API - The World Bank

Exempla (Python)

Marian Petruk

April 7 2017

Contents

1 Basics

```
import wbpy
from pprint import pprint

api = wbpy.IndicatorAPI()

iso_country_codes = ["GB", "FR", "JP"]
total_population = "SP.POP.TOTL"

dataset = api.get_dataset(total_population, iso_country_codes, date="2010:2012")
```

```
>>> dataset
<wbpy.indicators.IndicatorDataset(u'SP.POP.TOTL', u'Population, total') with id:
↪  203402188>

>>> dataset.as_dict()
{u'FR': {u'2010': 65031235.0, u'2011': 65371613.0, u'2012': 65696689.0},
 u'GB': {u'2010': 62271177.0, u'2011': 62752472.0, u'2012': 63227526.0},
 u'JP': {u'2010': 127450459.0, u'2011': 127817277.0, u'2012': 127561489.0}}

>>> dataset.api_url
'http://api.worldbank.org/countries/GBR;FRA;JPN/indicators/SP.POP.TOTL?date=2010\%3A2012&\-format=jso

>>> dataset.indicator_name
u'Population, total'

>>> dataset.indicator_topics
[{u'id': u'8', u'value': u'Health '},
 {u'id': u'19', u'value': u'Climate Change'}]

>>> dataset.countries
{u'FR': u'France', u'GB': u'United Kingdom', u'JP': u'Japan'}
```

2 Searching for indicators

```
>>> population_indicators = api.get_indicators(search="population")
>>> print("There are now only " + str(len(population_indicators)) + " indicators to
 ↪  browse!")
"There are now only 1312 indicators to browse!"

>>> health_topic_id = 8
>>> health_indicators = api.get_indicators(search="population", topic=health_topic_id)
>>> print("We've narrowed it down to " + str(len(health_indicators)) + " indicators!")
"We've narrowed it down to 35 indicators!"

>>> print(health_indicators[list(health_indicators)[1]])
{'name': 'Population, total', 'source': {'id': '2', 'value': 'World Development
 ↪  Indicators'}, 'sourceNote': 'Total population is based on the de facto definition of
 ↪  population, which counts all residents regardless of legal status or citizenship. The
 ↪  values shown are midyear estimates.', 'sourceOrganization': '(1) United Nations
 ↪  Population Division. World Population Prospects, (2) Census reports and other
 ↪  statistical publications from national statistical offices, (3) Eurostat: Demographic
 ↪  Statistics, (4) United Nations Statistical Division. Population and Vital Statistics
 ↪  Report (various years), (5) U.S. Census Bureau: International Database, and (6)
 ↪  Secretariat of the Pacific Community: Statistics and Demography Programme.',
 ↪  'topics': [{'id': '19', 'value': 'Climate Change'}, {'id': '8', 'value': 'Health '}]}


>>> countries = api.get_countries(search="united")
>>> api.print_codes(countries)
{'AE': {'id': 'ARE', 'name': 'United Arab Emirates', 'region': {'id': 'MEA', 'value':
 ↪  'Middle East & North Africa'}, 'adminregion': {'id': '', 'value': ''}, 'incomeLevel':
 ↪  {'id': 'HIC', 'value': 'High income'}, 'lendingType': {'id': 'LNX', 'value': 'Not
 ↪  classified'}, 'capitalCity': 'Abu Dhabi', 'longitude': '54.3705', 'latitude':
 ↪  '24.4764'}, 'GB': {'id': 'GBR', 'name': 'United Kingdom', 'region': {'id': 'ECS',
 ↪  'value': 'Europe & Central Asia'}, 'adminregion': {'id': '', 'value': ''},
 ↪  'incomeLevel': {'id': 'HIC', 'value': 'High income'}, 'lendingType': {'id': 'LNX',
 ↪  'value': 'Not classified'}, 'capitalCity': 'London', 'longitude': '-0.126236',
 ↪  'latitude': '51.5002'}, 'US': {'id': 'USA', 'name': 'United States', 'region': {'id':
 ↪  'NAC', 'value': 'North America'}, 'adminregion': {'id': '', 'value': ''},
 ↪  'incomeLevel': {'id': 'HIC', 'value': 'High income'}, 'lendingType': {'id': 'LNX',
 ↪  'value': 'Not classified'}, 'capitalCity': 'Washington D.C.', 'longitude': '-77.032',
 ↪  'latitude': '38.8895'}}
```

3 More searching

```
>>> import wbpy
>>> api = wbpy.IndicatorAPI()
>>> all_regions = api.get_regions()
>>> all_sources = api.get_sources()
>>> print("There are " + str(len(all_regions)) + " regions and " + str(len(all_sources))
↪  + " sources.")
"There are 48 regions and 42 sources.
"

>>> pprint(api.search_results("debt", all_sources))
{'20': {'description': '', 'name': 'Quarterly Public Sector Debt', 'url': ''},
 '22': {'description': '',
        'name': 'Quarterly External Debt Statistics/SDDS (New)',
        'url': ''},
 '23': {'description': '',
        'name': 'Quarterly External Debt Statistics/GDDS (New)',
        'url': ''},
 '54': {'description': '', 'name': '(JEDH) Joint External Debt Hub', 'url': ''},
 '6': {'description': '', 'name': 'International Debt Statistics', 'url': ''}}

>>> narrow_matches = api.get_topics(search="poverty")
>>> wide_matches = api.get_topics(search="poverty", search_full=True)
>>> print(str(len(narrow_matches)) + " topic(s) match(es) 'poverty' in the title field,
↪  and " + str(len(wide_matches)) + " topics match 'poverty' in all fields.")
1 topic(s) match(es) 'poverty' in the title field, and 8 topics match 'poverty' in all
↪  fields.
```

