pyopendata Documentation

Release 0.0.3.dev

sinhrks

October 26, 2014

Contents

Contents:

CHAPTER 1

What's New

1.1 0.0.2

- Support WorldBank API by *DataStore('worldbank')*.
- *DataResource* now has *get* method to search a resource in the package.
- *DataResource* now caches data once read from data source.
- *EuroStatStore* and *EuroStatResource* renamed to *EurostatStore* and *EurostatResource*.

1.2 0.0.1

Initial Release

CHAPTER 2

Overview

`pyopendata` is a Python utility to offer an unified API to read world various data sources, and output `pandas.DataFrame`.

This is an unstable release and API is forced to be changed.

CHAPTER 3

Installation

Use `pip`.

```
pip install pyopendata
```

CHAPTER 4

Basic Usage

This section explains how to retrieve data from website which uses CKAN API.You can create `DataStore` instance to access CKAN website by passing CKAN URL to `DataStore` class.

In this example, we're going to retrieve the 'California Unemployment Statistics' data from data.gov. The target URL is:

- https://catalog.data.gov/dataset/california-unemployment-statistics/resource/ffd05307-4528-4d15-a370-c16222119227

We can read abov URL as:

- CKAN API URL: https://catalog.data.gov/
- Package ID: california-unemployment-statistics
- Resource ID: ffd05307-4528-4d15-a370-c16222119227

```
In [1]: import pyopendata as pyod

In [2]: store = pyod.DataStore('http://catalog.data.gov/')

In [3]: store
Out[3]: CKANStore (http://catalog.data.gov)
```

`DataStore.serch` performs search by keyword. Results will be the list of packages. You can select a target package by slicing.

```
In [4]: packages = store.search('Unemployment Statistics')

In [5]: packages
Out[5]:
[annual-survey-of-school-system-finances (1 resource),
 current-population-survey (1 resource),
 federal-aid-to-states (1 resource),
 dataferrett (1 resource),
 unemployment-rate (3 resources),
 local-area-unemployment-statistics (2 resources),
 mass-layoff-statistics (1 resource),
 consolidated-federal-funds-report (1 resource),
 annual-survey-of-state-government-finances (1 resource),
 foreign-labor-statistics (1 resource)]

In [6]: packages[0]
Out[6]: annual-survey-of-school-system-finances (1 resource)
```

Otherwise, specify the package name to be retrieved.

```
In [7]: package = store.get('california-unemployment-statistics')

In [8]: package
Out[8]: california-unemployment-statistics (4 resources)
```

A package has resources (files) which contains actual data. You use *get* method to retrieve the resource.

```
In [9]: resource = package.get('ffd05307-4528-4d15-a370-c16222119227')

In [10]: resource
Out[10]:
Resource ID: ffd05307-4528-4d15-a370-c16222119227
Resource Name: Comma Separated Values File
Resource URL: https://data.lacity.org/api/views/5zrb-xqhf/rows.csv?accessType=DOWNLOAD
Format: CSV, Size: None
```

Once you get the resource, use `read` method to read data as pandas `DataFrame`.

Important: The target file must be the correct format which can be parsed by `pandas` IO functions.

```
In [11]: df = resource.read()

[====================]

In [12]: df.head()
Out[12]:
   Year Period                Area   Unemployment Rate  Labor Force  \
0  2013    Jan          California               10.4%     18556500
1  2013    Jan  Los Angeles County               10.9%      4891500
2  2013    Jan    Los Angeles City                 12%      1915600
3  2013    Feb          California  9.699999999999999%     18648300
4  2013    Feb  Los Angeles County               10.3%      4924000

   Employment  Unemployment Adjusted Preliminary
0    16631900       1924600  Not Adj  Not Prelim
1     4357800        533800  Not Adj  Not Prelim
2     1684800        230800  Not Adj  Not Prelim
3    16835900       1812400  Not Adj  Not Prelim
4     4418000        506000  Not Adj  Not Prelim
```

Or you can get raw data by specifying `raw=True`.

```
In [13]: raw = resource.read(raw=True)

In [14]: raw[:100]
Out[14]: 'Year,Period,Area,Unemployment Rate,Labor Force,Employment,Unemployment,Adjusted,Preliminary
```

CHAPTER 5

Access to CKAN Datasource

`DataStore` supports an access to CKAN API. Websites which has CKAN API are listed in the following URL.

http://ckan.org/instances/

5.1 Initialize CKAN Datasource

To create `DataStore` for CKAN access, pass URL offers API. For example, data.gov offers API via:

http://catalog.data.gov/api

In this case, any of following 4 URL formats can be used for `DataStore` initialization.

