pyql-weather Documentation

Publicación 0.2

Alex Dzul

07 de February de 2016

Índice general

Contenido:

CAPÍTULO 1

Instalación Pyql-Weather

1.1 Instalación

`pyql-weather` está escrita 100 % en Python y por tal motivo no se requiere la instalación de ninguna otra dependencia. Nuestra recomendación es que instales la librería en un entorno virtual

1. Creamos nuestro entorno con virtualenv:

```
$ virtual testenv
```

2. Activamos el entorno virtual:

```
$ source testenv/bin/activate
```

Ya que tenemos activado nuestro entorno virtual, podemos instalar `pyql-weather` de la siguiente manera:

1.1.1 Vía PIP

Instalarlo es muy fácil con la herramienta pip:

```
$ pip install pyql-weather
```

1.1.2 Instalación manual

También puedes descargar el código fuente directamente desde github en este link:

```
https://github.com/alexdzul/pyql-weather/tree/0.2.x
```

y después ejecutar el script de instalación:

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

Bien! Ya que tienes instalada nuestra librería te invitamos a continuar leyendo la documentación y encontrar ejemplos muy interesantes.

CAPÍTULO 2

Ejemplos pyql.geo

"pyql-weather" es una gran herramienta que nos permite interactuar fácilmente con los servicios del clima de Yahoo. Aquí presentaremos algunos escenarios en el que se puede utilizar los objetos almacenados en `pyql.geo`.

Nota: Para mayor información consulte la guía oficial de las tablas "geo" en https://developer.yahoo.com/geo/geoplanet/guide/yql-tables.html

2.1 Continents

Para poder completar estos ejemplos es necesario importar el objeto tipo *Continent*:

```
from pyql.geo.continents import Continent
```

2.1.1 Enlistar todos los Continentes

1. Instanciamos un objeto del tipo Continent utilizando la función filter la cual nos devolverá una lista de elementos:

```
    continents = Continent.filter()
```

3. Recorremos la lista de objetos y podremos acceder a las propiedades de cada continente de la siguiente manera:

```
    for cont in continents:
        print(cont.name)
```

2.1.2 Obtener un Continente en específico

Iniciamos un nuevo objeto pero en esta ocasión utilizaremos la función *get* la cual nos devuelve solamente un objeto y no una lista:

```
continent = Continent.get(name="Africa")

print(continent.name)
print(continent.lang)
print(continent.woeid)
```

Resultado:

```
Africa
en-US
24865670
```

2.2 PlaceFinder

Para poder completar estos ejemplos es necesario importar el objeto tipo *PlaceFinder*:

```
from pyql.geo.placefinder import PlaceFinder
```

2.2.1 Obtener información de un lugar vía Latitude / Longitude

Realizamos la búsqueda de la siguientea manera:

```
latitude = "20.632784"
longitude = "-103.359375"
lat_long = "{0},{1}".format(latitude, longitude)

finder = PlaceFinder.get(text=lat_long, gflags="R")
place_info = "{0}, {1} ZIP: {2} | WOEID:{3}".format(finder.city,
                                                    finder.country,
                                                    finder.uzip,
                                                    finder.woeid)
print(place_info)
```

Resultado:

```
Guadalajara, Mexico ZIP: 44940 | WOEID:24553135
```

2.2.2 Obtener lugar pasando parte de su nombre

Podemos de igual manera pasar una parte del nombre del lugar y obtener el mismo resultado:

```
find = PlaceFinder.get(text="sfo")
print(find.quality)
print(find.name)
print(find.line1)
print(find.line2)
```

2.3 Oceans

Para poder completar estos ejemplos es necesario importar el objeto tipo *Ocean*:

```
from pyql.geo.oceans import Ocean
```

2.3.1 Obtener el listado completo de océanos

Obtener todos los océanos e imprimir sus números *woeid*:

```
oceans = Ocean.filter()
for ocean in oceans:
    info = "{0}, woeid: {1}".format(ocean.name, ocean.woeid)
    print(info)
```

Resultado:

```
Atlantic Ocean, woeid: 55959709
Southern Ocean, woeid: 55959676
Indian Ocean, woeid: 55959675
Pacific Ocean, woeid: 55959717
Arctic Ocean, woeid: 55959707
```

2.3.2 Obtener la información de un océano en específico:

Para obtener solamente un objeto del tipo *Ocean* se requiere utilizar la función *get* seguida de los criterios de búsqueda:

```
ocean = Ocean.get(name="Atlantic Ocean")

print(ocean.name)
print(ocean.woeid)
print(ocean.lang)
```

2.4 States

1. Para poder consultar los estados de una región, es requisito construir el objeto pasando como argumento la llave "place" la cual es el nombre del País que deseamos conocer.

