Python client for dataquality.pl Documentation

Release 0.5.0

Mikołaj Olszewski

Oct 27, 2022

Contents

Contents:

CHAPTER 1

Python client for dataquality.pl

Python library which allows to use http://dataquality.pl in the easy way.

- Free software: Apache Software License 2.0
- Documentation: https://dq-client.readthedocs.io.

1.1 Features

- Full API client
- Automatic encoding file conversion

1.2 Credits

This package was created by Algolytics dev team.

CHAPTER 2

Installation

2.1 Requirements

For the full functionality Python 3 is required.

2.2 Stable release

To install Python client for dataquality.pl, run this command in your terminal:

```
$ pip install dq-client
```

This is the preferred method to install Python client for dataquality.pl, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.3 From sources

The sources for Python client for dataquality.pl can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/Algolytics/dq_client
```

Or download the tarball:

```
$ curl  -OL https://github.com/Algolytics/dq_client/tarball/master
```

Once you have a copy of the source, you can install it with:

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

CHAPTER 3

Usage

To use Python client for dataquality.pl in a project:

```
from dq import DQClient, JobConfig


dq = DQClient('https://app.dataquality.pl', user='<USER_EMAIL>', token='<API_TOKEN>')
```

API token can be obtain on the page “Moje konto”.

3.1 Account

Check account status:

```
account = dq.account_status()

print(account.email)           # user email
print(account.balance)         # account balance
print(account.total_records)   # processed records
```

3.2 Jobs

3.2.1 List jobs

```
jobs = dq.list_jobs()

for job in jobs:
    print(job.id)                # job id
    print(job.name)              # human readable job name
    print(job.status)            # job status
```

(continues on next page)

(continued from previous page)

```
    print(job.start_date)         # job start date
    print(job.end_date)           # job end date
    print(job.source_records)     # how many records were applied
    print(job.processed_records)  # how many records were processed
    print(job.price)              # price for processed records
```

3.2.2 Create new job

```
input_data = '''"ID","ADRES"
6876,"34-404, PYZÓWKA, PODHALAŃSKA 100"
'''

job_config = JobConfig('my job')
job_config.input_format(field_separator=',', text_delimiter='"')
job_config.input_column(0, name='ID', function='PRZEPISZ')
job_config.input_column(1, name='ADRES', function='DANE_OGOLNE')
job_config.module_std(address=1)
job_config.extend(gus=True, geocode=True)

job = dq.submit_job(job_config, input_data=input_data)
↪          # with data in a variable

job = dq.submit_job(job_config, input_file='my_file.csv')
↪          # with data inside file

print(job.id)
print(job.name)
print(job.status)
...
```

3.2.3 Create new deduplication job

```
input_data = '''unikalne_id;imie_i_nazwisko;kod_pocztowy;miejscowosc;adres;email;tel;
↪CrmContactNumber;data
1;Jan Kowalski;37-611;Cieszanów ;Dachnów 189;abc@wp.pl;605936000;abc123;2017-11-08␣
↪12:00:00.000
2;Adam Mickiewicz Longchamps de Berier;66-400;Gorzów Wlkp.;Widok 24;qqq@ft.com;
↪48602567000;a2b2c2;2017-11-08 12:00:00.000
3;Barbara Łęcka;76-200;Słupsk;Banacha 7;bb@gazeta.pl;79174000;emc2;2017-11-08␣
↪12:00:00.000
4;KAROL NOWAK;22-122;LEŚNIOWICE;RAKOLUPY DU--E 55;kn@ll.pp;0;f112358;2017-11-08␣
↪12:00:00.000
5;Anna Maria Jopek;34-722;Podwilk;Podwilk 464;amj@gmail.com;606394000;eipi10;2017-11-
↪08 12:00:00.000
6;Mariusz Robert;37-611;Cieszanów ;Dachnów 189;abc@wp.pl;605936000;abc123;2017-11-08␣
↪12:00:00.000
'''

job_config = JobConfig('pr2')
job_config.input_format(field_separator=';', text_delimiter='"')
job_config.input_column(0, name='unikalne_id', function='ID_REKORDU')
job_config.input_column(1, name='imie_i_nazwisko', function='IMIE_I_NAZWISKO')
```

(continues on next page)

(continued from previous page)