- http://catalog.data.gov/api
- http://catalog.data.gov/api/
- http://catalog.data.gov
- http://catalog.data.gov/

```
In [1]: import pyopendata as pyod

In [2]: pyod.DataStore('http://catalog.data.gov/api')
Out[2]: CKANStore (http://catalog.data.gov)

In [3]: pyod.DataStore('http://catalog.data.gov/api/')
Out[3]: CKANStore (http://catalog.data.gov)

In [4]: pyod.DataStore('http://catalog.data.gov')
Out[4]: CKANStore (http://catalog.data.gov)

In [5]: pyod.DataStore('http://catalog.data.gov/')
Out[5]: CKANStore (http://catalog.data.gov)
```

Important: *DateStore* validates the input URL using CKAN *site_read* function. If this function is closed on the target website, initialization should fail. In this case, instantiate `pyod.CKANStore` directly.

For example, data.go.jp doesn't allow to use *site_read*.

```
# should fail
In [6]: pyod.DataStore('http://www.data.go.jp/data')
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
```

```
<ipython-input-6-523a640bdcc0> in <module>()
----> 1 pyod.DataStore('http://www.data.go.jp/data')

/home/docs/checkouts/readthedocs.org/user_builds/pyopendata/envs/latest/local/lib/python2.7/site-pacl
    172             if store.is_valid():
    173                 return store
--> 174         raise ValueError('Unable to initialize DataStore with {0}'.format(kind_or_url))
    175
    176     def __init__(self, kid_or_url=None, proxies=None):

ValueError: Unable to initialize DataStore with http://www.data.go.jp/data

# should be OK
In [7]: pyod.CKANStore('http://www.data.go.jp/data')
Out[7]: CKANStore (http://www.data.go.jp/data)
```

5.2 Read Data

Once you create `DataStore`, you can access to data as described in *Basic Usage*.

CHAPTER 6

Access to Eurostat Datasource

Create `DataStore` for Eurostat by passing `eurostat` at initialization.

```
In [1]: import pyopendata as pyod

In [2]: store = pyod.DataStore('eurostat')

In [3]: store
Out[3]: EurostatStore (http://www.ec.europa.eu/eurostat/SDMX/diss-web/rest)
```

Get Employed doctorate holders in non managerial and non professional occupations by fields of science data. The result will be a `DataFrame` which has `DatetimeIndex` as index and `MultiIndex` of attributes or countries as column. The target URL is:

- http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=cdh_e_fos&lang=en

We can read above URL as:

- Resource ID: cdh_e_fos

```
In [4]: resource = store.get('cdh_e_fos')

In [5]: resource
Out[5]: EurostatResource (http://www.ec.europa.eu/eurostat/SDMX/diss-web/rest/data/cdh_e_fos/?)

In [6]: df = resource.read();

In [7]: df
Out[7]:
UNIT               Percentage                                \
Y_GRAD                  Total
FOS07        Natural sciences
GEO                   Austria  Belgium  Bulgaria  Cyprus
FREQ                   Annual   Annual    Annual  Annual
TIME_PERIOD
2006-01-01                NaN      NaN         0     100
2009-01-01                NaN      NaN       NaN     NaN

UNIT                                                          \
Y_GRAD
FOS07
GEO          Germany (until 1990 former territory of the FRG)
FREQ                                                   Annual
TIME_PERIOD
2006-01-01                                              36.65
```

```
2009-01-01                                                  NaN

UNIT                                ...                                  \
Y_GRAD                              ...                  1990 and after
FOS07                               ...                   Not specified
GEO                                 ...                          Russia
FREQ                                ...                          Annual
TIME_PERIOD                         ...
2006-01-01                          ...                             NaN
2009-01-01                          ...                             NaN

UNIT
Y_GRAD
FOS07
GEO          Sweden  Slovenia  Turkey  United States
FREQ         Annual    Annual  Annual         Annual
TIME_PERIOD
2006-01-01      NaN       NaN     NaN            NaN
2009-01-01      NaN       NaN     NaN            NaN

[2 rows x 336 columns]
```

