wikibase-api

Release 0.1.1

Feb 09, 2020

Getting Started

This is the documentation of the `wikibase-api` Python library. It is a wrapper around the MediaWiki and Wikibase API. You can use it to query and edit information on Wikidata or another Wikibase instance.

CHAPTER 1

Installation and Usage

1.1 1. Installation

```
pip install wikibase-api
```

1.2 2. Usage

To access the API, simply create an instance of the *Wikibase* class:

```
from wikibase_api import Wikibase

wb = Wikibase()
```

Note: The Wikibase instance which is accessed by default is Wikidata. To use another instance, e.g. a local one for testing, set the `api_url` parameter accordingly. You can find a guide on how to set up your own instance locally using Docker under *Local Wikibase Instance*.

1.3 3. Queries

You can query a Wikibase instance (e.g. Wikidata) by simply creating an object of the `Wikibase` class and calling a query function. For example, you ask for all information about an item:

```
from wikibase_api import Wikibase

wb = Wikibase()
r = wb.entity.get("Q1")
print(r)
```

Output:

```
{
  "entities": {
    "Q1": {
      # ...
    }
  },
  "success": 1,
}
```

1.4 4. Edits

You can also make edits using the Wikibase API, e.g. create an empty item:

```
r = wb.entity.add("item")
print(r)
```

Output:

```
{
  "entity": {
    "labels": {},
    "descriptions": {},
    "aliases": {},
    "sitelinks": {},
    "claims": {},
    "id": "Q1",
    "type": "item",
    "lastrevid": 1234
  },
  "success": 1
}
```

For a list of all available API functions, have a look at the *API Reference*.

Note: If you plan to make edits on Wikidata, it's a good idea to test them on the sandbox item.

Before being able to make requests, you need to authenticate yourself to the API. You have two options:

- Authentication using *OAuth*
- Authentication with a *user account*

OAuth is the recommended method as it is more secure than logging in with username and password. However, setting up OAuth is more complicated and requires you to apply for credentials.

1.4.1 a) OAuth

To be able to use OAuth, you need to obtain credentials for an owner-only consumer. This information can be obtained at `Special:OAuthConsumerRegistration/propose` (i.e. https://meta.wikimedia.org/wiki/Special:OAuthConsumerRegistration/propose for Wikimedia or http://localhost:8181/wiki/Special:OAuthConsumerRegistration/propose on a local instance):

1. Log in using your username and password
2. Fill in the registration form:
    - Application name and description
    - Tick "This consumer is for use only by <username>"
    - Select the grants you need, e.g. "High-volume editing", "Edit existing pages", "Create, edit, and move pages", and "Delete pages, revisions, and log entries"
3. Click the "Propose consumer" button at the bottom of the page
4. Write down your OAuth consumer information

Now, you can create an instance of the *Wikibase* class using your newly obtained OAuth credentials:

```
from wikibase_api import Wikibase

oauth_credentials = {
    "consumer_key": "...",
    "consumer_secret": "...",
    "access_token": "...",
    "access_secret": "...",
}

wb = Wikibase(oauth_credentials=oauth_credentials)
```

Note: Some additional steps are required when using OAuth on a local Wikibase instance (see oauth_on_local_wikibase_instance).

1.4.2 b) User Login

Bot passwords allow users to access the API without providing their account's main login credentials. You can generate a bot password under `Special:BotPasswords` (i.e. https://www.wikidata.org/wiki/Special:BotPasswords on Wikidata or http://localhost:8181/wiki/Special:BotPasswords on a local instance):

1. Log in using your username and password
2. Fill in the registration form:
    - Choose a bot name (this will be a suffix to your username)
    - Select the grants you need, e.g. "High-volume editing", "Edit existing pages", "Create, edit, and move pages", and "Delete pages, revisions, and log entries"
3. Click the "Create" button at the bottom of the page
4. Write down your bot username and password

Now, you can create an instance of the *Wikibase* class using your newly obtained bot credentials:

```
from wikibase_api import Wikibase

login_credentials = {
    "bot_username": "...",
    "bot_password": "...",
}

wb = Wikibase(login_credentials=login_credentials)
```

CHAPTER 2

API Reference

class wikibase_api.**Wikibase**(*api_url='https://www.wikidata.org/w/api.php', oauth_credentials=None, login_credentials=None, is_bot=False, summary='Modified using wikibase-api for Python', config_path=None*)

This is the Wikibase API wrapper class.

