woeid Documentation

Release 1

Renchen Sun

September 17, 2016

Contents

A Python interface for Yahoo GeoPlanet Web Services REST APIs. Python 2 and 3 are all supported!

Author	Renchen Sun
Email	sunrenchen@gmail.com
Github	https://github.com/Ray-SunR/
LinkedIn	https://ca.linkedin.com/in/renchensun

Contents:

CHAPTER 1

Installation & Testing

1.1 Installation

From PyPI

```
$ pip install woeid
```

From source

Download from pypi https://pypi.python.org/pypi/woeid/1.0.0

Download the latest *woeid* library from: https://github.com/Ray-SunR/woeid

Extract the source distribution and run:

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

1.2 Getting the code

The code is hosted at *Github* https://github.com/Ray-SunR/woeid

Check out the latest development version anonymously with:

```
$ git clone https://github.com/Ray-SunR/woeid.git
$ cd woeid
```

1.3 Testing

Navigate into test folder:

```
$ cd test
```

and run:

```
$ python test.py
```

CHAPTER 2

Getting Started

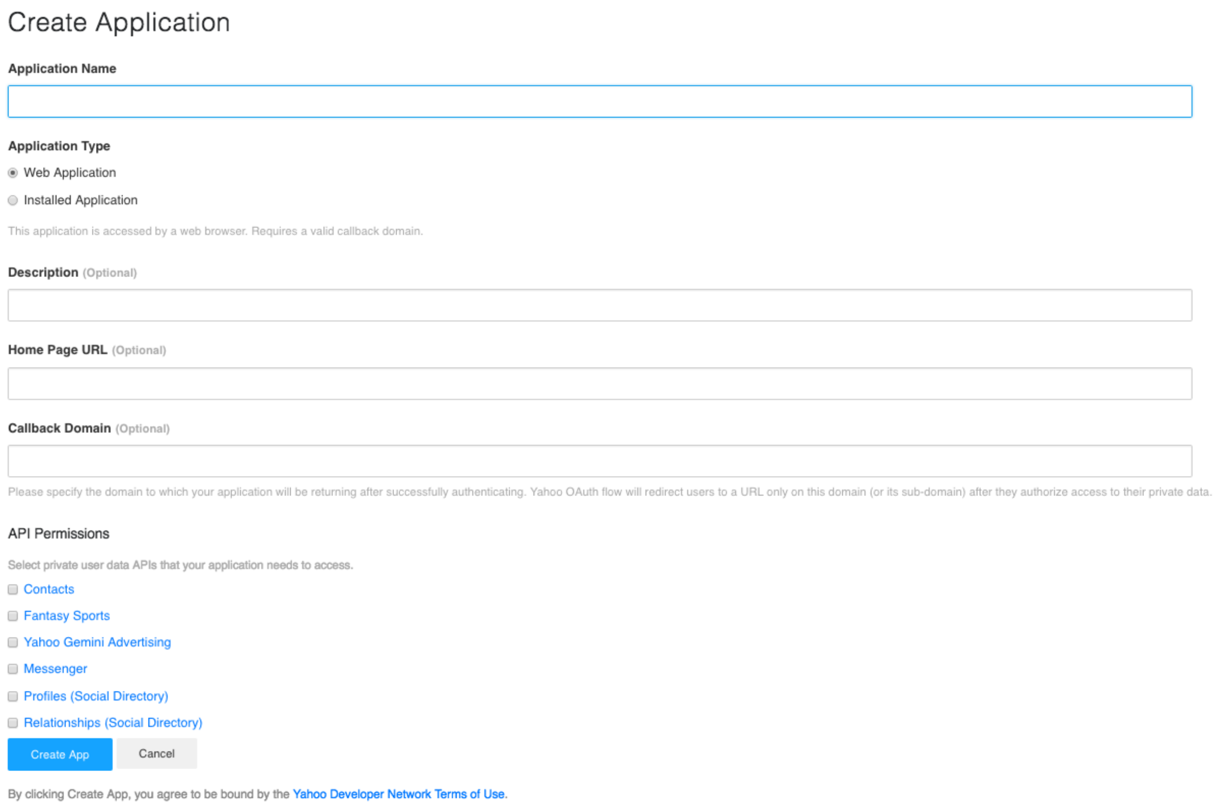
2.1 Getting your application tokens

This section is subject to changes made by Yahoo and may not always be completely up-to-date. If you see something change on their end, please create a new issue on Github or submit a pull request to update it.