2. Para poder completar estos ejemplos es necesario importar el objeto tipo *State*:

```
    from pyql.geo.states import State
```

2.4.1 Obtener todos los estados de México

Generamos la consulta con el place "Mexico" e imprimimos el resultado en un ciclo for:

```
states = State.filter(place="Mexico")
for state in states:
    print(state.name)
```

Resultado:

```
Zacatecas
Aguascalientes
San Luis Potosi
Nuevo Leon
Durango
Guanajuato
Nayarit
Jalisco
Tamaulipas
Queretaro de Arteaga
Coahuila de Zaragoza
Hidalgo
```

```
Michoacan de Ocampo
Mexico
Colima
Distrito Federal
Sinaloa
Tlaxcala
Morelos
Puebla
Veracruz-Llave
Chihuahua
Guerrero
Oaxaca
Baja California Sur
Tabasco
Sonora
Chiapas
Campeche
Yucatan
Baja California
Quintana Roo
```

2.4.2 Obtener información del estado de Yucatán, México

Para este ejemplo utilizamos la función *get* en lugar de *filter* para que el resultado sea solamente un objeto y no necesiten recorrerlo en un ciclo:

```
state = State.get(place="Mexico", name="Yucatan")
print(state.name, state.woeid)
```

2.5 Sea

Para poder completar estos ejemplos es necesario importar el objeto tipo *Sea*:

```
from pyql.geo.seas import Sea
```

2.5.1 Listado completo de mares

La forma más sencilla de obtener el listado completo de todos los mares del mundo es utilizando la función **filter** sin pasar ningún parámetro como filtro:

```
sea_list = Sea.filter()
for sea in sea_list:
    print("Nombre: {0} | woeid: {1}".format(sea.name, sea.woeid))
```

Resultado:

```
Nombre: Mediterranean Sea | woeid: 55959718
Nombre: Gulf of Aqaba | woeid: 55959677
Nombre: Red Sea | woeid: 55959678
Nombre: English Channel | woeid: 55959688
Nombre: Irish Sea | woeid: 28742112
Nombre: Black Sea | woeid: 55959689
Nombre: North Sea | woeid: 55959673
```

```
Nombre: Arabian Sea | woeid: 55959681
Nombre: Persian Gulf | woeid: 55959679
Nombre: Baltic Sea | woeid: 55961436
Nombre: Gulf of Oman | woeid: 55959680
Nombre: Norwegian Sea | woeid: 55959691
Nombre: Denmark Strait | woeid: 55959692
Nombre: Greenland Sea | woeid: 55959685
Nombre: Caribbean Sea | woeid: 55959687
Nombre: Labrador Sea | woeid: 55959684
Nombre: Barents Sea | woeid: 55961429
Nombre: Bay of Bengal | woeid: 55959674
Nombre: Davis Strait | woeid: 55959683
Nombre: Gulf of Mexico | woeid: 55959686
Nombre: Andaman Sea | woeid: 55959713
Nombre: Hudson Bay | woeid: 55959682
Nombre: Strait of Malacca | woeid: 55959714
Nombre: Nares Strait | woeid: 55959690
Nombre: Kara Sea | woeid: 55961432
Nombre: Gulf of Thailand | woeid: 55959699
Nombre: Java Sea | woeid: 55959715
Nombre: Gulf of Tonkin | woeid: 55959700
Nombre: South China Sea | woeid: 55959698
Nombre: Bali Sea | woeid: 55960587
Nombre: Flores Sea | woeid: 55960586
Nombre: Savu Sea | woeid: 55960588
Nombre: Laptev Sea | woeid: 55961431
Nombre: Taiwan Strait | woeid: 55959701
Nombre: Bohai Sea | woeid: 55959695
Nombre: Timor Sea | woeid: 55959706
Nombre: Yellow Sea | woeid: 55959696
Nombre: East China Sea | woeid: 55959694
Nombre: Korea Strait | woeid: 55959697
Nombre: Arafura Sea | woeid: 55959716
Nombre: Great Australian Bight | woeid: 55959703
Nombre: Beaufort Sea | woeid: 55959708
Nombre: Gulf of Carpentaria | woeid: 55959705
Nombre: East Sea/Sea of Japan | woeid: 55959693
Nombre: Gulf of Alaska | woeid: 55959710
Nombre: Coral Sea | woeid: 55959704
Nombre: Sea of Okhotsk | woeid: 55961433
Nombre: East Siberian Sea | woeid: 55961430
Nombre: Chukchi Sea | woeid: 55961435
Nombre: Tasman Sea | woeid: 55959702
Nombre: Bering Sea | woeid: 55961434
```