You can access to specific data by slicing column.

```
In [8]: usa = df['Percentage']['Total']['Natural sciences']['United States']

In [9]: usa
Out[9]:
FREQ         Annual
TIME_PERIOD
2006-01-01    46.31
2009-01-01    39.65
```

CHAPTER 7

Access to OECD Datasource

Create `DataStore` for OECD by passing `oecd` at initialization.

```
In [1]: import pyopendata as pyod

In [2]: store = pyod.DataStore('oecd')

In [3]: store
Out[3]: OECDStore (http://stats.oecd.org/SDMX-JSON/data)
```

Get Trade Union Density data. The result will be a `DataFrame` which has `DatetimeIndex` as index and conutries as column. The target URL is:

- http://stats.oecd.org/Index.aspx?DataSetCode=UN_DEN

We can read above URL as:

- Resource ID: UN_DEN

```
In [4]: resource = store.get('UN_DEN')

In [5]: resource
Out[5]: OECDResource (http://stats.oecd.org/SDMX-JSON/data/UN_DEN/SVN+FRA+EST+SVK+BEL+ESP+LUX+ISR+SWE

In [6]: df = resource.read();

In [7]: df
Out[7]:
Country     Australia    Austria    Belgium     Canada  Switzerland  \
Time
1960-01-01  50.172928  67.883195  39.289583  29.162502    31.016277
1961-01-01  49.471810  67.332974  38.320964  28.517935    30.221666
1962-01-01  49.521062  66.657029  36.851721  27.068670    29.564944
1963-01-01  49.163413  67.059648  37.687135  26.763945    28.907253
1964-01-01  48.192964  66.624361  37.280876  26.438817    28.487138
...               ...        ...        ...        ...          ...
2009-01-01  19.312195  28.650257  51.509849  27.260124    17.337015
2010-01-01  18.444057  28.371679  50.594410  27.408303    17.120516
2011-01-01  18.514077  27.769651  50.374782  27.111712    16.653467
2012-01-01  18.197189        NaN        NaN  27.462644    16.211944
2013-01-01  17.040712        NaN        NaN  27.182000          NaN

Country         ...       Slovak Republic   Slovenia     Sweden    Turkey  \
Time            ...
1960-01-01      ...                   NaN        NaN  72.081019       NaN
```

```
1961-01-01     ...                   NaN        NaN  72.396761       NaN
1962-01-01     ...                   NaN        NaN  72.932867       NaN
1963-01-01     ...                   NaN        NaN  66.066797       NaN
1964-01-01     ...                   NaN        NaN  66.725265       NaN
...            ...                   ...        ...        ...       ...
2009-01-01     ...             17.040898        NaN  68.414634  5.860599
2010-01-01     ...             16.948282  26.272989  68.219845  5.853252
2011-01-01     ...             16.955248  24.399640  67.496803  5.394394
2012-01-01     ...                   NaN        NaN  67.505935  4.541725
2013-01-01     ...                   NaN        NaN  67.732503       NaN

Country     United States
Time
1960-01-01      30.897484
1961-01-01      29.518912
1962-01-01      29.342769
1963-01-01      28.513375
1964-01-01      28.306461
...                   ...
2009-01-01      11.794019
2010-01-01      11.383460
2011-01-01      11.329488
2012-01-01      11.078848
2013-01-01      10.807891

[54 rows x 35 columns]
```

You can access to specific data by slicing column.

```
In [8]: usa = df['United States']

In [9]: usa
Out[9]:
Time
1960-01-01    30.897484
1961-01-01    29.518912
1962-01-01    29.342769
...
2011-01-01    11.329488
2012-01-01    11.078848
2013-01-01    10.807891
Name: United States, Length: 54
```

CHAPTER 8

Access to WorldBank Datasource

Create `DataStore` for WorldBank by passing `worldbank` at initialization.

```
In [1]: import pyopendata as pyod

In [2]: store = pyod.DataStore('worldbank')

In [3]: store
Out[3]: WorldBankStore (http://api.worldbank.org)
```

Get GDP per capita (current US$) data. The result will be a `DataFrame` which has `DatetimeIndex` as index, and indicator and conutries as column. The target URL is:

- http://data.worldbank.org/indicator/NY.GDP.PCAP.CD

We can read above URL as:

