py-basecamp Documentation

Release 0.0.1a

Greg Aker, nGen Works

December 22, 2016

Contents

Basecamp is a wrapper around the Basecamp Next API and makes use of the Requests library made by Kenneth Reitz.

The py-basecamp source code is hosted on GitHub: https://github.com/ngenworks/py-basecamp/

Documentation can be found at Read the Docs: http://basecamp.readthedocs.org/en/latest/index.html

Test coverage results can be found at https://secure.travis-ci.org/#!/ngenworks/py-basecamp

CHAPTER 1

Automatic Installation

Install the master branch from GitHub

```
pip -e git+git://github.com/ngenworks/py-basecamp.git#egg=basecamp
```

CHAPTER 2

Manual Installation

Download: https://github.com/ngenworks/py-basecamp/tarball/master

```
tar zxvf py-basecamp.tgz
cd py-basecamp
python setup.py install
```

sudo may be needed to install in the system-wide Python installation. Using in a virtualenv is recommended.

CHAPTER 3

Contents

3.1 Auth

The Basecamp API follows draft 5 of the oAuth 2 spec

In short, this is how it works:

- Ask for access
- A user authenticates with their Basecamp account
- Get a verification code.
- Trade that code in for an access token.
- Start performing authenticated requests with said token.

3.1.1 Basic usage

```
>>> import basecamp.api
>>> auth = basecamp.api.Auth(client_url, client_secret, redirect_url)
>>> launchpad_url = auth.launchpad_url
```

Redirect to the `launchpad_url` in your application after the user authenticates, they are redirected back to the redirect_url location, and a *code* GET variable will be present to exchange for a token.

```
>>> import basecamp.api
>>> auth = basecamp.api.Auth(client_url, client_secret, redirect_url)
>>> token = auth.get_token()
```

3.1.2 Examples

Here's a basic example of how this could work in a Flask application.

```
import basecamp.api
from secrets import client_id, client_secret, return_url
from flask import Flask, redirect, request

app = Flask(__name__)

@app.route('/basecamp-login/')
def basecamp_login():
```

```
    '''
    Redirect user to basecamp to authenticate.
    '''
    auth = basecamp.api.Auth(client_id, client_secret, return_url)

    return redirect(auth.launchpad_url)

@app.route('/auth-return/')
def auth_return():
    '''
    Get the code and exchange it for an access_token
    '''
    code = request.args.get('code')

    auth = basecamp.api.Auth(client_id, client_secret, return_url)

    token = auth.get_token(code)

    # do things now that you have a token.
```

class basecamp.auth.**Auth**(*client_id*, *client_secret*, *redirect_uri*)

Class to perform basic auth operations

get_accounts(*access_token*, *account_type='bcx'*)

Get 37signals accounts for the authenticated user.

Parameters

- **access_token** – access token obtained from *get_token()*
- **account_type** – type of basecamp account to return. Return only Basecamp Next accounts by default.

Return type dictionary

get_identity(*access_token*)

Get the users identity.

As per the docs:

> An identity is **NOT** used for determining who this user is within a specific application. The id field should NOT be used for submitting data within any application's API. This field can be used to get a user's name and email address quickly, and the id field could be used for caching on a cross-application basis if needed.

Parameters **access_token** – access token obtained from *get_token()*

Return type dictionary

get_token(*code*)

This function requests the auth token from basecamp after oAuth has happened and the user has approved the application.

Parameters **code** – the code returned from *launchpad_url()*

Return type dictionary

The response should contain the following:

•expires_in (seconds)

•access_token (a really long string, you'll need this later)

•refresh_token (another really long string. Hang onto this as well.)

launchpad_url
Get the URL to send your application to.

For instance, in a Django app, one could do something like:

```
>>> import basecamp.api
>>> from django import http
>>> auth = basecamp.api.Auth(client_id, client_secret, redirect_uri)
>>> http.HttpResponseRedirect(auth.get_launchpad_url)
```

3.2 Documents

For more information, please see the official Basecamp API documentation on documents

class basecamp.documents.**Document**(*account_url*, *access_token*, *refresh_token=None*)
Actions on a document

create(*project_id*, *title*, *content*)
Create a new document.

Parameters

- **project_id** – project id to create the document in.
- **title** – title of the document.
- **content** – content of the new document.

Rtype dictionary dictionary representation of the new document.

```
>>> import basecamp.api
>>> url = 'https://basecamp.com/1/api/v1'
>>> token = 'foo'
>>> refresh_token = 'bar'
>>> documents = basecamp.api.Document(url, token, refresh_token)
>>> documents.create(22, 'Cole Porter Songs', 'My favorite songs.')
```

Note: The JSON response will look similar to getting details of a document from *fetch()*

fetch(*document_id=None*, *project_id=None*)
Get a specific document, or a list of documents, either by project, or all documents a user has access to in the basecamp account.

Parameters

- **document_id** – integer of document
- **project_id** – integer of project

Rtype dictionary Dictionary of documents, or a single document.

Note: There are three methods of document retrieval.

1.Global for the account. document_id and project_id kwargs are omitted

2.Documents limited to a specific project. The project_id kwargs is passed to the method call.

3. Details on a specific document. The `project_id` and `document_id` kwargs are passed to the method call.

Warning: Passing a `document_id` but no `project_id` will cause a `BasecampAPIError` exception to be raised.