In order to use the woeid API client, you first need to acquire the consumer key. The `consumer key` will be required in order to create a `woeid.Api` object.

2.1.1 Create your app

The first step in doing so is to create a Yahoo App. Click the "Create an App" button and fill out the fields on the next page.



Create Application

Application Name

Application Type

[● Web Application]

○ Installed Application

This application is accessed by a web browser. Requires a valid callback domain.

Description (Optional)

Home Page URL (Optional)

Callback Domain (Optional)

Please specify the domain to which your application will be returning after successfully authenticating. Yahoo OAuth flow will redirect users to a URL only on this domain (or its sub-domain) after they authorize access to their private data.

API Permissions

Select private user data APIs that your application needs to access.

☐ Contacts

☐ Fantasy Sports

☐ Yahoo Gemini Advertising

☐ Messenger

☐ Profiles (Social Directory)

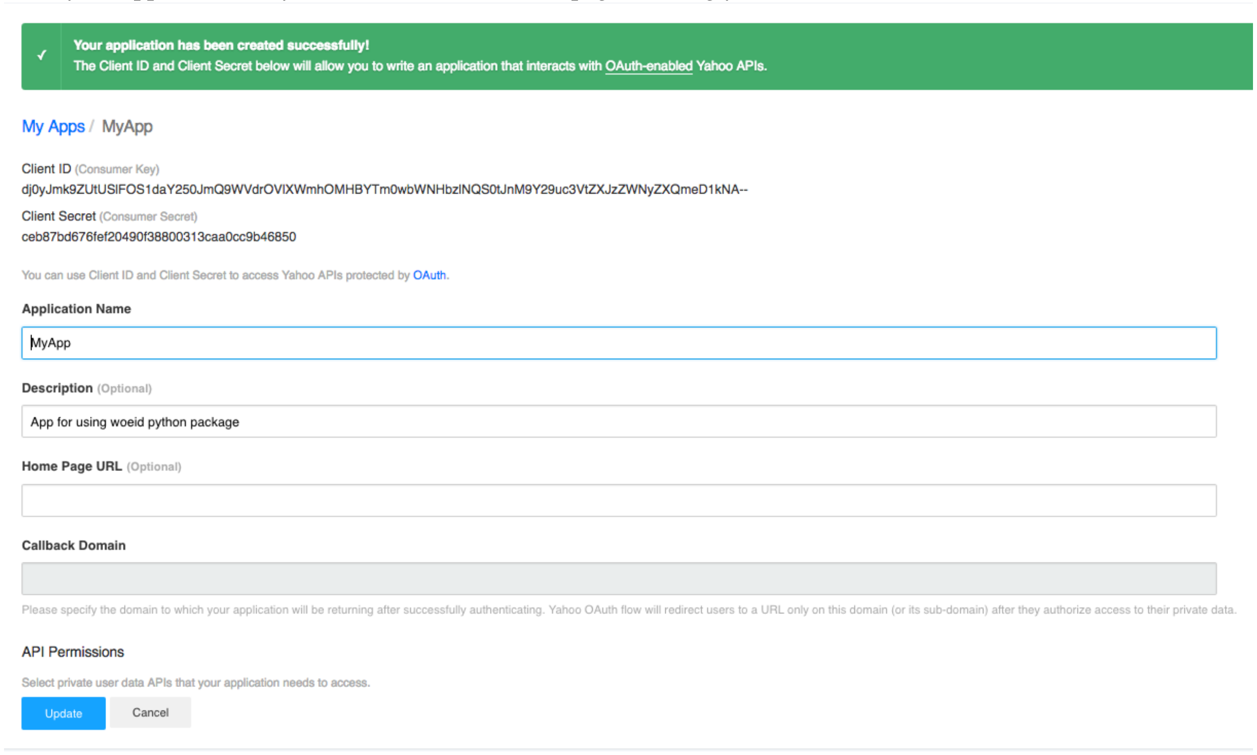
☐ Relationships (Social Directory)