```
job_config.input_column(2, name='kod_pocztowy', function='KOD_POCZTOWY')
job_config.input_column(3, name='miejscowosc', function='MIEJSCOWOSC')
job_config.input_column(4, name='adres', function='ULICA_NUMER_DOMU_I_MIESZKANIA')
job_config.input_column(5, name='email', function='EMAIL1')
job_config.input_column(6, name='tel', function='TELEFON1')
job_config.input_column(7, name='CrmContactNumber', function='PRZEPISZ')
job_config.input_column(8, name='data', function='CZAS_AKTUALIZACJI')
job_config.deduplication(on=True)
job_config.module_std(address=True, names=True, contact=True)
job_config.extend(gus=True, geocode=True, diagnostic=True)

job = dq.submit_job(job_config, input_data=input_data)

print(job)
...
```

Available column functions:

- **addresses**
  - KOD_POCZTOWY
  - MIEJSCOWOSC
  - ULICA_NUMER_DOMU_I_MIESZKANIA
  - ULICA
  - NUMER_DOMU
  - NUMER_MIESZKANIA
  - NUMER_DOMU_I_MIESZKANIA
  - WOJEWODZTWO
  - POWIAT
  - GMINA
- **names**
  - IMIE
  - NAZWISKO
  - NAZWA_PODMIOTU
  - IMIE_I_NAZWISKO
- **people/companies**
  - PESEL
  - NIP
  - REGON
- **contact**
  - EMAIL1
  - EMAIL2
  - TELEFON1

    - TELEFON2
- **dates**
    - DATA_URODZENIA
    - CZAS_AKTUALIZACJI
- **mixed**
    - DANE_OGOLNE
- **id**
    - ID_REKORDU
- **others**
    - PRZEPISZ
    - POMIN

To process input columns, you must enable the corresponding module. Method module_std is used to set active modules:

- address
- names
- contact
- id_numbers

For address module to be started it is necessary to ensure at least one column with the role listed below:

- DANE_OGOLNE
- KOD_POCZTOWY
- MIEJSCOWOSC

Analogously for other modules:

- **names require one of**
    - DANE_OGOLNE
    - IMIE
    - NAZWISKO
    - IMIE_I_NAZWISKO
    - NAZWA_PODMIOTU
- **contact**
    - DANE_OGOLNE
    - EMAIL1
    - EMAIL2
    - TELEFON1
    - TELEFON2
- **id**
    - DANE_OGOLNE

- PESEL
- NIP
- REGON

3.2.4 Check job state

```
state = dq.job_state('3f14e25e-9f6d-41ff-a4cb-942743a37b73')  # input parameter: job id

print(state)                                                  # 'WAITING' or 'FINISHED'
```

3.2.5 Cancel job

```
dq.cancel_job('3f14e25e-9f6d-41ff-a4cb-942743a37b73')  # input parameter: job id
```

3.2.6 Retrieve job report

```
report = dq.job_report('3f14e25e-9f6d-41ff-a4cb-942743a37b73')  # input parameter: job id

print(report.quality_issues)
print(report.quality_names)
print(report.results)
```

3.2.7 Save job results

```
dq.job_results('3f14e25e-9f6d-41ff-a4cb-942743a37b73', 'output.csv')
```

3.2.8 Delete job and its results

```
dq.delete_job('3f14e25e-9f6d-41ff-a4cb-942743a37b73')  # input parameter: job id
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/Algolytics/dq_client/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

Python client for dataquality.pl could always use more documentation, whether as part of the official Python client for dataquality.pl docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/Algolytics/dq_client/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *dq-client* for local development.

1. Fork the *dq-client* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/dq_client.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv dq_client
$ cd dq-client/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 dq tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/Algolytics/dq_client/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_dq_client
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search