Parameters

- **api_url** (*str*) – URL to the API of the relevant Wikibase instance. Default: `"https://www.wikidata.org/w/api.php"`. For a local instance, you might use `"http://localhost:8181/w/api.php"`
- **oauth_credentials** (*dict*) – Dictionary with the keys `consumer_key`, `consumer_secret`, `access_token`, and `access_secret`
- **login_credentials** (*dict*) – Dictionary with the keys `bot_username` and `bot_password`
- **is_bot** (*bool*) – Mark edits as created by a bot. Default: `false`
- **summary** (*str*) – Summary for edits. An auto-generated comment will be added before the summary. Together, they cannot be longer than 260 characters. Default: `"Modified using wikibase-api for Python"`
- **config_path** (*str*) – Path to a config.json configuration file. If specified, the other parameters are loaded from this file. The default values are the same as above

2.1 Alias

class wikibase_api.models.**Alias**(*api*)

Collection of API functions for aliases

Example function call:

```python
from wikibase_api import Wikibase

wb = Wikibase(
    # Parameters
)

r = wb.alias.add("Q1", "The Universe", "en")
print(r)
```

add(*entity_id*, *aliases*, *language*)
Add one or multiple new aliases to the specified entity

Parameters

- **entity_id** (*str*) – Entity identifier (e.g. "Q1")
- **aliases** (*str or list(str)*) – Aliases to add to the existing ones
- **language** (*str*) – Language of the description (e.g. "en")

Returns Response

Return type dict

remove(*entity_id*, *aliases*, *language*)
Remove one or multiple aliases from the specified entity

Parameters

- **entity_id** (*str*) – Entity identifier (e.g. "Q1")
- **aliases** (*str or list(str)*) – Existing aliases to remove
- **language** (*str*) – Language of the description (e.g. "en")

Returns Response

Return type dict

replace_all(*entity_id*, *aliases*, *language*)
Replace all existing aliases with the specified one(s) for an entity

Parameters

- **entity_id** (*str*) – Entity identifier (e.g. "Q1")
- **aliases** (*str or list(str)*) – Aliases to add after deleting all existing ones
- **language** (*str*) – Language of the description (e.g. "en")

Returns Response

Return type dict

2.2 Claim

class wikibase_api.models.**Claim**(*api*)
Collection of API functions for claims

Example function call:

```python
from wikibase_api import Wikibase

wb = Wikibase(
    # Parameters
)

r = wb.claim.get("Q1")
print(r)
```

add(*entity_id*, *property_id*, *value*, *snak_type='value'*)
Create a new claim for the specified entity

Parameters

- **entity_id** (*str*) – Entity identifier (e.g. `"Q1"`)
- **property_id** (*str*) – Property identifier (e.g. `"P1"`)
- **value** (*any*) – Value of the claim. If snak_type is set to "novalue" or "somevalue", value must be None
- **snak_type** (*str*) – Value type (one of `["value", "novalue", "somevalue"]`. `"value"` (default) is used for normal property-value pairs. `"novalue"` is used to indicate that an item has none of the property (e.g. a person has no children). `"somevalue"` is used when it is known that a value exists, but the value itself is not known

Returns Response

Return type dict

remove(*claim_ids*)
Delete one or multiple claims

Parameters **claim_ids** (*str or list(str)*) – Claim identifier(s) (e.g. `"Q2$8C67587E-79D5-4E8C-972C-A3C5F7ED06B3"` or `["Q2$8C67587E-79D5-4E8C-972C-A3C5F7ED06B3", "Q2$ACC73295-5CF2-4B6A-95AA-CF156AB2B036"]`), can be obtained with `get()`

Returns Response

Return type dict

update(*claim_id*, *value*, *snak_type='value'*)
Update the value of the specified claim

