would override the creating of an object, e.g. a Card, with your own object.

Example usage of the client:

```
client = trolly.client.Client(settings.API_KEY, settings.TOKEN)

print('Member: %s' % client.get_member())

print('Organisations:')
for organisation in client.get_organisations():
    print(' - %s' % organisation)

print('Organisations:')
for organisation in client.get_organisations():
    print(' - %s' % organisation)

print('Boards:')
for board in client.get_boards():
    print(' - %s' % board)

print('Cards:')
for card in client.get_cards():
    print(' - %s' % card)
```

3.3.2 Trello Object

This class is inhereted my all Trello object classes (Board, List, Card, etc). The class takes only one argument, a Trello client instance. It masks calls to the client as belonging to the class (for the purposes of modularity). There are also a number of methods for fetching (Board, List, etc) JSON from the API. Since the only thing that differs from class to class is the base URI, this is taken as the only argument.

There are a number of methods for creating Trello objects. The createOBJECTNAME methods in this class can be extended easily by passing them keyword arguments.

3.3.3 Extending Trello Classes

Extending these classes is the premise on which they were built. Below outlines an example of how this can be acheived.

If for example we wanted to pass extra variables to our a Trello Card object then we can do the below:

```
class MyList( List ):

    def __init__( self, trello_client, list_id, name = '' ):
        super( MyList, self ).__init__( trello_client, list_id, name )

    def getCards( self ):
        cards = self.getCardsJson( self.base_uri )
        return self.createCard( card_json = cards[0], test = 'this is a test argument' )
```

This MyList class overrides the getCards method to add the extra variable we need. This would need to be done to any Trello object that will return a custom card.

We declare and pass the extra ('test') variable as a keyword argument here. We then need to extend the card class to allow for the extra variables:

```
class MyCard( Card ):

    def __init__( self, trello_client, card_id, test, name = '' ):
        super( MyCard, self ).__init__( trello_client, card_id, name )
        self.test_arg = test
```

Finally, we extend and override the Client. Overriding the client means that any object that calls createCard will create one of our new client classes.

```
class MyClient( Client ):

    def __init__( self, api_key, user_auth_token ):
        super( MyClient, self ).__init__( api_key, user_auth_token )

    def createCard( self, card_json, **kwargs ):

        return MyCard(
                trello_client = self,
                card_id = card_json['id'],
                name = card_json['name'],
                test = kwargs['test']
            )
```

The above client will fail though if you fail to pass a "test" keyword argument. To get around this you could use:

```
kwargs.get('test',"default value")
```

This will help avoid a value not being passed. You could also, instead of extending the object creation, add a method to cache the details you want using the objects getObjectInformation method.

Hope this helps and happy Trelloing!

3.4 Running Test

In order to run the tests you will need: - API Key - User authorisation token - An organisation ID - A board ID - A list ID - A card ID - A checklist ID - A member ID

It's quite a lot of information to get hold of (sorry). If you don't need everything you can just comment out the tests you don't need.

To run the tests navigate to the Trolly in a terminal and run:

```
PYTHONPATH=. python test/tests.py
```

3.5 Licence

This code is licenced under the MIT Licence

CHAPTER 4

Indices and tables

- genindex
- modindex
- search