aioodbc Documentation

Release 0.0.3

Nikolay Novik

September 20, 2016

Contents

aioodbc is Python 3.5+ module that makes possible accessing **ODBC_** databases with asyncio. It is rely on awesome pyodbc library, preserve same look and feel. *aioodbc* was written *async/await* syntax (PEP492) thus not compatible with Python older then 3.5. Internally *aioodbc* employ threads to avoid blocking the event loop, btw threads are not that bad as you think :)

CHAPTER 1

Features

- Implements *asyncio* *DBAPI* *like* interface for *ODBC*. It includes aioodbc-connection, aioodbc-cursor and aioodbc-pool objects.
- Support connection pooling.

CHAPTER 2

Source code

The project is hosted on GitHub

Please feel free to file an issue on bug tracker if you have found a bug or have some suggestion for library improvement.

The library uses Travis for Continious Integration and Coveralls for coverage reports.

CHAPTER 3

Dependencies

- Python 3.5 (PEP492 coroutines)
- pyodbc
- unixODBC

CHAPTER 4

Authors and License

The `aioodbc` package is written by Nikolay Novik and aio-libs contributors. It's MIT licensed.

Feel free to improve this package and send a pull request to GitHub.

CHAPTER 5

Contents:

5.1 Examples of aioodbc usage

Below is a list of examples from aioodbc/examples

Every example is a correct tiny python program.

5.1.1 Basic Usage

Basic example, executes query that return important number 42.

```
import asyncio
import aioodbc


loop = asyncio.get_event_loop()


async def test_example():
    dsn = 'Driver=SQLite;Database=sqlite.db'
    conn = await aioodbc.connect(dsn=dsn, loop=loop)

    cur = await conn.cursor()
    await cur.execute("SELECT 42;")
    r = await cur.fetchall()
    print(r)
    await cur.close()
    await conn.close()

loop.run_until_complete(test_example())
```

Example of query execution in connection pool.

```
import asyncio
import aioodbc


loop = asyncio.get_event_loop()


async def test_pool():
    dsn = 'Driver=SQLite;Database=sqlite.db'
```

```
    pool = await aioodbc.create_pool(dsn=dsn, loop=loop)

    async with pool.acquire() as conn:
        cur = await conn.cursor()
        await cur.execute("SELECT 42;")
        r = await cur.fetchall()
        print(r)
        await cur.close()
        await conn.close()
    pool.close()
    await pool.wait_closed()

loop.run_until_complete(test_pool())
```

Example of using async context managers with Pool, Connection and Cursor objects.

```
import asyncio
import aioodbc


loop = asyncio.get_event_loop()


async def test_example():
    dsn = 'Driver=SQLite;Database=sqlite.db'

    async with aioodbc.create_pool(dsn=dsn, loop=loop) as pool:
        async with pool.acquire() as conn:
            async with conn.cursor() as cur:
                await cur.execute('SELECT 42;')
                val = await cur.fetchone()
                print(val)

loop.run_until_complete(test_example())
```

5.2 Glossary

DBAPI PEP 249 – Python Database API Specification v2.0

ipdb ipdb exports functions to access the IPython debugger, which features tab completion, syntax highlighting, better tracebacks, better introspection with the same interface as the pdb module.

MySQL A popular database server.

http://www.mysql.com/

ODBC Open Database Connectivity (ODBC) is a standard programming language middleware application programming interface (API) for accessing database management systems (DBMS)

pep8 Python style guide checker

pep8 is a tool to check your Python code against some of the style conventions in PEP 8 – Style Guide for Python Code.

pyflakes passive checker of Python programs

A simple program which checks Python source files for errors.

Pyflakes analyzes programs and detects various errors. It works by parsing the source file, not importing it, so it is safe to use on modules with side effects. It's also much faster.

https://pypi.python.org/pypi/pyflakes

5.3 Contributing

Thanks for your interest in contributing to `aioodbc`, there are multiple ways and places you can contribute.

5.3.1 Reporting an Issue

If you have found issue with *aioodbc* please do not hesitate to file an issue on the GitHub project. When filing your issue please make sure you can express the issue with a reproducible test case.

When reporting an issue we also need as much information about your environment that you can include. We never know what information will be pertinent when trying narrow down the issue. Please include at least the following information:

- Version of *aioodbc* and *python*.
- Version of your ODBC database
- Version of database ODBC driver
- Version of unixODBC
- Platform you're running on (OS X, Linux, Windows).

5.3.2 Instructions for contributors

In order to make a clone of the GitHub repo: open the link and press the "Fork" button on the upper-right menu of the web page.

I hope everybody knows how to work with git and github nowadays :)

Work flow is pretty straightforward:

1. Clone the GitHub repo
2. Make a change
3. Make sure all tests passed
4. Commit changes to own aioodbc clone
5. Make pull request from github page for your clone

5.3.3 Preconditions for running aioodbc test suite

We expect you to use a python virtual environment and docker to run our tests.

There are several ways to make a virtual environment.

If you like to use *virtualenv* please run:

```
$ cd aioodbc
$ virtualenv --python=`which python3.5` venv
```

For standard python *venv*:

```
$ cd aioodbc
$ python3.5 -m venv venv
```

For *virtualenvwrapper*:

```
$ cd aioodbc
$ mkvirtualenv --python=`which python3.5` aioodbc
```

There are other tools like *pyvenv* but you know the rule of thumb now: create a python3.5 virtual environment and activate it.

After that please install libraries required for development:

```
$ pip install -r requirements-dev.txt
```

We also recommend to install *ipdb* but it's on your own:

```
$ pip install ipdb
```

Congratulations, you are ready to run the test suite

5.3.4 Install database

You do not need to install any databases, docker will pull images and create containers for you automatically, after the tests, containers will be removed.

5.3.5 Run aioodbc test suite

After all the preconditions are met you can run tests typing the next command:

```
$ make test
```

Or if you want to run only one particular test:

```
$ py.test tests/test_connection.py -k test_basic_cursor
```

The command at first will run the static and style checkers (sorry, we don't accept pull requests with *pep8* or *pyflakes* errors).

On *flake8* success the tests will be run.

Please take a look on the produced output.

Any extra texts (print statements and so on) should be removed.

5.3.6 Tests coverage

We are trying hard to have good test coverage; please don't make it worse.

Use:

```
$ make cov
```

to run test suite and collect coverage information. Once the command has finished check your coverage at the file that appears in the last line of the output: `open file:///.../aioodbc/htmlcov/index.html`

Please go to the link and make sure that your code change is covered.

5.3.7 Documentation

We encourage documentation improvements.

Please before making a Pull Request about documentation changes run:

```
$ make doc
```

Once it finishes it will output the index html page `open file:///.../aioodbc/docs/_build/html/index.html`.

Go to the link and make sure your doc changes looks good.

5.3.8 The End

After finishing all steps make a GitHub Pull Request, thanks.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

D

I

M

O

P