postcode-api-wrapper Documentation

Release 1.0.0

F. Brekeveld

March 26, 2016

Postcode-API wrapper

Contents:

CHAPTER 1

Installation

1.1 Introduction

The postcode-api-wrapper package offers a simple API to the Postcode.nl API REST service. To use the API-service you will need an *access_key* and *access_secret*. For details check api.postcode.nl.

1.2 Download & Install

1.2.1 From pypi

Install the package with pip:

```
$ pip install postcodepy
```

You may consider using *virtualenv* to create isolated Python environments. Python 3.4 has *pyvenv* providing the same kind of functionality.

1.2.2 From Github

```
$ git clone https://github.com/hootnot/postcode-api-wrapper.git
$ cd postcode-api-wrapper

Run the tests:
$ python setup.py test
$ python setup.py install
```

CHAPTER 2

postcodepy API

Postcode API module.

2.1 Interface to the REST-service

The REST-service endpoints are covered by the methods of `postcode.API` class.

class `postcodepy.`**`API`**(*environment='practice'*, *access_key=None*, *access_secret=None*, *headers=None*)

Bases: `postcodepy.postcodepy.EndpointsMixin`, `object`

API - postcode API class.

`__init__`(*environment='practice'*, *access_key=None*, *access_secret=None*, *headers=None*)

Instantiate API wrapper.

Parameters

- **`environment`** (*`str`*) – the environment to use. Currently only 'live'.
- **`access_key`** (*`str`*) – the access key provided by postcode.nl . If not provided an ERRauthAccessUnknownKey exception is raised
- **`access_secret`** (*`str`*) – the access secret provided by postcode.nl . If not provided an ERRauthAccessUnknownSecret exception is raised
- **`headers`** (*`dict`*) – optional headers to set
- **`returns`** – a response dictionary

`get_postcodedata`(*postcode*, *nr*, *addition=''*, ***params*)

get_postcodedata - fetch information for 'postcode'.

Parameters

- **`postcode`** (*`string`*) – The full (dutch) postcode
- **`nr`** (*`int`*) – The housenumber
- **`addition`** (*`string (optional)`*) – the extension to a housenumber
- **`params`** (*`dict (optional)`*) – a list of parameters to send with the request.
- **`returns`** – a response dictionary

`get_signalcheck`(*sar*, ***params*)

get_signalcheck - perform a signal check.

Parameters

- **sar** (*dict*) – signal-api-request specified as a dictionary of parameters. All of these parameters are optional. For details check https://api.postcode.nl/documentation/signal-api-example.
- **returns** – a response dictionary

2.2 Exceptions

class postcodepy.**PostcodeError**(*exceptionId*, *response_data=None*)
Bases: exceptions.Exception

PostcodeError - Generic error class, catches response errors.

__init__(*exceptionId*, *response_data=None*)
instantiate PostcodeError instance.

Parameters

- **exceptionId** (*str*) – the id of the exception. It should match one of the known exception id's. If it does not match it is set to: ERRUnknownExceptionFromPostcodeNl
- **response_data** (*data*) – the data received at the moment the exception occurred

2.3 Typedefs

Type definitions.

postcodepy.typedefs.**translate_addresstype**(*f*)
decorator to translate the addressType field.

translate the value of the addressType field of the API response into a translated type.

postcodepy.typedefs.**translate_purposes**(*f*)
decorator to translate the purposes field.

translate the values of the purposes field of the API response into translated values.

```
POSTCODE_API_TYPEDEFS_ADDRES_TYPES =
{
    "building": "verblijfsobject",
    "house boat site": "location used for mooring house boats",
    "PO box": "PO box",
    "mobile home site": "location used for mobile homes and trailers"
}
```

```
POSTCODE_API_TYPEDEFS_PURPOSES =
{
    "assembly": "bijeenkomstfunctie",
    "office": "kantoorfunctie",
    "detention": "celfunctie",
    "lodging": "logiesfunctie",
    "shopping": "winkelfunctie",
    "sport": "sportfunctie",
    "education": "onderwijsfunctie",
    "industry": "industriefunctie",
    "other": "overige gebruiksfunctie",
    "residency": "woonfunctie",
```

```
    "healthcare": "gezondheidszorgfunctie"
}
```

The REST responses may contain fields that can be translated into the standard descriptions as provided by postcode.nl. This translation can be accomplished by simply applying the decorators to a function that parses the return value of the API-call.

2.3.1 Logging

In case values can't be translated, a warning is logged by *logger*. The message will contain the *postcode* that was responsible for the warning.

