Pyzotero Documentation

Release 0.8.2

Stephan Hügel

July 05, 2011

CONTENTS

A Python wrapper for the Zotero API. You'll require a user ID and access key, which can be set up here.

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

INSTALLATION

Using pip: `pip install pyzotero`

From a local clone, if you wish to install Pyzotero from a specific branch:

```
git clone git://github.com/urschrei/pyzotero.git
cd pyzotero
git checkout dev
pip install .
```

Alternatively, download the latest version from https://github.com/urschrei/pyzotero/downloads, and point pip at the zip file. Example: `pip install ~/Downloads/urschrei-pyzotero-v0.3-0-g04ff544.zip`

I assume that running setup.py will also work using `easy_install`, but I haven't tested it.

The feedparser module (>= 0.5.1) is required. It should automatically be installed when installing Pyzotero using pip.

2.1 Testing

Run `tests.py` in the pyzotero directory, or, using Nose, `nosetests` from this directory. If you wish to see coverage statistics, run `nosetests --with-coverage --cover-package=pyzotero`.

USAGE

3.1 Hello World

```python
from pyzotero import zotero
zot = zotero.Zotero(user_id, user_key)
items = zot.items()
for item in items:
    print 'Author: %s | Title: %s' % (item['creators'][0]['lastName'], item['title'])
```

3.2 General Usage

First, create a new Zotero instance:

> **class** `pyzotero.zotero.`**`Zotero`**(*userID*, *userKey*)
>
> > **Parameters**
> >
> > - **userID** (*str*) – a valid Zotero API user ID
> >
> > - **userKey** (*str*) – a valid Zotero API user key
>
> Example:
>
> ```
> zot = zotero.Zotero(123, ABC1234XYZ)
> ```

Additional parameters may be set on Read API methods using the following method. They are **optional**:

> `Zotero.`**`add_parameters`**($\big[$*limit=None*, *start=None*, *order=None*, *sort=None*$\big[$, *content=None*$\big[$, *style=None*$\big]$ $\big]$ $\big]$)
>
> > **Parameters**
> >
> > - **limit** (*int*) – 1 – 99 or None
> >
> > - **start** (*int*) – 1 – total number of items in your library or None
> >
> > - **order** (*str*) – A valid Zotero field or None
> >
> > - **sort** (*str*) – asc or desc or None
> >
> > - **content** (*str*) – 'html' or 'bib', default: 'html'. If 'bib' is passed, you may also pass:
> >
> > - **style** (*str*) – Any valid CSL style in the Zotero style repository
> >
> > **Return type** list of HTML strings or None

Example:

```
zot.add_parameters(limit=7,start=3)
```

Note: Any parameters you set will be valid **for the next call only**

The special parameters `content` and `style`:

Example:

```
zot.add_parameters(content='bib', format='mla')
```

If these are set, the return value is a **list** of UTF-8 formatted HTML `div` elements, each containing an item:

```
['<div class="csl-entry">(content)</div>', ...  ]
```

READ API METHODS

4.1 Retrieving Items

Zotero.**items**()
> Returns Zotero library items

>> **Return type** list of dicts

Zotero.**top**()
> Returns top-level Zotero library items

>> **Return type** list of dicts

Zotero.**item**(*itemID*)
> Returns a specific item

>> **Parameters itemID** (*str*) – a zotero item ID

>> **Return type** list of dicts

Zotero.**children**(*itemID*)
> Returns the child items of a specific item

>> **Parameters itemID** (*str*) – a zotero item ID

>> **Return type** list of dicts

Zotero.**tag_items**(*itemID*)
> Returns items for a specific tag

>> **Parameters itemID** (*str*) – a zotero item ID

>> **Return type** list of dicts

Zotero.**group_items**(*groupID*)
> Returns items from a specific group

>> **Parameters groupID** (*str*) – a Zotero group ID

>> **Return type** list of dicts

Zotero.**group_top**(*groupID*)
> Returns top-level items from a specific group

>> **Parameters groupID** (*str*) – a Zotero group ID

>> **Return type** list of dicts

Zotero.**group_item**(*groupID*, *itemID*)
> Returns a specific item from a specific group

> Parameters
>
> > - **groupID** (*str*) – a Zotero group ID
> >
> > - **itemID** (*str*) – a zotero item ID
>
> Return type list of dicts

Zotero.**group_item_children**(*groupID*, *itemID*)
    Returns the child items of a specific item from a specific group

> Parameters
>
> > - **groupID** (*str*) – a Zotero group ID
> >
> > - **itemID** (*str*) – a Zotero item ID
>
> Return type list of dicts

Zotero.**group_items_tag**(*groupID*, *tag*)
    Returns a specific group's items for a specific tag

> Parameters
>
> > - **groupID** (*str*) – a Zotero group ID
> >
> > - **tag** (*str*) – a tag whose items you wish to return
>
> Return type list of dicts

