Pyzotero Documentation

Release 0.9.6

Stephan Hügel

March 13, 2012

CONTENTS

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

CHAPTER

ONE

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/tags, 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 (>= 0.5.1) and pytz modules are required. They will be automatically installed when installing Pyzotero using pip.

1.1 Testing

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

1.2 Reporting issues

If you encounter an error while using Pyzotero, please open an issue on its Github issues page.

CHAPTER

TWO

USAGE

2.1 Hello World

```
# retrieve the last five top-level items you added to your library
from pyzotero import zotero
zot = zotero.Zotero(user_id, user_key)
zot.add_parameters(limit = 5)
items = zot.top()
# print each item's item type and ID
for item in items:
    print 'Item Type: %s | Key: %s' % (item['itemType'], item['key'])
```

2.2 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)
```

CHAPTER

THREE

READ API METHODS

3.1 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.**trash**()
Returns library items from the user's trash

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_trash**(*groupID*)
Returns items from a specific group's trash

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?',
 'updated': 'Mon, 14 Mar 2011 22:30:17 GMT',
 'url': '',
 'volume': ''} ... ]
```

See *'Hello World'* example, above

3.2 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"} ... ]
```

3.3 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''} ... ]
```

3.4 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', ... ]
```

3.5 The `follow()` method

This method (currently experimental) aims to make Pyzotero a little more RESTful. Following any Read API call which can retrieve **multiple items**, calling `follow()` will repeat that call, but for the next *x* number of items, where *x* is either a number set by the user for the original call, or 50 by default. Each subsequent call to `follow()` will extend the offset.

Example:

```
from pyzotero import zotero
zot = zotero.Zotero(user_id, user_key)
# only retrieve a single item
zot.add_parameters(limit = 1)
# this will retrieve the most recently added/modified top-level item
first_item = zot.top()
# now we can start retrieving subsequent items
next_item = zot.follow()
third_item = zot.follow()
```

3.6 The `everything()` method

This method (currently experimental) will retrieve **all** library items specified by its argument: a valid Read API call which can retrieve multiple items..

Example:

```
from pyzotero import zotero
zot = zotero.Zotero(user_id, user_key)
# retrieve all top-level items
toplevel = zot.everything(zot.top())
```

The `everything()` method should work with all Pyzotero Read API calls which can return multiple items, but has not yet been extensively tested. Feedback is welcomed.

Warning: The `follow()` and `everything()` methods are only valid for methods which can return multiple library items. For instance, you cannot use `follow()` after an `item()` call.

3.7 Retrieving item counts

If you wish to retrieve item counts for subsets of a library, you can use the following methods:

`Zotero.`**`num_items`**`()`
Returns the count of top-level items in the library

Return type int

`Zotero.`**`num_collectionitems`**`(`*collectionID*`)`
Returns the count of items in the specified collection

Return type int

`Zotero.`**`num_tagitems`**`(`*tag*`)`
Returns the count of items for the specified tag

Return type int

`Zotero.`**`num_groupitems`**`(`*groupID*`)`
Returns the count of items in the specified group

Return type int

3.8 Additional Parameters for Read API calls

Additional parameters may be set on Read API methods using the following method. All parameters are optional. **You may also set a search term here, using the 'itemType', 'q', or 'tag' parameters**. This area of the Zotero Read API is under heavy development as of early 2012, and may change frequently. See the API documentation for the most up-to-date details of search syntax usage and export format details.

`Zotero.`**`add_parameters`**`(`[*format=None*, *itemKey=None*, *itemType=None*, *q=None*, *tag=None*, *limit=None*, *start=None*, *order=None*, *sort=None*[, *content=None*[, *style=None*]]])

Parameters

- **itemKey** (*str*) – A comma-separated list of item keys. Valid only for item requests. Up to 50 items can be specified in a single request.
- **itemType** (*str*) – item type search
- **q** (*str*) – a search term, which currently matches titles and individual creator fields
- **tag** (*str*) – tag search
- **limit** (*int*) – 1 – 99 or None
- **start** (*int*) – 1 – total number of items in your library or None
- **order** (*str*) – any one of the following: "dateAdded", "dateModified", "title", "creator", "type", "date", "publisher", "publication", "journalAbbreviation", "language", "accessDate", "libraryCatalog", "callNumber", "rights", "addedBy", "numItems"
- **sort** (*str*) – 'asc' or 'desc'
- **format** (*str*) – only 'keys' is currently supported as an alternate format
- **content** (*str*) – 'bib', or one of the export formats (see below). 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**

A note on the `content` and `style` parameters:

Example:

```
zot.add_parameters(content='bib', style='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>', ...  ].
```

You may also set `content='citation'` if you wish to retrieve citations. Similar to `bib`, the result will be a list of one or more HTML `span` elements.

If you select one of the available export formats as the `content` parameter, pyzotero will in most cases return a list of unicode strings in the format you specified. The exception is the `csljson` format, which is parsed into a list of dicts. Please note that you must provide a `limit` parameter if you specify one of these export formats. Multiple simultaneous retrieval of particular formats, e.g. `content="json,coins"` is not currently supported.

If you set `format='keys'`, a newline-delimited string containing item keys will be returned

CHAPTER

FOUR

WRITE API METHODS

4.1 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.**creator_fields**()
Returns a dict containing all localised creator fields

Return type dict

Zotero.**item_type_fields**(*itemtype*)
Returns all valid fields 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 list of dicts

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.**check_items**(*items*)
Check whether items to be created on the server contain only valid keys. This method first creates a set of valid keys by calling `item_fields()`, then compares the user-created dicts to it. If any keys in the user-created dicts are unknown, a `KeyError` exception is raised.

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

Return type Boolean

Zotero.**create_items**(*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_items([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:

```
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:

```
i = zot.items()
# only delete the last five items we added
to_delete = i[:5]
for d in to_delete:
    zot.delete_item(d)
```

Zotero.**add_tags**(*item*, *tag*[, *tag* ...])
Add one or more tags to an item, and update it on the server

Parameters

- **item** (*dict*) – a dict containing item data
- **tag** (*string*) – the tag you'd like to add to the item

Return type list of dicts

Example:

```
zot.add_parameters(limit=1)
z = zot.top()
# we've now retrieved the most recent top-level item
updated = zot.add_tags(z[0], 'tag1', 'tag2', 'tag3')
# updated now contains a representation of the updated server item
```

4.2 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.

CHAPTER

FIVE

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.

CHAPTER

SIX

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

INDEX