Koordinates Python API Client Documentation

Release 0.2.0

Koordinates

April 20, 2017

Release v0.2.0.

A BSD-licensed Python client library for a number of Koordinates web APIs.

The library provides easy access to Koordinates web services, particularly the Administration APIs.:

```python
import koordinates

client = koordinates.Client(host='labs.koordinates.com', token='MY_API_TOKEN')

# print the 10 most recently created layers
for layer in client.layers.order_by('created_at')[:10]:
   print(layer)
```

CHAPTER 1

Features

The library aims to reflect the available Koordinates web APIs. Currently the following APIs have support in the library:

- Data Catalog API
- Layers & Tables API
- Sets API
- Publishing API
- Licenses API
- Metadata API
- Token API

We're working hard to add support for additional APIs to the library, so expect this list to grow soon.

Compatibility

- Python 2.7
- Python 3.3+

CHAPTER 2

User Guide

Introduction

License

The Koordinates Python Client Library is released under the terms of the BSD License.

Installation

This part of the documentation covers the installation of the Koordinates Python Client Library.

Pip

Installing the Python Library is simple with pip. All you need to do is run the following in your terminal:

```
$ pip install koordinates
```

Getting the Code

The Koordinates Python Client Library is on GitHub.

Development occurs in the master branch, and releases are tagged and pushed to PyPI regularly.

You can either clone the public git repository:

```
$ git clone git://github.com/koordinates/python-client.git
```

Or, download the latest release.

Once you have a copy of the source, you can embed it in your Python package, or install it into your virtualenv/site-packages:

```
$ python setup.py install
```

Upgrading

We strongly encourage you to use the latest release of the Python Library to ensure you get the benefit of fixes, improvements, and new features. The library follows the Semantic Versioning guidelines, so all releases with the same major version number (eg. `1.x.x`) will be backwards compatible.

Quick Start

In this guide, we provide a very short overview of how the library may be used to achieve some common tasks.

Before you begin, you'll need to know the Koordinates site you're accessing (eg. labs.koordinates.com), and have a valid API token for the site, created with the scopes you need for the APIs you're using. See Authentication for more information. You'll also need sufficient permissions on the site to take actions (for example, creating a Layer).

First, import the Koordinates module:

```
>>> import koordinates
```

Prepare to use the library by creating a client:

```
>>> client = koordinates.Client('labs.koordinates.com', 'MY_API_TOKEN')
```

Fetch all the Layer objects via the Layers & Tables API and iterate over them:

```
>>> for layer in client.layers.list():
...     print(layer.id)
>>>
```

Fetch filtered and sorted Layer objects via the Data Catalog API and iterate over them:

```
>>> for layer in client.catalog.list().filter(license__type='cc')\
...                                   .filter(type='layer')\
...                                   .order_by('created_at'):
...     print(layer.title)
>>>
```

The results of `.list()` returns a `Query` object which is chainable. It will only make an API request once it gets iterated over or `len()` is called on it.

Fetch a single Layer object:

```
>>> # Fetch the Layer with id = 123
>>> layer = client.layers.get(123)
>>> print(layer.title)
>>>
```

Make use of the hierarchy of data within a single object exposed as Python class instances via the library:

```
>>> # Fetch the Layer with id = 123 and extract the
>>> # data.crs value
>>> layer = client.layers.get(123)
>>> print(layer.data.crs)
>>>EPSG:2193
```

Create a new Layer from existing datasources:

```
>>> layer = koordinates.Layer()
>>> layer.name = "A Test Layer"
>>> layer.group = 999
>>> layer.data = koordinates.LayerData(datasources=[123456])
>>> layer = client.layers.create(layer)
>>> print(layer.url)
```

Publish multiple objects of various types:

```
>>> # Publish a number of items, in this case one
>>> # Table and one Layer
>>> publish = koordinates.publishing.Publish()
>>> publish.add_layer_version(1111)
>>> publish.add_table_version(2222)
>>> publish.strategy = publish.STRATEGY_TOGETHER
>>> publish = client.publishing.create(publish)
>>> print(publish.url)
```

Reimport an existing Layer from its previous data sources and create a new version:

```
>>> # Take the version with id=9999 of the Layer
>>> # with id = 8888 and reimport it
>>> layer = client.layers.get(8888)
>>> layer = layer.reimport()
```

Publish a specific version of a Layer:

```
>>> # Fetch the version with id=9999 of the Layer
>>> # with id = 8888 and publish it
>>> layer_version = client.layers.get(8888).get_version(9999)
>>> layer_version.publish()
```

Authentication

See the Token API documentation for details on creating API tokens for use with this library.