4 Country codes

```
>>> print(api.print_codes(api.NON_STANDARD_REGIONS))
1A                              Arab World
1W                              World
4E                              East Asia & Pacific (developing only)
7E                              Europe & Central Asia (developing only)
8S                              South Asia
A4                              Sub-Saharan Africa excluding South Africa
A5                              Sub-Saharan Africa excluding South Africa and Nigeria
A9                              Africa
C4                              East Asia and the Pacific (IFC classification)
C5                              Europe and Central Asia (IFC classification)
C6                              Latin America and the Caribbean (IFC classification)
C7                              Middle East and North Africa (IFC classification)
C8                              South Asia (IFC classification)
C9                              Sub-Saharan Africa (IFC classification)
EU                              European Union
JG                              Channel Islands
KV                              Kosovo
M2                              North Africa
OE                              OECD members
S1                              Small states
S2                              Pacific island small states
S3                              Caribbean small states
S4                              Other small states
XC                              Euro area
XD                              High income
XE                              Heavily indebted poor countries (HIPC)
XJ                              Latin America & Caribbean (developing only)
XL                              Least developed countries: UN classification
XM                              Low income
XN                              Lower middle income
XO                              Low & middle income
XP                              Middle income
XQ                              Middle East & North Africa (developing only)
XR                              High income: nonOECD
XU                              North America
XY                              Not classified
Z4                              East Asia & Pacific (all income levels)
Z7                              Europe & Central Asia (all income levels)
ZF                              Sub-Saharan Africa (developing only)
ZG                              Sub-Saharan Africa (all income levels)
ZJ                              Latin America & Caribbean (all income levels)
ZQ                              Middle East & North Africa (all income levels)
```

5 Climate API

5.1 Instrumental data

```
>>> c_api = wbpy.ClimateAPI()
>>> print(c_api.ARG_DEFINITIONS["instrumental_types"])
{'pr': 'Precipitation (rainfall and assumed water equivalent), in millimeters', 'tas':
 ↪  'Temperature, in degrees Celsius'}

>>> print(c_api.ARG_DEFINITIONS["instrumental_intervals"])
['year', 'month', 'decade']
```

5.2 Modelled data

```
>>> import wbpy
>>> api = wbpy.IndicatorAPI()
>>> c_api = wbpy.ClimateAPI()
>>> print(c_api.ARG_DEFINITIONS["modelled_types"])
{'tmin_means': 'Average daily minimum temperature, Celsius', 'tmax_means': 'Average daily
 ↪  maximum temperature, Celsius', 'tmax_days90th': "Number of days with max temperature
 ↪  above the control period's 90th percentile (hot days)", 'tmin_days90th': "Number of
 ↪  days with min temperature above the control period's 90th percentile (warm nights)",
 ↪  'tmax_days10th': "Number of days with max temperature below the control period's 10th
 ↪  percentile (cool days)", 'tmin_days10th': "Number of days with min temperature below
 ↪  the control period's 10th percentile (cold nights)", 'tmin_days0': 'Number of days
 ↪  with min temperature below 0 degrees Celsius', 'ppt_days': 'Number of days with
 ↪  precipitation > 0.2mm', 'ppt_days2': 'Number of days with precipitation > 2mm',
 ↪  'ppt_days10': 'Number of days with precipitation > 10mm', 'ppt_days90th': "Number of
 ↪  days with precipitation > the control period's 90th percentile", 'ppt_dryspell':
 ↪  'Average number of days between precipitation events', 'ppt_means': 'Average daily
 ↪  precipitation', 'pr': 'Precipitation (rainfall and assumed water equivalent), in
 ↪  millimeters', 'tas': 'Temperature, in degrees Celsius'}

>>> c_api.ARG_DEFINITIONS["modelled_intervals"]
{'aanom': 'Average annual change (anomaly).',
 'aavg': 'Annual average',
 'annualanom': 'Average annual change (anomaly).',
 'annualavg': 'Annual average',
 'manom': 'Average monthly change (anomaly).',
 'mavg': 'Monthly average'}
```

6 Cache

The default cache function uses system temporary files. The function has to take a url, and return the corresponding web page as a string.

```
import wbpy


def func(url):
    # Basic function that doesn't do any caching
    import urllib2
    return urllib2.urlopen(url).read()

# Either pass it in on instantiation...
ind_api = wbpy.IndicatorAPI(fetch=func)

# ...or point api.fetch to it.
climate_api = wbpy.ClimateAPI()
climate_api.fetch = func
```