By clicking Create App, you agree to be bound by the Yahoo Developer Network Terms of Use.

If there are any problems with the information on that page, Yahoo will complain and you can fix it. (Make sure to get the name correct - it is unclear if you can change this later.) On the next screen, you'll see the application that you created and some information about it:

2.1.2 Your app

Once your app is created, you'll be directed to a new page showing you some information about it.



✓ **Your application has been created successfully!**
The Client ID and Client Secret below will allow you to write an application that interacts with OAuth-enabled Yahoo APIs.

My Apps / MyApp

Client ID (Consumer Key)
dj0yJmk9ZUtUSlFOS1daY250JmQ9WVdrOVlXWmhOMHBYTm0wbWNHbzlNQS0tJnM9Y29uc3VtZXJzZWNyZXQmeD1kNA--

Client Secret (Consumer Secret)
ceb87bd676fef20490f38800313caa0cc9b46850

You can use Client ID and Client Secret to access Yahoo APIs protected by OAuth.

Application Name
MyApp

Description (Optional)
App for using woeid python package

Home Page URL (Optional)

Callback Domain

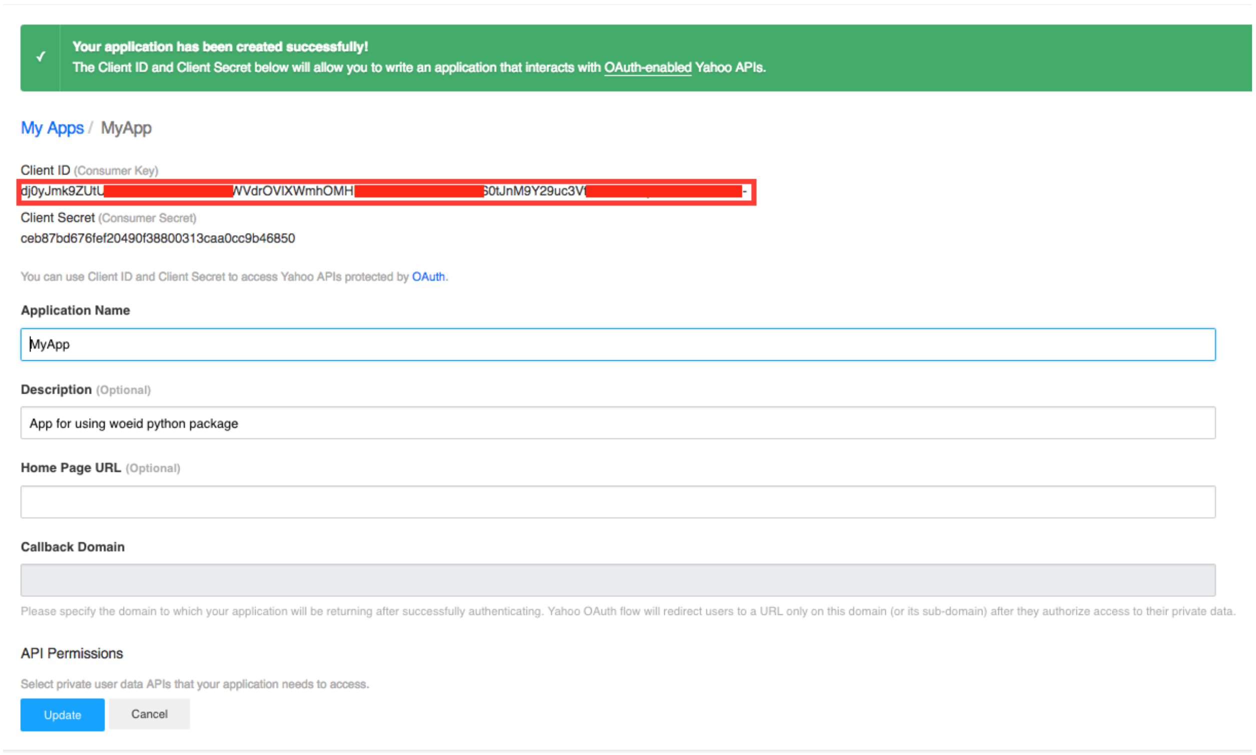
Please specify the domain to which your application will be returning after successfully authenticating. Yahoo OAuth flow will redirect users to a URL only on this domain (or its sub-domain) after they authorize access to their private data.