Parameters

- **claim_id** (*str*) – Claim identifier (e.g. `"Q2$8C67587E-79D5-4E8C-972C-A3C5F7ED06B3"`), can be obtained with `get()`
- **value** (*any*) – Value of the claim. If snak_type is set to "novalue" or "somevalue", value must be None
- **snak_type** (*str*) – Value type (one of `["value", "novalue", "somevalue"]`. `"value"` (default) is used for normal property-value pairs. `"novalue"` is used to indicate that an item has none of the property (e.g. a person has no children). `"somevalue"` is used when it is known that a value exists, but the value itself is not known

Returns Response

Return type dict

2.3 Description

`class wikibase_api.models.`**`Description`**`(`*api*`)`

Collection of API functions for descriptions

Example function call:

```python
from wikibase_api import Wikibase

wb = Wikibase(
    # Parameters
)

r = wb.description.set("Q1", "totality of space and all matter and radiation in it
↪")
print(r)
```

`set``(`*entity_id*, *description*, *language*`)`

Set the title in the specified language for an entity

Parameters

- **`entity_id`** (*`str`*) – Entity identifier (e.g. `"Q1"`)
- **`description`** (*`str`*) – Value to set the description to (e.g. `"third planet from the Sun in the Solar System"`)
- **`language`** (*`str`*) – Language of the description (e.g. `"en"`)

Returns Response

Return type dict

2.4 Entity

`class wikibase_api.models.`**`Entity`**`(`*api*`)`

Collection of API functions for Wikibase entities (items, properties, . . .)

Example function call:

```python
from wikibase_api import Wikibase

wb = Wikibase(
    # Parameters
)

content = {"labels": {"en": {"language": "en", "value": "Updated label"}}}
r = wb.entity.update("Q1", content=content)
print(r)
```

`add``(`*entity_type*, *content=None*`)`

Create a new Wikibase entity

Parameters

- **`entity_type`** (*`str`*) – Type of entity to be created (e.g. `"item"`)

- **`content`** (*`dict`*) – Content of the new entity

Returns Response

Return type dict

`get` (*entity_ids*, *attributes=None*, *languages=None*)

Get the data of one or multiple Wikibase entities

Parameters

- **`entity_ids`** (*`str or list(str)`*) – Entity identifier(s) (e.g. `"Q1"` or `["Q1", "Q2"]`)
- **`attributes`** (*`list(str)`*) – Names of the attributes to be fetched from each entity (e.g. `"claims"`)
- **`languages`** (*`list(str)`*) – Languages to return the fetched data in (e.g. `"en"`)

Returns Response

Return type dict

`remove` (*title*, *reason=None*)

Delete the specified Wikibase entity

Parameters

- **`title`** (*`str`*) – Entity title (e.g. `"Item:Q1"` or `"Property:P1"`)
- **`reason`** – Reason for the deletion (if not set, Wikibase will use an automatically generated reason)

Returns Response

Return type dict

`search` (*search_key*, *language*, *entity_type='item'*, *limit=10*, *offset=0*)

Search for entities based on their labels and aliases

Parameters

- **`search_key`** (*`str`*) – String for which Wikibase entities' labels and aliases are searched
- **`language`** (*`str`*) – Languages to search in (e.g. `"en"`)
- **`entity_type`** (*`str`*) – Type of entities to search for. Default: "item"
- **`limit`** (*`int`*) – Maximum number of results to return. Default: 10
- **`offset`** (*`int`*) – Offset where to continue a search. Default: 0

Returns Response

Return type dict

`update` (*entity_id*, *content*)

Modify the specified Wikibase entity

Parameters

- **`entity_id`** (*`str`*) – Entity identifier (e.g. `"Q1"`)
- **`content`** (*`dict`*) – Content to add to the entity

Returns Response

Return type dict

2.5 Label

class wikibase_api.models.**Label**(*api*)
Collection of API functions for labels

Example function call:

```
from wikibase_api import Wikibase

wb = Wikibase(
    # Parameters
)

r = wb.label.set("Q1", "univers", "fr")
print(r)
```

set(*entity_id*, *label*, *language*)
Set the label in the specified language for an entity

Parameters

- **entity_id** (*str*) – Entity identifier (e.g. "Q1")
- **label** (*str*) – Value to set the label (site title) to (e.g. "Universe")
- **language** (*str*) – Language of the description (e.g. "en")