- Resource ID: NY.GDP.PCAP.CD

```
In [4]: resource = store.get('NY.GDP.PCAP.CD')

In [5]: resource
Out[5]: WorldBankResource (http://api.worldbank.org/countries/all/indicators/NY.GDP.PCAP.CD?format=js

In [6]: df = resource.read();

In [7]: df.columns
Out[7]:
MultiIndex(levels=[[u'GDP per capita (current US$)'], [u'Afghanistan', u'Albania', u'Algeria', u'Amer
           labels=[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
           names=[u'indicator', u'country'])
```

You can access to specific data by slicing column.

```
In [8]: jp = df['GDP per capita (current US$)']['Japan']

In [9]: jp
Out[9]:
date
1960-01-01    478.995340
1961-01-01    563.586760
1962-01-01    633.640315
...
2011-01-01    46203.698037
2012-01-01    46548.269637
```

```
2013-01-01    38492.088895
Name: Japan, Length: 54
```

CHAPTER 9

IO Methods

`io` module has some methods to parse specific types of XMML/JSON.

- *read_jstat*
- *read_sdmx*
- *read_jsdmx*

9.1 read_jstat

Read JSON-stat file.

```
In [1]: import pyopendata as pyod

In [2]: pyod.io.read_jstat('http://json-stat.org/samples/us-gsp.json')
Out[2]:
concept             Gross State Product  \
year state
2013 Alabama                     174400
     Alaska                       45600
     Arizona                     261300
     Arkansas                    105800
     California                 2080600
...                                 ...
     Virginia                    427700
     Washington                  351100
     West Virginia                66600
     Wisconsin                   251400
     Wyoming                      38200

concept             Gross State Product as percentage of national GDP  \
year state
2013 Alabama                                                     1.20
     Alaska                                                      0.31
     Arizona                                                     1.80
     Arkansas                                                    0.73
     California                                                 13.34
...                                                               ...
     Virginia                                                    2.95
     Washington                                                  2.42
     West Virginia                                               0.46
     Wisconsin                                                   1.73
```

```
     Wyoming                                                    0.26

concept             Gross State Product per capita  Population
year state
2013 Alabama                                 36333         4.8
     Alaska                                  65143         0.7
     Arizona                                 40828         6.4
     Arkansas                                36483         2.9
     California                              51914        37.3
...                                            ...         ...
     Virginia                                53463         8.0
     Washington                              52403         6.7
     West Virginia                           35053         1.9
     Wisconsin                               44105         5.7
     Wyoming                                 63667         0.6

[51 rows x 4 columns]
```

9.2 read_sdmx

Read SDMX-XML file. The format is used in Eurostat.

9.3 read_jsdmx

Read SDMX-JSON file. The format is used in OECD.

API:

CHAPTER 10

pyopendata package

10.1 Subpackages

10.1.1 pyopendata.io package

Subpackages

pyopendata.io.tests package

Submodules

pyopendata.io.tests.test_jsdmx module

class pyopendata.io.tests.test_jsdmx.**TestSDMX**(*methodName='runTest'*)

Bases: pandas.util.testing.TestCase

setUp()

test_land_use()

test_tourism()

pyopendata.io.tests.test_jstat module

class pyopendata.io.tests.test_jstat.**TestJStat**(*methodName='runTest'*)

Bases: pandas.util.testing.TestCase

setUp()

test_hierarchy()

test_oecd_canada()

test_omit_values()

test_order()

test_us_gsp()

test_us_labor()

test_us_unr()

pyopendata.io.tests.test_sdmx module

class pyopendata.io.tests.test_sdmx.**TestSDMX**(*methodName='runTest'*)

Bases: pandas.util.testing.TestCase

setUp()

test_tourism()

Module contents

Submodules

pyopendata.io.jsdmx module

pyopendata.io.jsdmx.**read_jsdmx**(*path_or_buf*)

Convert a SDMX-JSON string to pandas object

filepath_or_buffer [a valid SDMX-JSON string or file-like] http://sdmx.org/wp-content/uploads/2014/07/sdmx-json-data-message.pdf

results : Series, DataFrame, or dictionaly of Series or DataFrame.

pyopendata.io.jstat module

pyopendata.io.jstat.**read_jstat**(*path_or_buf*, *typ=u'frame'*, *squeeze=True*)

Convert a JSON-Stat string to pandas object

filepath_or_buffer [a valid JSON-Stat string or file-like] http://json-stat.org/

typ [{'frame', 'series'}] Type of object to recover (series or frame), default 'frame'

squeeze [bool, default True] If True, return DataFrame or Series when the input has only one dataset. When the input has multiple dataset, returns dictionary of results. If False, always return a dictionary.