Zotero.**group_collection_items**(*groupID*, *collection ID*)
    Returns a specific collection's items from a specific group

> Parameters
>
> > - **groupID** (*str*) – a Zotero group ID
> >
> > - **collectionID** (*str*) – a Zotero collection ID
>
> Return type list of dicts

Zotero.**group_collection_item**(*groupID*, *collectionID*, *itemID*)
    Returns a specific collection's item from a specific group

> Parameters
>
> > - **groupID** (*str*) – a Zotero group ID
> >
> > - **collectionID** (*str*) – a Zotero collection ID
> >
> > - **itemID** (*str*) – a zotero item ID
>
> Return type list of dicts

Zotero.**group_collection_top**(*groupID*, *collectionID*)
    Returns a specific collection's top-level items from a specific group

> Parameters
>
> > - **groupID** (*str*) – a Zotero group ID
> >
> > - **groupID** – a Zotero collection ID
>
> Return type list of dicts

Zotero.**collection_items**(*collectionID*)
    Returns items from the specified collection

> Parameters **collectionID** (*str*) – a Zotero collection ID

> **Return type** list of dicts

`Zotero.`**`get_subset`**(*itemIDs*)
> Retrieve an arbitrary set of non-adjacent items. Limited to 50 items per call.
>
> > **Parameters itemIDs** (*list*) – a list of Zotero Item IDs
> >
> > **Return type** list of dicts

Example of returned data:

```
[{'DOI': '',
 'ISSN': '1747-1532',
 'abstractNote': '',
 'accessDate': '',
 'archive': '',
 'archiveLocation': '',
 'callNumber': '',
 'creators': [{'creatorType': 'author',
               'firstName': 'T. J.',
               'lastName': 'McIntyre'}],
 'date': '2007',
 'extra': '',
 'issue': '',
 'itemType': 'journalArticle',
 'journalAbbreviation': '',
 'language': '',
 'libraryCatalog': 'Google Scholar',
 'pages': '',
 'publicationTitle': 'Journal of Intellectual Property Law & Practice',
 'rights': '',
 'series': '',
 'seriesText': '',
 'seriesTitle': '',
 'shortTitle': 'Copyright in custom code',
 'tags': [],
 'title': 'Copyright in custom code: Who owns commissioned software?',
 'url': '',
 'volume': ''} ... ]
```

See *'Hello World'* example, above

4.2 Retrieving Collections

`Zotero.`**`collections`**()
> Returns a user's collections
>
> > **Return type** list of dicts

`Zotero.`**`collections_sub`**(*collectionID*)
> Returns a sub-collection from a specific collection
>
> > **Parameters collectionID** (*str*) – a Zotero library collection ID
> >
> > **Return type** list of dicts

`Zotero.`**`group_collections`**(*groupID*)
> Returns collections for a specific group
>
> > **Parameters groupID** (*str*) – a Zotero group ID

> **Return type** list of dicts

`Zotero.`**`group_collection`**(*groupID*, *collectionID*)
: Returns a specific collection from a specific group

> **Parameters**
>
> - **groupID** (*str*) – a Zotero group ID
>
> - **collectionID** (*str*) – a Zotero collection ID
>
> **Return type** list of dicts

Example of returned data:

```
[{'key': 'PRMD6BGB', 'name': "A Midsummer Night's Dream"} ... ]
```

4.3 Retrieving groups

`Zotero.`**`groups`**()
: Retrieve the Zotero group data to which the current user key has access

> **Return type** list of dicts

Example of returned data:

```
[{u'description': u'%3Cp%3EBGerman+Cinema+and+related+literature.%3C%2Fp%3E',
    u'fileEditing': u'none',
    u'group_id': u'153',
    u'hasImage': 1,
    u'libraryEditing': u'admins',
    u'libraryEnabled': 1,
    u'libraryReading': u'all',
    u'members': {u'0': 436,
        u'1': 6972,
        u'15': 499956,
        u'16': 521307,
        u'17': 619180},
    u'name': u'German Cinema',
    u'owner': 10421,
    u'type': u'PublicOpen',
    u'url': u''} ... ]
```

4.4 Retrieving Tags

`Zotero.`**`tags`**()
: Returns a user's tags

> **Return type** list of strings

`Zotero.`**`item_tags`**(*itemID*)
: Returns tags from a specific item

> **Parameters** **itemID** (*str*) – a valid Zotero library Item ID
>
> **Return type** list of strings

`Zotero.`**`group_tags`**(*groupID*)
: Returns tags from a specific group

> **Parameters** **groupID** (*str*) – a valid Zotero library group ID
>
> **Return type** list of strings

Zotero.**group_item_tags**(*groupID*, *itemID*)
> Returns tags from a specific item from a specific group

> **Parameters**
>
> - **groupID** (*str*) – a valid Zotero library group ID
> - **itemID** (*str*) – a valid Zotero library Item ID
>
> **Return type** list of strings