Returns Response

Return type dict

2.6 Qualifier

class wikibase_api.models.**Qualifier**(*api*)
Collection of API functions for qualifiers

Example function call:

```
from wikibase_api import Wikibase

wb = Wikibase(
    # Parameters
)

claim_id = "Q2$8C67587E-79D5-4E8C-972C-A3C5F7ED06B3"
r = wb.qualifier.add(claim_id, "P585", "13700 million years BCE")
print(r)
```

add(*claim_id*, *property_id*, *value*, *snak_type='value'*)
Create a new qualifier for the specified claim

Parameters

- **claim_id** (*str*) – Claim identifier (e.g. "Q2$8C67587E-79D5-4E8C-972C-A3C5F7ED06B3")
- **property_id** (*str*) – Property identifier (e.g. "P1")
- **value** (*any*) – Value of the qualifier. If snak_type is set to "novalue" or "somevalue", value must be None

- **snak_type** (*str*) – Value type (one of `["value", "novalue", "somevalue"]`. `"value"` (default) is used for normal property-value pairs. `"novalue"` is used to indicate that an item has none of the property (e.g. a person has no children). `"somevalue"` is used when it is known that a value exists, but the value itself is not known

Returns Response

Return type dict

remove (*claim_id*, *qualifier_ids*)
Delete the specified qualifier(s)

Parameters

- **claim_id** (*str*) – Claim identifier (e.g. `"Q2$8C67587E-79D5-4E8C-972C-A3C5F7ED06B3"`)
- **qualifier_ids** (*str or list(str)*) – Hash(es) of the qualifier(s) to be deleted (e.g. `"e3401fd064ec7c3cb7169aca6efff7419d95312a"`, `["e3401fd064ec7c3cb7169aca6efff7419d95312a", "d86fda314abf561afca0d1fef97546ea050f3c1e"]`)

Returns Response

Return type dict

update (*claim_id*, *qualifier_id*, *property_id*, *value*, *snak_type='value'*)
Update the value of the specified qualifier

Parameters

- **claim_id** (*str*) – Claim identifier (e.g. `"Q2$8C67587E-79D5-4E8C-972C-A3C5F7ED06B3"`)
- **property_id** (*str*) – Property identifier (e.g. `"P1"`)
- **qualifier_id** (*str*) – Hash of the qualifier to be updated (e.g. `"e3401fd064ec7c3cb7169aca6efff7419d95312a"`)
- **value** (*any*) – Value of the qualifier. If snak_type is set to "novalue" or "somevalue", value must be None
- **snak_type** (*str*) – Value type (one of `["value", "novalue", "somevalue"]`. `"value"` (default) is used for normal property-value pairs. `"novalue"` is used to indicate that an item has none of the property (e.g. a person has no children). `"somevalue"` is used when it is known that a value exists, but the value itself is not known

Returns Response

Return type dict

2.7 Reference

class wikibase_api.models.**Reference** (*api*)
Collection of API functions for references

Example function call:

```
from wikibase_api import Wikibase

wb = Wikibase(
```

(continues on next page)

(continued from previous page)

```
    # Parameters
)

claim_id = "Q2$8C67587E-79D5-4E8C-972C-A3C5F7ED06B3"
r = wb.reference.add(claim_id, "P854", "https://example.com")
print(r)
```

add(*claim_id*, *property_id*, *value*, *snak_type='value'*, *index=None*)
Create a new reference for the specified claim

Parameters

- **claim_id** (*str*) – Claim identifier (e.g. `"Q2$8C67587E-79D5-4E8C-972C-A3C5F7ED06B3"`)
- **property_id** (*str*) – Property identifier (e.g. `"P1"`)
- **value** (*any*) – Value of the reference. If snak_type is set to "novalue" or "somevalue", value must be None
- **snak_type** (*str*) – Value type (one of `["value", "novalue", "somevalue"]`. `"value"` (default) is used for normal property-value pairs. `"novalue"` is used to indicate that an item has none of the property (e.g. a person has no children). `"somevalue"` is used when it is known that a value exists, but the value itself is not known
- **index** (*int*) – Position of the new reference within the list of references (e.g. `0` to add the reference to the top of the list)