results : Series, DataFrame, or dictionaly of Series or DataFrame.

pyopendata.io.sdmx module

class pyopendata.io.sdmx.**SDMXCode**

Bases: tuple

SDMXCode(codes, ts)

codes

Alias for field number 0

ts

Alias for field number 1

pyopendata.io.sdmx.**read_sdmx**(*path_or_buf*, *dtype=u'float64'*, *dsd=None*)

Convert a SDMX-XML string to pandas object

filepath_or_buffer [a valid SDMX-XML string or file-like] https://webgate.ec.europa.eu/fpfis/mwikis/sdmx/index.php/Main_Pag

dtype [str] dtype to coerce values

dsd [dict] parsed DSD dict corresponding to the SDMX-XML data

results : Series, DataFrame, or dictionaly of Series or DataFrame.

pyopendata.io.util module

Module contents

10.1.2 pyopendata.tests package

Submodules

pyopendata.tests.test_base module

class pyopendata.tests.test_base.**TestDataStore**(*methodName='runTest'*)
Bases: pandas.util.testing.TestCase

test_initialize()

pyopendata.tests.test_ckan module

class pyopendata.tests.test_ckan.**CKANTestBase**(*methodName='runTest'*)
Bases: pandas.util.testing.TestCase

setUp()

test_formats()

test_groups()

test_isvalid()

test_packages()

test_resource()

test_search()

class pyopendata.tests.test_ckan.**TestCKANTestSite**(*methodName='runTest'*)
Bases: pyopendata.tests.test_ckan.CKANTestBase

test_formats()

test_tags()

test_urlformat()

class pyopendata.tests.test_ckan.**TestDATAGOJP**(*methodName='runTest'*)
Bases: pyopendata.tests.test_ckan.CKANTestBase

test_formats()

test_isvalid()

test_mining_manufacture()

test_tags()

class pyopendata.tests.test_ckan.**TestDATAGOV**(*methodName='runTest'*)
Bases: pyopendata.tests.test_ckan.CKANTestBase

test_formats()

test_groups()

test_raw_data()

test_tags()

pyopendata.tests.test_eurostat module

class pyopendata.tests.test_eurostat.**TestEurostatTestSite**(*methodName='runTest'*)
Bases: pandas.util.testing.TestCase

setUp()

test_datasets()

test_get_cdh_e_fos()

test_get_sts_cobp_a()

test_isvalid()

pyopendata.tests.test_oecd module

class pyopendata.tests.test_oecd.**TestOECDTestSite**(*methodName='runTest'*)
Bases: pandas.util.testing.TestCase

setUp()

test_get_tourism()

test_get_un_den()

test_isvalid()

pyopendata.tests.test_worldbank module

class pyopendata.tests.test_worldbank.**TestWorldBankTestSite**(*methodName='runTest'*)
Bases: pandas.util.testing.TestCase

setUp()

test_get_co2_emit()

test_get_gdp_per_capita()

test_isvalid()

Module contents

10.1.3 pyopendata.util package

Submodules

pyopendata.util.network module

class pyopendata.util.network.**ProgressBar**(*total=None*, *length=20*)
Display Progress Bar

total [int] Total size to be processed

length: Number of progress bar characters

update(*current*)

Module contents

10.2 Submodules

10.3 pyopendata.base module

class pyopendata.base.**DataResource**(*format=None*, *id=None*, *name=None*, *url=None*, *proxies=None*, *size=None*, ***kwargs*)

Bases: pandas.core.base.StringMixin

Represents a data contained in the URL

read(*raw=False*, ***kwargs*)

Read data from resource

raw [bool, default False] If False, return pandas.DataFrame. If True, return raw data

kwargs: Keywords passed to pandas.read_xxx function

data : pandas.DataFrame or requests.raw.data

- Depending on the target format, parsing to pandas.DataFrame may fail. Use raw=True to get raw data in such cases.

class pyopendata.base.**DataStore**(*kid_or_url=None*, *proxies=None*)

Bases: pyopendata.base.DataResource

datasets

get(*name*)

Get resource by resource_id.

resource_id [str] id to specify resource

result : DataResource

is_valid()

Check whether the site has valid API.

is_valid : bool

search(*serch_string*)