2.5.2 Búsqueda con filtros.

Ahora veremos un ejemplo similar al anterior pero aplicando un filtro. Realizaremos la búsqueda de los mares del continente de África:

```
african_seas = Sea.filter(place="Africa")

for sea in african_seas:
    print(sea.name)
```

Resultado:

```
Red Sea
Gulf of Aqaba
Mediterranean Sea
Arabian Sea
```

2.6 District

Devuelve información sobre los lugares que son áreas administrativas de tercer nivel dentro de un país. Tenga en cuenta que el término "**distrito**" se refiere a cualquier área administrativa que subdivide una zona administrativa de segundo nivel, como los distritos, comunas, municipios.

Para poder completar estos ejemplos es necesario importar el objeto tipo *District*:

```
from pyql.geo.districts import District
```

2.6.1 Listado de distritos

Para realizar una búsqueda de los distritos de "Greater London" escribimos lo siguiente:

```
districts = District.filter(place="Greater London")

for element in districts:
    print(element.name)
```

Resultado:

```
City of Westminster
City of London
Royal Borough of Kensington and Chelsea
London Borough of Camden
London Borough of Islington
London Borough of Lambeth
London Borough of Southwark
London Borough of Hammersmith and Fulham
London Borough of Hackney
London Borough of Tower Hamlets
London Borough of Wandsworth
London Borough of Haringey
London Borough of Lewisham
London Borough of Brent
London Borough of Merton
London Borough of Newham
London Borough of Waltham Forest
London Borough of Greenwich
London Borough of Barnet
London Borough of Ealing
London Borough of Richmond upon Thames
London Borough of Enfield
London Borough of Hounslow
London Borough of Redbridge
London Borough of Sutton
London Borough of Croydon
London Borough of Harrow
Royal Borough of Kingston upon Thames
London Borough of Barking and Dagenham
```

```
London Borough of Bromley
London Borough of Bexley
London Borough of Hillingdon
London Borough of Havering
```

2.7 Places

Este objeto retorna un **lugar** o **lugares** que concuerden con los criterios de búsqueda especificada. Para poder completar estos ejemplos es necesario importar el objeto tipo *Place*:

```
from pyql.geo.places import Place
```

2.7.1 Filtro de lugares por nombre

En este ejemplo realizaremos la búsqueda de todos los lugares que tengan como nombre "Yucatán":

```
places = Place.filter(text="Yucatan")


for place in places:
    print("{0}: {1} {2}".format(place.place_type_name.content,
                                place.name,
                                place.timezone.content))
```

Resultado:

```
State: Yucatan America/Merida
Town: Yucatan America/Chicago
Town: Yucatan America/Chicago
Suburb: Yucatan America/Mexico_City
Suburb: Yucatan America/Monterrey
Suburb: Yucatan America/Merida
```

En el resultado anterior puede notar que la librería nos permite conocer el tipo de objeto que Yahoo ha encontrado: State, Town, Suburb.