Once you have an API token, you can either pass it into the `koordinates.client.Client` object when you create it, or set it in the `KOORDINATES_TOKEN` environment variable.

```
# Pass token explicitly
client = koordinates.Client(host='labs.koordinates.com', token='abcdef1234567890abcdef')

# Token from environment variable KOORDINATES_TOKEN
client = koordinates.Client(host='labs.koordinates.com')
```

Tokens are specific to a Koordinates site. For example, a token created for `labs.koordinates.com` wouldn't be valid for another site, such as `koordinates.com`.

Tokens need to be created with scopes appropriate for the APIs you are utilising. For example, to query Sets you need a token with the `sets:read` scope, and to create or update a Set you need a token with the `sets:write` scope.

If a required scope isn't associated with the token, you will receive an `koordinates.exceptions.InvalidTokenScope` exception.

In addition to the scopes, the user or group owner of the token needs appropriate permissions for the actions they're attempting to take - for example, viewing a particular Set.

If required permissions aren't present, you will receive a `koordinates.exceptions.Forbidden` exception.

Creating tokens from the command line

The library includes a command line tool `koordinates-create-token` that can create API tokens.

```
usage: koordinates-create-token [-h] [--scopes SCOPE [SCOPE ...]]
                                [--referrers HOST [HOST ...]] [--expires DATE]
                                SITE EMAIL NAME

Command line tool to create a Koordinates API Token.

positional arguments:
  SITE                  Domain (eg. labs.koordinates.com) for the Koordinates
                        site.
  EMAIL                 User account email address
  NAME                  Description for the key

optional arguments:
  -h, --help            show this help message and exit
  --scopes SCOPE [SCOPE ...]
                        Scopes for the new API token
  --referrers HOST [HOST ...]
                        Restrict the request referrers for the token. You can
                        use * as a wildcard, eg. *.example.com
  --expires DATE        Expiry time in ISO 8601 (YYYY-MM-DD) format
```

The tool will prompt for the Koordinates account password corresponding to the email address, and request a new API token. The token will only be printed once, so you should copy/save it to a safe place.

Pagination

The library handles pagination of the results of `.list()` and related methods. These methods all act as generators and transparently fetch subsequent pages of results from the APIs in the background during iteration.

Limiting Results

Limiting the results of `.list()` and related methods is available via the python slicing syntax. Only the `[:N]` slicing style is supported. For example:

```
# Limit to a maximum of three results
for layer in client.layers.list()[:3]:
    print(layer)
```

Counting Results

In order to count the results of a query or list, use `len()`. For example:

```
print(len(client.layers.list()))
print(len(client.layers.filter(license='cc')))
```

This will perform a HEAD request unless a request has already been made (via a previous call to `len()` or iteration over the results), in which case the previous cached value will be returned.

Result Expansion

To prevent additional API requests, you can get the API to expand some relations and levels of detail in responses.

Not all properties or relations can be expanded. Refer to the Koordinates API documentation for details.

Important: Using expansions may have significant performance implications for some API requests.

To expand results in a list request:

```
for object in client.catalog.list().expand():
    # object will be a detailed model instance with
    # a full set of attributes
    print(object)
```

To expand an attribute in a get request:

```
set = client.sets.get(id=123, expand='items')
# the following get_items() call will use the .expand() results
# instead of making an additional request.
print(set, len(set.get_items()))
```

Contributing

Koordinates welcomes bug reports and contributions by the community to this module. This process is intended to be as easy as possible for both contributors and the Koordinates development team.