Search resources by search_string.

search storing [str] keyword to search

result : list of DataResource

10.4 pyopendata.ckan module

class pyopendata.ckan.**CKANPackage**(*_store=None*, *resources=None*, *name=None*, ***kwargs*)

Bases: pyopendata.base.DataResource

get(*resource_id*)

Get resource by resource_id.

resource_id [str] id to specify resource

result : CKANResource

get_resource(*resource_id*)
Get resource by resource_id.

resource_id [str] id to specify resource

result : CKANResource

read(*raw=False*, ***kwargs*)
Read data from its resource if the number of resource is 1. Otherwise, raise ValueError.

raw [bool, default False] If False, return pandas.DataFrame. If True, return raw data

kwargs: Keywords passed to pandas.read_xxx function

data : pandas.DataFrame or requests.raw.data

resources

class pyopendata.ckan.**CKANResource**(*_store=None*, *resources=None*, ***kwargs*)
Bases: pyopendata.base.DataResource

read(*raw=False*, ***kwargs*)
Read data from resource

raw [bool, default False] If False, return pandas.DataFrame. If True, return raw data

kwargs: Keywords passed to pandas.read_xxx function

data : pandas.DataFrame or requests.raw.data

- Depending on the target format, parsing to `pandas.DataFrame` may fail. Use `raw=True` to get raw data in such cases.

class pyopendata.ckan.**CKANStore**(*kid_or_url=None*, *proxies=None*)
Bases: pyopendata.base.DataStore

Storage class to read data via CKAN API

url [string] URL for CKAN API

datasets

get(*object_id*)
Get instance related to object_id. First try to get_resource, then get_package.

object_id [str] id to specify resource or package

result : CKANResouce or CKANPackage

get_package(*package_id*)
Get package related to package_id.

package_id [str] id to specify package

result : CKANPackage

get_packages_from_group(*group_id*)

get_resource(*resource_id*)
Get resource by resource_id.

resource_id [str] id to specify resource

result : CKANResource

get_resources_from_tag(*tag*)

groups

is_valid()
Check whether the site has valid API.

is_valid : bool

packages

search(*search_string*)
Search resources by search_string.

search storing [str] keyword to search

result : list of CKANResource

search_package(*search_string*)

search_resource(*search_string*)

tags

10.5 pyopendata.eurostat module

class pyopendata.eurostat.**EurostatResource**(***kwargs*)
Bases: pyopendata.base.DataResource

dsd

class pyopendata.eurostat.**EurostatStore**(*kid_or_url=None*, *proxies=None*)
Bases: pyopendata.base.DataStore

Storage class to read Eurostat data

datasets

get(*data_id*)
Get resource by resource_id.

resource_id [str] id to specify resource

result : EurostatResource

10.6 pyopendata.oecd module

class pyopendata.oecd.**OECDResource**(*format=None*, *id=None*, *name=None*, *url=None*, *proxies=None*, *size=None*, ***kwargs*)
Bases: pyopendata.base.DataResource

class pyopendata.oecd.**OECDStore**(*kid_or_url=None*, *proxies=None*)
Bases: pyopendata.base.DataStore

Storage class to read OECD data

get(*resource_id*)
Get resource by resource_id.

resource_id [str] id to specify resource

result : OECDResource

10.7 pyopendata.undata module

class pyopendata.undata.**UNdataResource**(***kwargs*)
Bases: pyopendata.base.DataResource

class pyopendata.undata.**UNdataStore**(*kid_or_url=None*, *proxies=None*)
Bases: pyopendata.base.DataStore

Storage class to read UNdata data

datasets

get(*data_id*)

is_valid()
Check whether the site has valid API.

is_valid : bool

10.8 pyopendata.version module

10.9 pyopendata.worldbank module

class pyopendata.worldbank.**WorldBankResource**(*format=None*, *id=None*, *name=None*, *url=None*, *proxies=None*, *size=None*, ***kwargs*)
Bases: pyopendata.base.DataResource

entries_per_page = 500

class pyopendata.worldbank.**WorldBankStore**(*kid_or_url=None*, *proxies=None*)
Bases: pyopendata.base.DataStore

Storage class to read World Bank data

get(*resource_id*)
Get resource by resource_id.

resource_id [str] id to specify resource

result : WorldBankResource

10.10 Module contents

p

C

D

E

G

I

O

P

U

W