API Permissions

Select private user data APIs that your application needs to access.

2.1.3 Your Keys

The stirng which is ecnlosed in red rectangle is your `consumer key`.


✓ **Your application has been created successfully!**
The Client ID and Client Secret below will allow you to write an application that interacts with OAuth-enabled Yahoo APIs.

My Apps / MyApp

Client ID (Consumer Key)
dj0yJmk9ZUtU[redacted]WVdrOVlXWmhOMH[redacted]60tJnM9Y29uc3V[redacted]

Client Secret (Consumer Secret)
ceb87bd676fef20490f38800313caa0cc9b46850

You can use Client ID and Client Secret to access Yahoo APIs protected by OAuth.

Application Name
MyApp

Description (Optional)
App for using woeid python package

Home Page URL (Optional)

Callback Domain

Please specify the domain to which your application will be returning after successfully authenticating. Yahoo OAuth flow will redirect users to a URL only on this domain (or its sub-domain) after they authorize access to their private data.

API Permissions

Select private user data APIs that your application needs to access.

At this point, you can test out your application using the `consumer key` to instantiate a `woeid.Api(client_id='YOUR_CLIENT_ID')` as follows:

```
import woeid
api = woeid.Api(client_id=[consumer_key])
```

CHAPTER 3

Examples

3.1 Example 1: Create an `woeid` api object

```
import woeid
api = woeid.Api(client*id=`YOUR-CLIENT-ID`, select='long', format='xml')

# Specify the requesting language
api.Lang = 'zh*Hans'

# Specify the view
api.Select = 'short'

# Set the response format
api.Format = 'json'

# Set the maximum number of records returned
api.Count = 5
```

3.2 Example 2: Retrieving the Most Likely Place for a Given Place Name'

```
woeid.PrettyPrintResult(api.GetPlaces(q=u''))
```

3.3 Example 3: Retrieving the Five Most Likely Places for a Given Placename'

```
woeid.PrettyPrintResult(api.GetPlaces(q=u''))
```

3.4 Example 4: Retrieving All Places for a Given `place name` and `placetype`‘

```
woeid.PrettyPrintResult(api.GetPlaces(q='Long Island', typ=22, nd=True))
```

3.5 Example 5: Retrieving Places That Have the Given `woeids`‘

```
woeid.PrettyPrintResult(api.GetPlaces(woeid=[2488042, 2488836, 2486340]))
```

3.6 Example 6: Retrieving a Place Using a Given `woeid`‘

```
woeid.PrettyPrintResult(api.GetPlace(woeid=2507854))
```

3.7 Example 7: Retrieving a Place with a Given `woeid`, in *short* Representation

```
api.Select = 'short'
woeid.PrettyPrintResult(api.GetPlace(woeid=2507854))
```

3.8 Example 8: Retrieving the Parent Place of a Given `woeid`, as a `long` Representation

```
api.Select = 'long'
woeid.PrettyPrintResult(api.GetPlace(woeid=638242, parent=True))
```