CHAPTER 3

Examples

3.1 Get information by `postcode`:

```
import os
import sys
import json
import postcodepy

access_key = os.getenv("ACCESS_KEY")
access_secret = os.getenv("ACCESS_SECRET")
# get your access key and secret at https://api.postcode.nl

"""First and third are OK, the 2nd is not OK and raises an Exception
     the exception is written to stderr
"""
api = postcodepy.API(environment='live',
                     access_key=access_key, access_secret=access_secret)

# exit program if one of these exceptions occurs
fatals = [
    'PostcodeNl_Controller_Plugin_HttpBasicAuthentication_NotAuthorizedException',
    'PostcodeNl_Controller_Plugin_HttpBasicAuthentication_PasswordNotCorrectException'
]

# 2nd and last should fail
for pc in [('1071XX', 1),
           ('1077XX', 1),
           ('7514BP', 129),
           ('7514BP', 129, 'A'),
           ('7514BP', 129, 'B')]:
    try:
        retValue = api.get_postcodedata(*pc)
        print("\nresults for: {}".format(str(pc)))
        print(json.dumps(retValue, sort_keys=True, indent=2))

    except postcodepy.PostcodeError as e:
        if e.exceptionId in fatals:
            sys.stderr.write("Exiting on fatal exception: {} [{}]".
                             format(e.exceptionId, e.msg))
            sys.exit(2)
        else:
            sys.stderr.write("--------------------------\n")
```

```
            sys.stderr.write("{}\n".format(str(pc)))
            sys.stderr.write("{}\n".format(e.exceptionId))
            sys.stderr.write("{}\n".format(json.dumps(
                  e.response_data, sort_keys=True, indent=2))
            )
```

3.2 Output

```
results for: ('1071XX', 1)
{
  "addressType": "building",
  "bagAddressableObjectId": "0363010012073352",
  "bagNumberDesignationId": "0363200012073684",
  "city": "Amsterdam",
  "houseNumber": 1,
  "houseNumberAddition": "",
  "houseNumberAdditions": [
    ""
  ],
  "latitude": 52.35994439,
  "longitude": 4.88538896,
  "municipality": "Amsterdam",
  "postcode": "1071XX",
  "province": "Noord-Holland",
  "purposes": [
    "assembly"
  ],
  "rdX": 120816,
  "rdY": 485901,
  "street": "Museumstraat",
  "surfaceArea": 38149
}

results for: ('7514BP', 129)
{
  "addressType": "building",
  "bagAddressableObjectId": "0153010000345343",
  "bagNumberDesignationId": "0153200000345342",
  "city": "Enschede",
  "houseNumber": 129,
  "houseNumberAddition": "",
  "houseNumberAdditions": [
    "",
    "A"
  ],
  "latitude": 52.22770127,
  "longitude": 6.89701549,
  "municipality": "Enschede",
  "postcode": "7514BP",
  "province": "Overijssel",
  "purposes": [
    "assembly"
  ],
  "rdX": 258149,
  "rdY": 472143,
  "street": "Lasondersingel",
```

```
  "surfaceArea": 6700
}

results for: ('7514BP', 129, 'A')
{
  "addressType": "building",
  "bagAddressableObjectId": "0153010000329929",
  "bagNumberDesignationId": "0153200000329928",
  "city": "Enschede",
  "houseNumber": 129,
  "houseNumberAddition": "A",
  "houseNumberAdditions": [
    "",
    "A"
  ],
  "latitude": 52.22770127,
  "longitude": 6.89701549,
  "municipality": "Enschede",
  "postcode": "7514BP",
  "province": "Overijssel",
  "purposes": [
    "residency"
  ],
  "rdX": 258149,
  "rdY": 472143,
  "street": "Lasondersingel",
  "surfaceArea": 119
}
```

3.2.1 Exceptions

```
---------------------------
('1077XX', 1)
PostcodeNl_Service_PostcodeAddress_AddressNotFoundException
{
  "exception": "Combination does not exist.",
  "exceptionId": "PostcodeNl_Service_PostcodeAddress_AddressNotFoundException"
}
---------------------------
('7514BP', 129, 'B')
ERRHouseNumberAdditionInvalid
{
  "exception": "Invalid housenumber addition: 'None'",
  "exceptionId": "ERRHouseNumberAdditionInvalid",
  "validHouseNumberAdditions": [
    "",
    "A"
  ]
}
```

3.3 Using typedefs on API responses

The example below applies the decorators to *parse_result*, this function will translate the *addressType* field and the values of the *purposes* field.

Listing 3.1: Typedef example

```
...
from postcodepy import typedefs
...

@typedefs.translate_purposes
@typedefs.translate_addresstype
def parse_result(r, pc):
    return r

rv = api.get_postcodedata(*pc)
rv = parse_result(rv, pc)
```

Will result in:


```
{
  "addressType": "verblijfsobject",
  "bagAddressableObjectId": "0080010000394794",
  "bagNumberDesignationId": "0080200000394793",
  "city": "Leeuwarden",
  "houseNumber": 7,
  "houseNumberAddition": "",
  "houseNumberAdditions": [
    ""
  ],
  "latitude": 53.1926878,
  "longitude": 5.83081603,
  "municipality": "Leeuwarden",
  "postcode": "8936AS",
  "province": "Friesland",
  "purposes": [
    "celfunctie"
  ],
  "rdX": 184649,
  "rdY": 578538,
  "street": "Holstmeerweg",
  "surfaceArea": 19570
}
```


In this output are the values of *addressType* and *purposes* translation results.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

p

Symbols

A

G

P

T