Returns Response

Return type dict

remove(*claim_id*, *reference_ids*)
Delete the specified reference(s)

Parameters

- **claim_id** (*str*) – Claim identifier (e.g. `"Q2$8C67587E-79D5-4E8C-972C-A3C5F7ED06B3"`)
- **reference_ids** (*str or list(str)*) – Hash(es) of the reference(s) to be deleted (e.g. `"9d5f29a997ad9ced2b1138556a896734148c4a0c"`, `["9d5f29a997ad9ced2b1138556a896734148c4a0c", "0b0ca37729a3f637c100832d2a30fe9d867ef385"]`)

Returns Response

Return type dict

update(*claim_id*, *property_id*, *reference_id*, *value*, *snak_type='value'*, *index=None*)
Update the value of the specified reference

Parameters

- **claim_id** (*str*) – Claim identifier (e.g. `"Q2$8C67587E-79D5-4E8C-972C-A3C5F7ED06B3"`)
- **property_id** (*str*) – Property identifier (e.g. `"P1"`)
- **reference_id** (*str*) – Hash of the reference to be updated (e.g. `"9d5f29a997ad9ced2b1138556a896734148c4a0c"`)
- **value** (*any*) – Value of the reference. If snak_type is set to "novalue" or "somevalue", value must be None

- **`snak_type`** (*`str`*) – Value type (one of `["value", "novalue", "somevalue"]`. `"value"` (default) is used for normal property-value pairs. `"novalue"` is used to indicate that an item has none of the property (e.g. a person has no children). `"somevalue"` is used when it is known that a value exists, but the value itself is not known
- **`index`** (*`int`*) – Position of the new reference within the list of references (e.g. `0` to add the reference to the top of the list)

Returns Response

Return type dict

CHAPTER 3

Local Wikibase Instance

The following is an introduction on how to build your own Wikibase instance using Docker. This instance can then be used for testing modifications using the Wikibase API without risking unwanted changes to the live instance.

3.1 Requirements

Install Docker and `docker-compose` if you don't have the tools already.

3.2 Setting up Wikibase

Next, use wikibase-docker to set up Wikibase and its query service on your machine:

1. Create an empty directory: `mkdir wikibase-docker && cd wikibase-docker`
2. Download the `docker-compose.yml` file from the wikibase-docker repo to the directory: `wget https://raw.githubusercontent.com/wmde/wikibase-docker/master/docker-compose.yml`
3. Set up the Docker containers: `docker-compose pull`
4. Start the containers: `docker-compose up`

After a while, you should have a Wikibase instance up and running. MediaWiki can be accessed on http://localhost:8181 and the SPARQL UI at http://localhost:8282.

CHAPTER 4

More information

- **MediaWiki API specification:** https://www.wikidata.org/w/api.php?action=help
- **OAuth:**
  - OAuth setup: https://www.mediawiki.org/wiki/OAuth/For_Developers
  - OAuth app guidelines: https://meta.wikimedia.org/wiki/OAuth_app_guidelines
- **API libraries in other languages:**
  - JavaScript: wikidata-edit (https://github.com/maxlath/wikidata-edit)
  - PHP: wikibase-api (https://github.com/addwiki/wikibase-api)

CHAPTER 5

Development

You can use the following steps to test and modify this library on your machine.

5.1 Requirements

Make sure you have Python 3.6+ and Poetry installed.

5.2 Setup

1. Clone the repository from GitHub.
2. Run `make install` to install the project's dependencies and Git hooks.
3. Set up `wikibase-docker` by following the "*Local Wikibase Instance*" guide.
4. Rename the `config-example.json` file to `config-tests.json`. This is the configuration file that will be used for testing. Fill in either the `oauth_credentials` or the `login_credentials` parameters and delete the other. If you didn't change `wikibase-docker`'s configuration, you can use the following:

   ```
   {
     "apiUrl": "http://localhost:8181/w/api.php",
     "loginCredentials": {
       "botUsername": "WikibaseAdmin",
       "botPassword": "WikibaseDockerAdminPass"
     }
   }
   ```

   In `config-tests.json`, you can also specify other parameters you want to pass to the *`Wikibase`* class during testing.
5. Make your changes to the code.
6. Make sure the tests are still passing (`make test`).

Index