3.9 Example 9. Retrieving the Ancestors for a Given `woeid`‘

```
woeid.PrettyPrintResult(api.GetPlace(woeid=12587712, ancestors=True))
```

3.10 Example 10. Retrieving a Place That is a `common` Ancestor of Two Places’

```
woeid.PrettyPrintResult(api.GetPlace(woeid=(2507854, 2380824), common=True))
```

3.11 Example 11. Retrieving a Place That is `common` Ancestor of Three Places’

```
woeid.PrettyPrintResult(api.GetPlace(woeid=(2488042, 2488836, 2486340), common=True))
```

3.12 Example 12. Retrieving All Continents’

```
woeid.PrettyPrintResult(api.GetContinents())
```

3.13 Example 13. Retrieving the Seas Adjacent to or Part of the Pacific Ocean’

```
woeid.PrettyPrintResult(api.GetSeas(place='Pacific Ocean'))
```

3.14 Example 14. Retrieving the Countries Within North America (NA)’

```
woeid.PrettyPrintResult(api.GetCountries(place='NA'))
```

3.15 Example 15. Retrieving the States Within the United States (US)’

```
woeid.PrettyPrintResult(api.GetStates(country='US'))
```

3.16 Example 16. Retrieving the Districts of Greater London’

```
woeid.PrettyPrintResult(api.GetDistricts(county='Greater London'))
```

3.17 Example 17. Retrieving the WOEID and FIPs Code for a Given ISO Code’

```
woeid.PrettyPrintResult(api.GetConcordance(namespace='iso', id='CA-BC'))
```

3.18 Example 18. Retrieving a Partial Collection of Place Types’

```
woeid.PrettyPrintResult(api.GetPlacetypes(typ=[0,2,22,37,38,15,16]))
```

CHAPTER 4

Modules Documentation

4.1 API

A library that provides a Python interface to the Yahoo GeoPlanet API

class `woeid.api.`**`Api`**(*client_id*, *base_url='http://where.yahooapis.com/v1/'*, *select='long'*, *format='json'*, *lang='en-us'*, *count=0*, *start=0*)

Bases: `object`

A python interface into the Woeid API

Example usage: To create an instance of the woeid.Api class. Note: you will need to initialize it with your client id (consumer key) which you can obtain from Yahoo at https://developer.yahoo.com. The detailed guide can be found in the Getting Started section.

```
>>> import woeid
>>> api = woeid.Api(client_id='client_key')
```

Args:

client_id(`str`): Your yahoo client id (consumer key). Note: this is not the id of your app!

base_url(`str`, optional): Specify the base url for making requests. (You probably wouldn't want to change it)

select(`str`, optional): Specify how you would like your result to return. Only *short* or *long* are accepted.

format(`str`, optional): Specify which format you would like to receive as a response format. Three values are accepted: *json*, *xml*, *geojson*.

lang(`str`, optional): Specify the language you would like the return names to be be shown. RFC 4646 language codes are accepted.

count(`str` or `int`, optional): Specify the maximum number of results to return. A count of 0 is interpreted as *no maximum* (all resources)

start(`str` or `int`, optional): Skip the first N results.

`BaseUrl`

The base url

Getter Get the base url

Setter Set the base url

Type `str`

ClientId
The client id

Getter Get the client id

Setter Set the client id

Type `str`

Count
The format code

Getter Get the maximum number of results to return.

Setter Set the maximum number of results to return. A value of 0 is interpreted as no maximum (all resources)

Type `int` or `str`

Format
The response format code

Getter Get the response format code

Setter Set the response format code. Supported values are: 'json', 'xml', 'geojson'

Type `str`

GetConcordance(*namespace*, *id*)
Returns information that maps dentifiers (codes) from other providers to WOEIDs. The supported namespace and id values are provided in https://developer.yahoo.com/geo/geoplanet/guide/api-reference.html#api-concordance

Args:

namespace(`str`): A namespace string

id(`str`): An id string

Returns: Response content in `bytes`

GetContinents()
Returns a collection of places that are continents.