2.8 Countries.

Este objeto retorna la información de elementos que son Países o territorios independientes. Para poder completar estos ejemplos es necesario importar el objeto tipo `Country`:

```
from pyql.geo.countries import Country
```

2.8.1 Listado de Países.

En este ejemplo realizaremos la búsqueda de todos los países que se encuentren en "North America":

```
countries = Country.filter(place="North America")

for country in countries:
    print("{0}: {1}".format(country.name, country.place_type_name.content))
```

Resultado:

```
Aruba: Country
Antigua and Barbuda: Country
Anguilla: Territory
Barbados: Country
Bermuda: Territory
The Bahamas: Country
Belize: Country
Canada: Country
Cayman Islands: Territory
Costa Rica: Country
Cuba: Country
Dominica: Country
Dominican Republic: Country
El Salvador: Country
Grenada: Country
Greenland: Province
Guadeloupe: Overseas Region
Guatemala: Country
Haiti: Country
Honduras: Country
Jamaica: Country
Martinique: Overseas Region
Montserrat: Territory
Mexico: Country
Nicaragua: Country
Panama: Country
Puerto Rico: Territory
Saint Pierre and Miquelon: Territory
Saint Kitts and Nevis: Country
St. Lucia: Country
Trinidad and Tobago: Country
Turks and Caicos Islands: Territory
United States: Country
Saint Vincent and the Grenadines: Country
British Virgin Islands: Territory
US Virgin Islands: Territory
United States Minor Outlying Islands: Territory
Saint Barthelemy: Overseas Collectivity
Saint-Martin: Overseas Collectivity
```

2.9 Counties

Con este objeto podemos encontrar información de la división de segundo nivel de los Países. Para poder completar estos ejemplos es necesario importar el objeto tipo `Countie`:

```
from pyql.geo.counties import Countie
```

2.9.1 Lista filtrada

A continuación presentamos un ejemplo de filtro de los municipios del estado de “Tabasco” del país “México”:

```
localidades = Countie.filter(place="Tabasco")
for local in localidades:
    print("{0}-{1}".format(local.woeid, local.name))
```

Resultado:

```
12601626-Macuspana
12601623-Jalapa
12601618-Centro
12601625-Jonuta
12601617-Centla Municipality
12601627-Nacajuca
12601629-Tacotalpa
12601630-Teapa
12601624-Jalpa de Mendez
12601620-Cunduacan
12601628-Paraiso
12601619-Comalcalco
12601621-Emiliano Zapata
12601622-Huimanguillo
12601616-Cardenas
12601615-Balancan
12601631-Tenosique
```

2.10 PlaceType

Objeto que nos permite conocer todos los tipos de datos que maneja Yahoo YQL. Para poder completar estos ejemplos es necesario importar el objeto tipo `PlaceType`:

```
from pyql.geo.placetypes import PlaceType
```

2.10.1 Obtener todos los tipos

Podemos obtener todos los tipos disponibles si utilizamos la función "filter" sin pasar ningún parámetro:

```
for tipo in tipos:
    print ("{0}: {1}".format(tipo.place_type_name.content,
                             tipo.place_type_description))
```

Resultado:

```
Undefined: An undefined place
Town: A populated settlement such as a city, town, village
State: One of the primary administrative areas within a country
County: One of the secondary administrative areas within a country
Local Administrative Area: One of the tertiary administrative areas within a country
Postal Code: A partial or full postal code
Country: One of the countries or dependent territories defined by the ISO 3166-1 standard
Island: An island
Airport: An airport
Drainage: A water feature such as a river, canal, lake, bay, ocean
```

CAPÍTULO 3

Ejemplos pyql.weather

"pyql-weather" es una gran herramienta que nos permite interactuar fácilmente con los servicios del clima de Yahoo. Aquí presentaremos algunos escenarios en el que se puede utilizar los objetos almacenados en pyql.weather

3.1 Forecast

Para poder completar estos ejemplos es necesario importar el objeto tipo *Forecast*:

```
from pyql.weather.forecast import Forecast
```

Posteriormente debemos instanciar un objeto del tipo *Forecast* utilizando la función *get* pasándole el parámetro *woeid* y *u*:

```
my_woeid = 24553135
forecast = Forecast.get(woeid=my_woeid, u="c")
```

Nota: El parámetro woeid es el "Where on Earth Identifiers" del lugar que deseamos obtener y el segundo parámetro "u" nos permite definir si la unidad de medida del clima deseada es en grados **Celsius** (" c ") o **Fahrenheit** (" f ").

