pnvdb Documentation

Release 0.4.0

Jan Tore Kyrdalen

Jun 02, 2021

Contents

pnvdb stands for “python veg database” and is a python wrapper for the NVDB Rest API

Data pulled from the API is licensed under Norsk lisens for offentlige data NLOD

Contents:

CHAPTER 1

Installation

Installing is as easy as

```
>>> pip install pnvdb
```

CHAPTER 2

Getting started

Start with initalizing an instance of pnvdb:

```
>>> import pnvdb
>>> nvdb = pnvdb.Nvdb(client='Your-App-Name', contact='Your-contact-information')
```

Now we can test our connection to NVDB:

```
>>> print(nvdb.status())
{'datagrunnlag': {'sist_oppdatert': '2017-11-05 11:59:37'}, 'datakatalog': {'id': 782,
↪ 'dato': '2017-09-29', 'versjon': '2.10'}}
```

To work with a spesific nvdb object, we can initialize it like this:

```
>>> objekt = nvdb.objekt(objekt_type=67, nvdb_id=86543444)
```

This will get us access to a number of attributes assosiated with this object. Let's print one out:

```
>>> print(objekt.metadata)
{'type': {'id': 67, 'navn': 'Tunnelløp'}, 'versjon': 14, 'startdato': '2014-09-19',
↪'sist_modifisert': '2017-10-24 15:40:48'}
```

We can search using area and property filters. This will return a generator object that can be itterated over.

```
>>> criteria = {'fylke':'2','egenskap':'1820>=20'} # 1820 = "Takst liten bil"
>>> objekttype = 45 # Bomstasjon
>>> tollstations = nvdb.hent(objekttype, criteria)
>>> for tollstation in tollstations:
        if tollstation.egenskap(1078): # Check for existance
                print(tollstation.egenskap(1078)['verdi'])
Hovinmoen - Dal
Dal - Boksrud
Minnesund - Hedmark grense
```

Add data to datafangst:

```
>>> datafangst = pnvdb.Datafangst(username, password, contractId)
>>> a_point = (10.39241731, 63.43053048) # Geometry
```

initialize the feature

```
>>> skiltpunkt = datafangst.feature(96, point, "Skilt")
```

Add attribute data to the feature

```
>>> skiltpunkt.attribute(1876,4605)
>>> skiltpunkt.attribute(1877,1)
>>> skiltpunkt.attribute(1671,2435)
>>> skiltpunkt.attribute(1887,1)
```

Add a comment

```
>>> skiltpunkt.comment("Fra Pnvdb")
```

Initialize a feature collection to hold the feautures

```
>>> datafangst_collection = datafangst.feature_collection()
```

Add the feature we build

```
>>> datafangst_collection.add_feature(skiltpunkt)
```

Push the feature to datafangst

```
>>> datafangst_collection.push()
```

Query the status of the transaction with datafangst

```
>>> print(datafangst_collection.status())
```

CHAPTER 3

Code refference

3.1 Nvdb

class pnvdb.**Nvdb**(*client='pnvdb'*, *contact=''*, *autoupdate=True*)

The main class for interfacing with the API.

Parameters

- **client** (*str*) – Name of client using the API
- **contact** (*str*) – Contact information of user of the API
- **autoupdate** (*Bool*) – Indicated wether constants should be up to date with latest API-Version. Default value = True

Returns Nvdb Class

Usage

```
>>> import pnvdb
>>> nvdb = pnvdb.Nvdb(client='Your-App-Name', contact='Your-contact-information')
```

fylker()

Returns an mArea object for all fylker

Returns list of *Area*

Usage

```
>>> for region in nvdb.regioner():
>>>     print(region.metadata)
```

hent(*objekt_type*, *kriterie=None*)

Return a generator object that can be itterated over to fetch the results of the query.

Parameters

- **objekt_type** (*int*) – nvdb objekttype id.

- **payload** (*dict*) – filters for the query

Returns generator of *Objekt*

Usage

```
>>> criteria = {'fylke':'2','egenskap':'1820>=20'}
>>> bomstasjoner = nvdb.hent(45, kriterie=criteria)
>>> for bomstasjon in bomstasjoner:
>>>     print(bomstasjon)
```

kommuner()

Returns an Area object for all kommuner

Returns list of *Area*

Usage

```
>>> for region in nvdb.regioner():
>>>     print(region.metadata)
```

kontraktsomrader()

Returns an Area object for all kontraktsomrader

Returns list of *Area*

Usage

```
>>> for region in nvdb.regioner():
>>>     print(region.metadata)
```

name2id = None

status = _fetch_data(self, 'status')

if autoupdate and last_seen_version != float(status['datakatalog']['versjon']):

try: update_CONST()

except: print('Autoupdate of the CONST.py file failed.