Examples:

All documents in the account:

```
>>> import basecamp.api
>>> url = 'https://basecamp.com/1/api/v1'
>>> token = 'foo'
>>> refresh_token = 'bar'
>>> documents = basecamp.api.Document(url, token, refresh_token)
>>> documents.fetch()
```

Get documents within a project:

```
>>> import basecamp.api
>>> url = 'https://basecamp.com/1/api/v1'
>>> token = 'foo'
>>> refresh_token = 'bar'
>>> documents = basecamp.api.Document(url, token, refresh_token)
>>> documents.fetch(project_id=123)
```

Get details on a single document:

```
>>> import basecamp.api
>>> url = 'https://basecamp.com/1/api/v1'
>>> token = 'foo'
>>> refresh_token = 'bar'
>>> documents = basecamp.api.Document(url, token, refresh_token)
>>> documents.fetch(document_id=123, project_id=123)
```

remove (*project_id*, *document_id*)

Delete a document from the project/account

Parameters

- **project_id** – integer of project id
- **document_id** – integer of document id to remove

Rtype boolean `True` if the document is removed.

```
>>> import basecamp.api
>>> url = 'https://basecamp.com/1/api/v1'
>>> token = 'foo'
>>> refresh_token = 'bar'
>>> documents = basecamp.api.Document(url, token, refresh_token)
>>> documents.remove(22, 244)
```

Note: If the document is not removed, or if a problem occurs, a :class::*BasecampAPIError* exception will be raised.

---

update (*project_id*, *document_id*, *title*, *content*)

Update a document.

Parameters

- **project_id** – integer of project id
- **document_id** – integer of document id
- **title** – string of title
- **content** – string of document content

Rtype dictionary Document information

```
>>> import basecamp.api
>>> url = 'https://basecamp.com/1/api/v1'
>>> token = 'foo'
>>> refresh_token = 'bar'
>>> documents = basecamp.api.Document(url, token, refresh_token)
>>> documents.update(22, 244, 'foo title', 'bar content')
```

Note: The JSON response will look similar to getting details of a document from *fetch()*

3.3 Projects

Create, edit, list, delete and archive projects in a Basecamp account.

See the Basecamp API docs for more info.

An `access_token` is needed to perform any tasks within this class.

class `basecamp.projects.`**Project**(*account_url*, *access_token*, *refresh_token=None*)

Operations on Projects in the API

archive(*project_id*, *archive=True*)

Archive or unarchive a project.

Parameters

- **project_id** – project id to archive or unarchive
- **archive** – boolean True to archive, False to unarchive

Rtype dictionary Dictionary of project details.

```
>>> import basecamp.api
>>> account_url = 'https://basecamp.com/12345/api/v1'
>>> access_token = 'access_token'
>>> api = basecamp.api.Project(account_url, access_token)
>>> projects = projects.archive(675, archive=True)
```

create(*name*, *description*)

Create a new project in a basecamp account.

Parameters

- **name** – New project name.
- **description** – New project description.

Rtype dictionary Project details dictionary.

```
>>> import basecamp.api
>>> account_url = 'https://basecamp.com/12345/api/v1'
>>> access_token = 'access_token'
>>> api = basecamp.api.Project(account_url, access_token)
>>> projects = projects.create('My Favorites Things', 'John Coltrane')
```

fetch (*project=None*, *archived=False*)
Get a project, or a list of projects.

Parameters

- **project** – project id or None
- **archived** – True or False - By default, non-archived projects are not included in the list of projects returned.

Rtype dictionary dictionary of projects see the following for the returned structure.

```
>>> import basecamp.api
>>> account_url = 'https://basecamp.com/12345/api/v1'
>>> access_token = 'access_token'
>>> api = basecamp.api.Project(account_url, access_token)
>>> projects = projects.fetch()
```

remove (*project_id*)
Remove a project

Parameters **project_id** – id of the project to delete.

Return type True if the project is removed, otherwise a `BasecampAPIError` exception.

```
>>> import basecamp.api
>>> account_url = 'https://basecamp.com/12345/api/v1'
>>> access_token = 'access_token'
>>> api = basecamp.api.Project(account_url, access_token)
>>> projects = projects.remove(675)
```

update (*project_id*, *name*, *description*)
Update an existing basecamp project.

Parameters

- **project_id** – integer of project id to update.
- **name** – project name
- **description** – project description.

Rtype dictionary Dictionary of project details.

```
>>> import basecamp.api
>>> account_url = 'https://basecamp.com/12345/api/v1'
>>> access_token = 'access_token'
>>> api = basecamp.api.Project(account_url, access_token)
>>> projects = projects.update(675, 'Giant Steps', 'John Coltrane')
```

3.4 People

Get and delete people. A typical JSON response for a person looks like:

```
{
    "id": 8675309,
    "name": "Tommy",
    "email_address": "me@example.com",
    "avatar_url": "https://example.com/foo.jpg",
    "updated_at": "2012-03-22T16:56:48-05:00",
    "url": "https://basecamp.com/9/api/v1/people/8675309-tommy.json"
},
```

This class is not meant to add/remove people from projects, or grant them access to documents/etc. That's what access is for.

Todo

Link to `accesses` class when it is complete.

See the Basecamp API docs on people for more info.

class `basecamp.people.`**`Person`**(*account_url*, *access_token*, *refresh_token=None*)

Operations on People in a particular project

`fetch`(*person=None*)

Get a person, or a list of people.

`remove`(*person*)

Delete a person

3.5 Indices and tables

- genindex
- modindex
- search

b