Ya que tenemos este objeto en memoria ahora podemos utilizarlo para conocer todos los detalles del pronóstico del clima:

3.1.1 Location

Para obtener más detalle del lugar que estamos consultando el pronóstico del clima podemos utilizar lo siguiente:

```
print(forecast.location.city)
print(forecast.location.region)
print(forecast.location.country)
```

Resultado:

```
Guadalajara
JA
Mexico
```

3.1.2 Astronomía

Podemos consultar los datos de la salida y puesta del sol de una manera muy sencilla:

```
print("Salida del sol: {0}".format(forecast.astronomy.sunrise))
print("Puesta del sol: {0}".format(forecast.astronomy.sunset))
```

Resultado:

```
Salida del sol: 7:24 am
Puesta del sol: 8:14 pm
```

3.1.3 Atmósfera

Los datos atmosféricos pueden ser encontrados en la propiedad **atmosphere** y la información disponible es la siguiente:

```
print("************** Atmósfera **********************")
print("Humedad: {0}".format(forecast.atmosphere.humidity))
print("Presión: {0}".format(forecast.atmosphere.pressure))
print("Incremento: {0}".format(forecast.atmosphere.rising))
print("Visibilidad: {0}".format(forecast.atmosphere.humidity))
print("*******************************************")
```

Resultado:

```
************** Atmósfera **********************
Humedad: 49
Presión: 1015.92
Incremento: 0
Visibilidad: 49
*******************************************
```

3.1.4 Viento

Para obtener la información referente al viento podemos realizar lo siguiente:

```
print("Enfriamiento: {0}".format(forecast.wind.chill))
print("Dirección: {0}".format(forecast.wind.direction))
print("Velocidad: {0}".format(forecast.wind.speed))
```

Resultado:

```
Enfriamiento: 24
Dirección: 190
Velocidad: 8.05
```

3.1.5 Unidades de Medición

Hemos visto en los ejemplos anteriores que la mayoría de los números se encuentran crudos y no sabemos con certeza en qué medidas están expresados. Para poder conocer esta información se debe realizar lo siguiente:

```
print("Unidad de velocidad: {0}".format(forecast.units.speed))
print("Unidad de presión: {0}".format(forecast.units.pressure))
print("Unidad de distancia: {0}".format(forecast.units.distance))
print("Unidad de temperatura: {0}".format(forecast.units.temperature))
```

Resultado:

```
Unidad de velocidad: km/h
Unidad de presión: mb
Unidad de distancia: km
Unidad de temperatura: C
```

3.1.6 Condición actual del clima

`pyql-weather` nos permite obtener las condiciones del clima de una manera sencilla:

```
print(forecast.item.condition.code)
print(forecast.item.condition.date)
print(forecast.item.condition.temp)
print(forecast.item.condition.text)
```

Resultado:

```
28
Tue, 28 Apr 2015 12:53 pm CDT
26
Mostly Cloudy
```

- Esta información que hemos obtenido no nos permite entender con claridad las condiciones climatológicas actuales.

En el siguiente ejemplo uniremos varias partes de la librería para presentar la información de una manera más formal:

```
ciudad = forecast.location.city
region = forecast.location.region
pais = forecast.location.country
print("Condiciones del clima en {0}, {1}, {2}.\n".format(ciudad, region, pais))
print("Fecha: {0}".format(forecast.item.condition.date))
print("Temperatura: {0}° {1}".format(forecast.item.condition.temp, forecast.units.temperature))
print("Condición: {0} ({1})".format(forecast.item.condition.text, forecast.item.condition.code))
```

Resultado:

```
Condiciones del clima en Guadalajara, JA, Mexico.

Fecha: Tue, 28 Apr 2015 12:53 pm CDT
Temperatura: 26° C
Condición: Mostly Cloudy (28)
```

3.1.7 Pronóstico 5 días.

Yahoo Weather nos permite conocer el pronóstico del clima de 5 fechas continuas (incluyendo la fecha de consulta).