Try initializing with adminstrative privleleges, or set autoupdate = False')

logging.info('Updated name2id and kommune values from version: {} to version {}'. format(last_seen_version, status['datakatalog']['versjon']))

objekt(*objekt_type*, *nvdb_id*)

Method for creating a spesific nvdb python Objekt

Parameters

- **objekt_type** (*int*) – nvdb objekttype id.
- **nvdb_id** (*int*) – the unique nvdb id

Returns *Objekt*

Usage

```
>>> obj = nvdb.objekt(objekt_type=67, nvdb_id=89204552)
>>> print(obj.metadata)
{'versjon': 3, 'type':P {'navn': 'Tunnelløp', 'id': 67}, 'startdato': '2014-
↪01-17',
 'sist_modifisert': '2017-10-23 15:15:50'}
```

objekt_type(*objekt_type*)
Method for creating a spesific nvdb python

Parameters **objekt_type** (*int*) – nvdb objekttype id.

Returns *ObjektType*

Usage

```
>>> obj = nvdb.objekt_type(objekt_type=67)
>>> print(obj.metadata['sosinvdbnavn'])
Tunnelløp_67
```

objekt_typer()
Returns objekt_type of every avaliable obj type in nvdb

Returns List of *ObjektType*

Usage

```
>>> obj_types = nvdb.objekt_typer()
>>> print(obj_types[0].metadata['sosinvdbnavn'])
Skjerm_3
```

posisjon(*x_coordinate=None*, *y_coordinate=None*, *lat=None*, *lon=None*)
Returns a posisjon object for a given location

Parameters

- **x** (*float*) – X-coordinate in EUREF89 UTM 33
- **y** (*float*) – Y-coordinate in EUREF89 UTM 33
- **lat** (*float*) – Lattitude in EUREF89
- **lon** (*float*) – Longitude in EUREF89

Returns *Posisjon*

Usage

```
>>> pos = nvdb.posisjon(x=269815,y=7038165)
>>> print(pos.vegreferanse)
```

regioner()
Returns an Area object for all regions

Returns list of *Area*

Usage

```
>>> for region in nvdb.regioner():
>>>     print(region.metadata)
```

riksvegruter()
Returns an Area object for all riksvegruter

Returns list of *Area*

Usage

```
>>> for region in nvdb.regioner():
>>>     print(region.metadata)
```

status()
Method for getting information about the current status of the API

Returns Dict

Keys ['datakatalog', 'datagrunnlag']

Usage

```
>>> status = nvdb.status()
>>> print(status['datakatalog']['versjon'])
2.13
```

vegavdelinger()
Returns an Area object for all vegavdelinger

Returns list of *Area*

Usage

```
>>> for region in nvdb.regioner():
>>>     print(region.metadata)
```

vegreferanse(*vegreferanse*)
Return vegreferanse object. PS : Only support point refferences

Parameters **vegreferanse** (*string*) – The road refferences to objectify

Returns *Vegreferanse*

Usage

```
>>> print(nvdb.vegreferanse('1600Ev6hp12m1000'))
```

3.2 Datafangst

class pnvdb.**Datafangst**(*username=None*, *password=None*, *contractId=None*)
Main class for interfacing with the 'Datafangst' API

Parameters

- **username** (*str*) – Datafangst username
- **password** (*str*) – Datafangst password
- **contractId** (*str*) – Datafangst contract ID

Returns *Datafangst*

Usage

```
>>> import pnvdb
>>> datafangst = pnvdb.Datafangst(username, password, contractId)
```

feature(*objekt_type*, *coordinates*, *tag*)
Method for initialzing and working with a datafangst feature

Parameters

- **objekt_type** (*int*) – NVDB object type of the feature

- **coordinates** (*list of tuples or singel tuple for points*) – Coordinates describing the feature geometry
- **tag** (*str*) – Identifying tag for the feature. Identical tags will be made uniqe with a number.

Returns *Feature*

feature_collection()
Method for initializing and working with a datafangst feature collection

Returns *FeatureCollection*

3.3 Objekt

class pnvdb.models.**Objekt**(*nvdb*, *objekt_type*, *nvdb_id*, *data=None*)
Class for individual nvdb-objects.

barn

Attribute type List of *Objekt*

dump(*file_format='json'*)
Function for dumping raw API-result for object.

Parameters **file_format** (*string*) – Type of data to dump as. json or xml

Returns str

egengeometri
Boolean value that tell if the object has egengeometri or not.