Returns: Response content in `bytes`

GetCounties(*state*)
Returns a collection of places that are second-level administrative areas (counties) within a top-level administrative area (state).

Args:

state(`str`): A state string.

Returns: Response content in `bytes`

GetCountries(*place=None*)
Returns a collection of places that are countries if *place* is not provided. Returns a collection of places that are countries and are part of or adjacent to the specified continent or ocean.

Args:

place(`str`, optional): A place string

Returns: Response content in `bytes`

GetDistricts(*county*)

Returns a collection of places that are third-level administrative areas (districts) within a second-level administrative area (county).

Args:

county(`str`): A county string

Returns: Response content in `bytes`

GetOceans()

Returns a collection of places that are oceans.

Returns: Response content in `bytes`

GetPlace(*woeid*, *degree=None*, *typ=None*, *nd=None*, *parent=False*, *ancestors=False*, *belongtos=False*, *neighbors=False*, *children=False*, *siblings=False*, *descendants=False*, *common=False*)

Return a place object that matches a specified woeid or with a specified relationship specifier (parent, ancestors, belongtos, neightbors, children, siblings, descendants, common) or with filters (degree, typ, nd).

Args:

woeid(`str` or `int` or `tuple` or `list`): The unique place specifier.

degree(`int` or `str`, optional): *.degree* specifier which represents the degree to which two places are neighborhoods. Only consider valid if either *neighbors* or *children* filters are set.

typ(`str` or `int` or `list`, optional): *.type* specifier which is used to specify placetypes. Up to ten place types may be provided. Only consider valid if either *belongtos*, *children* or *placetypes* are set.

nd(boolean, optional): *$and* specifier which is used to join two fitlers together.

parent(`boolean`, optional): A relationship specifier used to return a parent place of a given woeid.

ancestors(`boolean`, optional): A relationship specifier used to return one or more acestors of a place of a given woeid.

belongtos(`boolean`, optional): A relationship specifier used to return a collection of places that have a place as a child or descendant (child of a child).

neighbors(`boolean`, optional): A relationship specifier used to return a collection of places that neighbor of a place.

children(`boolean`, optional): A relationship specifier used to return a collection of places that are children of a place.

siblings(`boolean`, optional): A relationship specifier used to return a collection of places that are siblings of a place.

descendants(`boolean`, optional): A relationship specifier used to return a collection of places that are in the child hierarchy (the child, the child of child, etc).

common(`boolean`, optional): A relationship specifier used to return the common ancestor of both places.

Returns: Response content `in ``bytes```

GetPlaces(*q=None*, *woeid=None*, *typ=None*, *nd=None*)

Returns a collection of places that match a specified place name, and optionally, a specified place type. The resources in the collection are long representations of each place (unless short representations are explicitly requested).Supported Filters *.q*, *.type*, *$and*, *.woeid*.

Args:

q(`str` or `tuple`, optional): Specify a place name to search for or a tuple that has a place name and a focus. This filter is mutually exclusive with the *woeid* filter. The specified place can be any unicode characters. Focus can be either an ISO-3166-1 country code or a WOEID. For a "startswith" filter, specify the place as a string followed by an asterisk (*).

woeid(`list(str)` or `list(int)`, optional): Specify a *Where On Earth Identifier* (*woeid*). Up to ten WOEIDs may be specified. This filter is mutually exclusive with the *q* filter. Example: woeid=(1,2,3)

typ(`list(str)` or `list(int)` or `int`, optional): Specify one or more place type codes (https://developer.yahoo.com/geo/geoplanet/guide/concepts.html#placetypes). Up to ten place type codes or names may be provided.

nd(`boolean`, optional): Specify a join operations on two filters. Example:

```
>>> import woeid
>>> api = woeid.Api(client_id='YOUR_CLIENT_ID')
>>> ret = api.GetPlaces(q='StringField', typ=22, nd=True)
```

Returns: Response content in `bytes`

`GetPlacetype`(*typ*, *country*)

Returns a place type resource if only *typ* is specified. Returns information about a single placetype known by its code. The placetype name returned will be country-specific, and may vary depending upon the language requested.