Testing

The client includes a suite of unit and functional tests. These should be used to verify that your changes don't break existing functionality, and that compatibility is maintained across supported Python versions. Tests run automatically on CircleCI for branch commits and pull requests.

To run the tests you need to:

```
$ pip install -r requirements.txt
$ tox
```

Patches

All patches should be sent as a pull request on GitHub, including tests and documentation where needed. If you're fixing a bug or making a large change the patch *must* include test coverage before it will be merged.

If you're uncertain about how to write tests, take a look at some existing tests that are similar to the code you're changing.

Release Process

This guide describes the process to release a new version of the library. In this example, `v0.0.0`. The library follows the Semantic Versioning guidelines, so select major, minor, and patch version numbers appropriately.

Preparations

1. Close or update all tickets for the next milestone on Github..
2. Update the *minimum* required versions of dependencies in `setup.py`. Update the *exact* version of all entries in `requirements.txt`.
3. Run **`tox`** from the project root. All tests for all supported Python versions must pass:

```
$ tox
[...]
________ summary ________
py27: commands succeeded
py34: commands succeeded
congratulations :)
```

Note: Tox will use the `requirements.txt` to setup the virtualenvs, so make sure you've updated it.

4. Build the Sphinx docs. Make sure there are no errors and undefined references.

```
$ make clean docs
```

5. Check the CircleCI build is passing.
6. Update the version number in `koordinates/__init__.py` and commit:

```
$ git commit -m 'Version 0.0.0 release' koordinates/__init__.py
```

Warning: Don't tag and push the changes yet so that you can safely rollback if you need change something!

7. Create a draft release in Github with a list of changes, acknowledgements, etc.

Build and release

1. Test the release process. Build a source distribution and test it:

```
$ python setup.py sdist
$ ls dist/
koordinates-0.0.0.tar.gz
```

Try installing them:

```
$ rm -rf /tmp/koordinates-sdist  # ensure clean state
$ virtualenv /tmp/koordinates-sdist
$ /tmp/koordinates-sdist/bin/pip install dist/koordinates-0.0.0.tar.gz
$ /tmp/koordinates-sdist/bin/python
>>> import koordinates
>>> koordinates.__version__
'0.0.0'
```

2. Create or check your accounts for the *test server <https://testpypi.python.org/pypi>* and PyPI. Update your `~/.pypirc` with your credentials:

```
[distutils]
index-servers =
    pypi
    test

[test]
repository = https://testpypi.python.org/pypi
username = <test username>
password = <test password>

[pypi]
repository = http://pypi.python.org/pypi
username = <production username>
password = <production password>
```

3. Upload the distributions for the new version to the test server and test the installation again:

```
$ python setup.py register -r test
$ python setup.py sdist upload -r test

$ rm -rf /tmp/koordinates-sdist  # ensure clean state
$ virtualenv /tmp/koordinates-sdist
$ /tmp/koordinates-sdist/bin/pip install -i https://testpypi.python.org/pypi --extra-index-url h
```

4. Check if the package is displayed correctly: https://testpypi.python.org/pypi/koordinates
5. Upload the package to PyPI and test its installation one last time:

```
$ python setup.py register -r pypi
$ python setup.py sdist upload -r pypi

$ rm -rf /tmp/koordinates-sdist  # ensure clean state
$ virtualenv /tmp/koordinates-sdist
$ pip install -U koordinates
```

6. Check the package is displayed correctly: https://pypi.python.org/pypi/koordinates

Post release

1. Push your changes:

```
    $ git tag -a v0.0.0 -m "Version 0.0.0"
    $ git push origin v0.0.0
```

2. Activate the documentation build for the new version.
3. Make the Github release public.
4. Update related Zendesk pages if necessary.

   If you are looking for information on a specific function, class or method, this part of the documentation is for you.

Developer Interface

Classes

- `catalog`
- `client`
- `layer`
- `license`
- `metadata`
- `publishing`
- `set`
- `token`
- `user`

Exceptions

- `exceptions`

CHAPTER 3

Support

Please report bugs as Github issues, or see user/contributing below if you wish to suggest an improvement or make a change. For general technical support for the APIs and libraries, please contact us via support.koordinates.com.

Index