Example of returned data:

```
['Authority in literature', 'Errata', ... ]
```

FIVE

WRITE API METHODS

5.1 Item Methods

Zotero.**item_types**()
> Returns a dict containing all available item types

>> **Return type** dict

Zotero.**item_fields**()
> Returns a dict of all available item fields

>> **Return type** dict

Zotero.**item_creator_types**(*itemtype*)
> Returns a dict of all valid creator types for the specified item type

>> **Parameters** **itemtype** (*str*) – a valid Zotero item type. A list of available item types can be obtained by the use of `item_types()`

>> **Return type** dict

Zotero.**item_template**(*itemtype*)
> Returns an item creation template for the specified item type

>> **Parameters** **itemtype** (*str*) – a valid Zotero item type. A list of available item types can be obtained by the use of `item_types()`

>> **Return type** dict

Zotero.**create_item**(*items*)
> Create Zotero library items.

>> **Parameters** **items** (*list*) – one or more dicts containing item data.

>> **Return type** list of dicts

> Returns a copy of the created item(s), if successful. The use of `item_template()` is recommended in order to first obtain a dict with a structure which the API will accept. Example:

```
template = zot.item_template('book')
template['creators'][0]['firstName'] = 'Monty'
template['creators'][0]['lastName'] = 'Cantsin'
template['title'] = 'Maris Kundzins: A Life'
resp = zot.create_item([template])
```

> If successful, `resp` will have the same structure as items retrieved with an `items()` call, e.g. a list of one or more dicts (see *Item Data*, above).

Zotero.**update_item**(*item*)
    Update an item in your library.

        Parameters item (*dict*) – a dict containing item data.

        Return type Boolean

    Example:

```python
i = zot.items()
# see above for example of returned item structure
# modify the latest item which was added to your library
i[0]['title'] = 'The Sheltering Sky'
i[0]['creators'][0]['firstName'] = 'Paul'
i[0]['creators'][0]['lastName'] = 'Bowles'
zot.update_item(i[0])
```

Zotero.**delete_item**(*item*)
    Delete an item from your library.

        Parameters item (*dict*) – a dict containing item data. As in the previous example, you must first retrieve the item(s) you wish to delete, and pass it/them to the method one by one. Deletion of multiple items is most easily accomplished using e.g. a `for` loop.

        Return type Boolean

    Example:

```python
i = zot.items()
# only delete the last five items we added
to_delete = i[:6]
for d in to_delete:
    zot.delete_item(d)
```

5.2 Collection Methods

Zotero.**create_collection**(*name*)
    Create a new collection in the Zotero library.

        Parameters name (*dict*) – dict containing the key `name` and the value of the new collection name you wish to create. May optionally contain a `parent` key, the value of which is the ID of an existing collection. If this is set, the collection will be created as a child of that collection.

        Return type Boolean

Zotero.**addto_collection**(*collection*, *items*)
    Add the specified item(s) to the specified collection.

        Parameters

            • **collection** (*str*) – a collection key

            • **items** (*list*) – list of one or more item dicts.

        Return type Boolean

    Collection keys can be obtained by a call to `collections()` (see details above).

Zotero.**deletefrom_collection**(*collection*, *item*)
    Remove the specified item from the specified collection.

> **Parameters**
>
> - **collection** (*str*) – a collection key
>
> - **item** (*dict*) – dict containing item data.
>
> **Return type** Boolean

See the `delete_item()` example for multiple-item removal.

Zotero.**update_collection**(*collection*)
Update an existing collection name.

> **Parameters** **collection** (*dict*) – a dict containing collection data, previously retrieved using one of the Collections calls (e.g. `collections()`).
>
> **Return type** Boolean

Example:

```
# get existing collections, which will return a list of dicts
c = zot.collections()
# rename the last collection created in the library
c[0]['name'] = 'Whither Digital Humanities?'
# update collection name on the server
zot.update_collection(c[0])
```

Zotero.**delete_collection**(*collection*)
Delete a collection from the Zotero library.

> **Parameters** **collection** (*dict*) – a dict containing collection data, previously retrieved using one of the Collections calls (e.g. `collections()`).
>
> **Return type** Boolean

See the `delete_item()` example for ways to delete multiple collections.

NOTES

All Read API methods return **lists** of **dicts** or, in the case of tag methods, **lists** of **strings**. Most Write API methods return either `True` if successful, or raise an error. See `zotero_errors.py` for a full listing of these.

> **Warning:** URL parameters will supersede API calls which should return e.g. a single item: `https://api.zotero.org/users/436/items/ABC?start=50&limit=10` will return 10 items beginning at position 50, even though `ABC` does not exist. Be aware of this, and don't pass URL parameters which do not apply to a given API method. This is a limitation/foible of the Zotero API, and there's nothing I can do about it.

LICENSE

Pyzotero is licensed under the GNU GPL Version 3 license, in line with Zotero's own license. Details can be found in the file `license.txt`.

PYTHON MODULE INDEX

p

pyzotero.zotero, 1

INDEX