Args:

typ(`str` or `list` or `int`): A type filter.

country(`str`): A country code string

Returns: Response content in `bytes`

`GetPlacetypes`(*country=None*, *typ=None*)

Returns the complete collection of place types supported in GeoPlanet if neither *country* or *typ* is specified. Returns information about all the known placetypes. The placetype name returned will be country-specific, and may vary upon the language requested if country is specified.

Args:

country(`str`, optional): A country code string

typ(`str` or `list` or `int`, optional): A type filter.

Returns: Response content in `bytes`

`GetSeas`(*place=None*)

Returns a collection of places that are seas if *place* is not provided. Returns a collection of places that are seas and are part of or adjacent to the specified continent or ocean if *place* is specified.

Args: place(`str`, optional):

Returns: Response content in `bytes`

`GetStates`(*country*)

Returns a collection of places that are top-level administrative areas (states) within a country.

Args:

country(`str`): A country string

Returns: Response content in `bytes`

`Lang`
The language code (RFC 4646) used for making requests

Getter Get the language code used for making requests

Setter Set the language code used for making requests. Possible language codes are defined in RFC 4646

Type `str`

`Select`
The result representation mode

Getter Get the result representation mode

Setter Set the result representation mode. Supported values are: *long*, *short*

Type `str`

`Start`
The number of records to skip

Getter Get the number of records to skip

Setter Set the number of records to skip

Type `int` or `str`

4.2 Modules

class `woeid.modules.`**`Filters`**(*q=None*, *woeid=None*, *typ=None*, *degree=None*, *aand=None*)
A class that encapsulates all filters Args:

q(`str` or `tuple`, optional): Specify a place name to search for or a tuple that has a place name and a focus. This filter is mutually exclusive with the *woeid* filter. The specified place can be any unicode characters. Focus can be either an ISO-3166-1 country code or a WOEID. For a “startswith” filter, specify the place as a string followed by an asterisk (*).

woeid(`list``(``str`) or `list``(``int`), optional): Specify a *Where On Earth Identifier* (*woeid*). Up to ten WOEIDs may be specified. This filter is mutually exclusive with the *q* filter. Example: woeid=(1,2,3)

typ(`list``(``str`) or `list``(``int`) or `int`, optional): Specify one or more place type codes (https://developer.yahoo.com/geo/geoplanet/guide/concepts.html#placetypes). Up to ten place type codes or names may be provided.

degree(`int` or `str`, optional): *.degree* specifier which represents the degree to which two places are neighborhoods. Only consider valid if either *neighbors* or *children* filters are set.

nd(`boolean`, optional): Specify a join operations on two filters. Example:

```
>>> import woeid
>>> api = woeid.Api(client_id='YOUR_CLIENT_ID')
>>> ret = api.GetPlaces(q='StringField', typ=22, nd=True)
```

`HasAnd`()
Return if the filter object has *$and* filter

`HasDegree`()
Return if the filter object has *.degree* filter

HasQ()
Return if the filter object has *.q* filter.

HasType()
Return if the filter object has *.type* filter

HasWoeid()
Return if the filter object has *.woeid* filter.

IsValid()

class `woeid.modules.`**Relationships**(*parent=False*, *ancestors=False*, *belongstos=False*, *neighbors=False*, *siblings=False*, *children=False*, *descendants=False*, *common=False*)
"A class that encapsulates all relationships

Args:

parent(`boolean`, optional): A relationship specifier used to return a parent place of a given woeid.

ancestors(`boolean`, optional): A relationship specifier used to return one or more acestors of a place of a given woeid.

belongtos(`boolean`, optional): A relationship specifier used to return a collection of places that have a place as a child or descendant (child of a child).

neighbors(`boolean`, optional): A relationship specifier used to return a collection of places that neighbor of a place.

children(`boolean`, optional): A relationship specifier used to return a collection of places that are children of a place.

siblings(`boolean`, optional): A relationship specifier used to return a collection of places that are siblings of a place.

descendants(`boolean`, optional): A relationship specifier used to return a collection of places that are in the child hierarchy (the child, the child of child, etc).

common(`boolean`, optional): A relationship specifier used to return the common ancestor of both places.