Returns Bool. If it's not found it will return None

egenskap(*egenskaps_id=None*)
Function for returning egenskap based on id

Parameters **egenskaps_id** (*int*) – Id of the property type you want returned

Returns dict unless property is not found. Then None is returned.

egenskaper

Attribute type List of Dict

Keys ['datatype_tekst', 'id', 'datatype', 'verdi', 'navn']

foreldre

Attribute type List of *Objekt*

geometri

Attribute type Well Known Text

kommuner

Attribute type list of dict

Keys [fylke, navn, nummer, region, vegavdeling]

metadata

Attribute type Dict

Keys ['versjon', 'sist_modifisert', 'startdato', 'type']

vegreferanser

Attribute type *Vegreferanse*

vegsegmenter

Attribute type list of dict

Keys []

3.4 Objekt_type

class pnvdb.models.**ObjektType**(*nvdb*, *objekt_type*, *meta=None*)
Class for individual nvdb-object types. (Data catalogue)

barn

Attribute type list of *ObjektType*

dump(*file_format='json'*)
Function for dumping raw API-result for object.

Parameters **file_format** (*string*) – Type of data to dump as. json or xml

Returns str

egenskapstype(*egenskapstype_id=None*)
Function for returning egenskap based on id

Parameters **egenskaps_id** (*int*) – Id of the property type you want returned

Returns dict unless property is not found. Then None is returned.

egenskapstyper

Attribute type list of Dicts

Keys ['liste', 'navn', 'datatype_tekst', 'veiledning', 'beskrivelse', 'sensitivitet', 'sosinvdbnavn', 'objektliste_dato', 'feltlengde', 'sorteringsnummer', 'id', 'styringsparametere', 'viktighet', 'viktighet_tekst', 'datatype']

foreldre

Attribute type list of *ObjektType*

i_objekt_lista()
Function checking of an object type is part of "Objektlista"

Returns bool

metadata

Todo: Possible bug. Returns None after reading other attributes

Attribute type Dict

Keys ['navn', 'veiledning', 'beskrivelse', 'objektliste_dato', 'sosinvdbnavn', 'sorteringsnummer', 'stedfesting', 'id', 'kategorier']

relasjonstyper

Attribute type Dict

Keys [‘barn’, ‘foreldre’]

Keys in keys [‘type’, ‘relasjonstype’, ‘id’]

styringsparametere

Attribute type Dict

Keys [‘abstrakt_type’, ‘sideposisjon_relevant’, ‘retning_relevant’, ‘ajourhold_splitt’, ‘må_ha_mor’, ‘avledet’, ‘sektype_20k’, ‘er_dataserie’, ‘høyde_relevant’, ‘dekningsgrad’, ‘overlapp, ‘filtrering’, ‘flyttbar’, ‘tidsrom_relevant’, ‘ajourhold_i’, ‘kjørefelt_relevant’]

3.5 Vegreferanse

class pnvdb.models.**Vegreferanse**(*nvdb*, *vegreferanse*)

Class for working with road refferences.

fylke

The county of the road refference :Attribute type: int

hp

The hp of the road refference :Attribute type: int

kategori

The kategori of the road refference :Attribute type: String

kommune

The kommune of the road refference :Attribute type: int

meter

The meter of the road refference :Attribute type: int

nummer

The nummer of the road refference :Attribute type: int

status

The status of the road refference :Attribute type: String

3.6 Posisjon

class pnvdb.models.**Posisjon**(*nvdb*, *payload*)

Class for connecting coordinates to road refferences

vegreferanse

Attribute type *Vegreferanse*

3.7 Area

class pnvdb.models.**Area**(*nvdb*, *area_data*)

Class for area objects.

kartutsnitt

Attribute type Well Known Text

metadata

Attribute type Dict

Keys ['nummer','navn']

objekt

Attribute type *Objekt* of the Area

senterpunkt

Attribute type Well Known Text

3.8 Feature

class pnvdb.models.**Feature**(*objekt_type*, *coordinates*, *tag*)

Class for defining objects ready to push to Datafangst

attribute(*attribute_id*, *attribute_value*)

Method for adding an attribute to the feature

Parameters

- **attribute_id** (*int*) – nvdb attribute ID
- **attribute_value** (*str or int*) – value for the attribute

comment(*comment*)

Method for adding an commnet to the feature

Parameters **comment** (*str*) – The comment to add to the feature

coordinates(*geometry*)

Method for setting the geometry of the feature

Parameters **coordinates** (*list of tuples or singel tuple for points*) – Coordinates describing the feature geometry

tag(*tag*)

Method for adding a tag to the feature :param tag: Identifying tag for the feature. Identical tags will be made uniqe with a number. :type tag: str

3.9 FeatureCollection

class pnvdb.models.**FeatureCollection**(*url*, *username*, *password*, *headers*)

Class for defining a set of objects ready to push to datafangst

add_feature(*feature*)

Method that adds a .*Feature* to the instance

push()

Method that pushes the FeatureCollection to datafangst

Returns xml respons from datafangst

status()

Method for polling the status of the instance from datafangst returns None if data not pushed to datafangst

Returns xml respons from datafangst

CHAPTER 4

Indices and tables

- genindex

Link to pnvdb on Github

Index

P

R

S

T

V