- Por ejemplo: Si consultamos el clima el día de hoy **28-04-2015 (día 1)**, Yahoo nos ofrecerá el pronóstico para los días **29-04-2015 (día 2)**, **30-04-2015 (día 3)**, **01-05-2015 (día 4)** y **02-05-2015 (día 5)**.

A continuación veremos un ejemplo básico para extraer el pronóstico para la ciudad de Mérida, Yuc, México:

```
woeid = 133327
forecast = Forecast.get(woeid=woeid, u="c")

for day in forecast.item.forecast:
    print(day)
```

Resultado:

```
{u'code': u'30', u'text': u'Partly Cloudy', u'high': u'41', u'low': u'26', u'date': u'28 Apr 2015', u
{u'code': u'38', u'text': u'AM Thunderstorms', u'high': u'31', u'low': u'23', u'date': u'29 Apr 2015
{u'code': u'39', u'text': u'AM Showers', u'high': u'28', u'low': u'21', u'date': u'30 Apr 2015', u'da
{u'code': u'30', u'text': u'Partly Cloudy', u'high': u'32', u'low': u'19', u'date': u'1 May 2015', u
{u'code': u'34', u'text': u'Mostly Sunny', u'high': u'33', u'low': u'19', u'date': u'2 May 2015', u'c
```

Podemos observar que el resultado que `pyql-weather` retorna es un arreglo de objetos en formato json los cuales podemos manipular para formatear la información de la siguiente manera:

```
woeid = 133327
forecast = Forecast.get(woeid=woeid, u="c")
ciudad = forecast.location.city
region = forecast.location.region
pais = forecast.location.country
print("Condiciones del clima para la ciudad de {0}, {1}, {2}: \n".format(ciudad, region, pais))
for day in forecast.item.forecast:
    print("Fecha: {0} {1}".format(day['day'], day['date']))
    print("Pronóstico: {0} ({1})".format(day['text'], day['code']))
    print("Temperatura Mínima: {0}° {1}".format(day['low'], forecast.units.temperature))
    print("Temperatura Máxima: {0}° {1}".format(day['high'], forecast.units.temperature))
    print("**************************************************************************")
```

Resultado:

```
Condiciones del clima para la ciudad de Merida, YU, Mexico:

Fecha: Tue 28 Apr 2015
Pronóstico: Partly Cloudy (30)
Temperatura Mínima: 26° C
Temperatura Máxima: 41° C
**************************************************************************
Fecha: Wed 29 Apr 2015
Pronóstico: AM Thunderstorms (38)
Temperatura Mínima: 23° C
Temperatura Máxima: 31° C
**************************************************************************
Fecha: Thu 30 Apr 2015
Pronóstico: AM Showers (39)
Temperatura Mínima: 21° C
Temperatura Máxima: 28° C
**************************************************************************
Fecha: Fri 1 May 2015
Pronóstico: Partly Cloudy (30)
Temperatura Mínima: 19° C
Temperatura Máxima: 32° C
**************************************************************************
Fecha: Sat 2 May 2015
Pronóstico: Mostly Sunny (34)
Temperatura Mínima: 19° C
Temperatura Máxima: 33° C
**************************************************************************
```

3.1.8 Elementos as_json()

La librería pyql-weather nos permite acceder a los datos de cada propiedad en formato JSON. La función a utilizar es llamada `as_json()`. A continuación presentamos algunos ejemplo:

1. Viento en JSON:

```
print(forecast.wind.as_json())
```

- **Resultado**:

```
{u'direction': u'260', u'speed': u'14.48', u'chill': u'40'}
```

2. Astronomía en JSON:

```
print(forecast.astronomy.as_json())
```

- **Resultado**:

```
{u'sunset': u'7:21 pm', u'sunrise': u'6:29 am'}
```

3. Localización en JSON:

```
print(forecast.location.as_json())
```

- **Resultado**:

```
{u'city': u'Merida', u'region': u'YU', u'country': u'Mexico'}
```

4. Objeto Forecast completo en JSON:

```
print(forecast.as_json())
```

- Resultado:

```
{u'lastBuildDate': u'Tue, 28 Apr 2015 2:46 pm CDT', u'atmosphere': {u'pressure': u'982.05', u'ri
```