4.3 Utilities

class `woeid.utility.`**ResponseCheck**(*code*)
A utility class reponsible for chekcing the reponse code and raise corresponding error

Args:

code(`int`): The response code in integer

class `woeid.utility.`**Utility**

static BuildParams(*appid*, *format='json'*, *select='short'*, *lang='en-us'*)
For constructing a parameter dictionary.

Returns: Parameters dictionary

static BuildUrls(*url*, *path_elements*, *extra_params=None*, *extra_woeid=None*, *filters=None*, *relationships=None*, *count=None*, *start=None*)
An utility class reponsible for building the url string

Args:

path_elements(`list``(``str`)): A list of paths that will be appended to the base url.

extra_params(`dict`, optional): A dictionary representing the parameters for making the url.

extra_woeid(`list``(``int`) or `list``(``str`), optional): This is useful when the *common* filter has been set. Aiming for making urls such as *1234/common/3456/73923*

filters(`Filters`, optional): A *Filter* object

relationships(`Relationships`, optional): A *Relationship* object

count(`int`, optional): Specify the maximum number of results to return. A count of 0 is interpreted as *no maximum* (all resources)

start(`int`, optional): Skip the first N results.

Returns: A valid url containing all queries, filters, parameters.

static `EncodeParameters`(*parameters*)

Return a string in key=value&key=value form. Values of None are not included in the output string.

Args:

parameters (`OrderedDict`): dictionary of query parameters to be converted into a string for encoding and sending to Twitter.

Returns: A URL-encoded string in "key=value&key=value" form

static `GetLastRequestUrl`()

An utility function for fetching the most recent requesting url

Returns: Last request url in `str`

static `GetLastResponseCode`()

A utility function for fetching the most recent reponse code

Returns: Last response code in `int`

static `MakeRequest`(*url*)

An utility function for making url requests. Will do response check

Returns: Response content in `bytes`

static `PrettyPrintResult`(*bts*)

An utility function for pretty printing the result with indentation and new lines.

Returns: None

CHAPTER 5

About me

Hello! My name is Renchen. I am a 24-year-old Full-Stack Software Developer from Vancouver BC, Canada.

I consider myself:

- A heavy `vimer`
- A command line lover
- Most capable of `C/C++` and `Python` programming. Also capable of programming in `Java`, `Javascript`, `C#`, `ObjC`
- Comfortable writing portable codes for different platforms (`Windows`, `Linux`, `MacOS`, `iOS`, `Android`)
- Passionate about server-side development (`Node`, `Meteor`)
- An expert in `OpenXml` standards

CHAPTER 6

Introduction

This library provides a pure Python interface for the Yahoo GeoPlanet API. It works with Python 2.7+ and Python 3.

Yahoo! GeoPlanetTM is designed to bridge the gap between the Real and Virtual worlds by providing the Internet with an open, comprehensive, and intelligent infrastructure for geo-referencing data on Earth's surface.

In practical terms, Yahoo! GeoPlanet is a resource for managing all geo-permanent named places on earth. It provides the Geographic Developer Community with the vocabulary and grammar to describe the world's geography in an unequivocal, permanent, and language-neutral manner, and is designed to facilitate spatial interoperability and geographic discovery. Developers looking to geo-enable their applications can use GeoPlanet to traverse the spatial hierarchy, identify the geography relevant to their users and their business, and in turn, unambiguously geotag, geotarget, and geolocate data across the Web.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index