aiomysql-core

Release 0.0.3

May 11, 2020

Contents

This part of the documentation will show you how to get started in using Aiomysql-Core with aiomysql.

CHAPTER 1

Installation

Install Aiomysql-Core with `pip`

```
pip install aiomysql-core
```

The development version can be downloaded from its page at GitHub.

```
git clone https://github.com/linzhiming0826/aiomysql-core.git
cd aiomysql-core
python setup.py develop
```

Aiomysql-Core has the following dependencies (which will be automatically installed if you use `pip`):

- aiomysql version 0.0.20 or greater

Aiomysql-Core requires Python version 3.6 or greater

CHAPTER 2

Quickstart

It's time to write your first example. This guide assumes you have a working understanding of aiomysql, and that you have already installed both aiomysql and Aiomysql-Core. If not, then follow the steps in the *Installation* section.

2.1 A Minimal Example

A minimal Aiomysql-Core example looks like this:

```
import asyncio
import aiomysql
from aiomysql_core import AioMysqlCore


async def test_example(loop):
    pool = await aiomysql.create_pool(host='', port=3306,
                                      user='', password='',
                                      db='', loop=loop)
    core = AioMysqlCore(pool=pool)
    rows = await core.query('select * from users where uid=%s', 113)
    print(rows)
    rows = await core.gener('select * from users limit 100')
    async for row in rows:
        print(row.id)
    row = await core.get('select * from users where uid=%(uid)s', {'uid': 113})
    print(row)
    rowcount = await core.execute_rowcount('select * from users where uid=%(uid)s', {
↪'uid': 113})
    print(rowcount)
    pool.close()
    await pool.wait_closed()
```

(continues on next page)

(continued from previous page)

```
loop = asyncio.get_event_loop()
loop.run_until_complete(test_example(loop))
```

2.2 Query

Execute a query and return number of affected rows Looks like this:

```
import asyncio
import aiomysql
from aiomysql_core import AioMysqlCore


async def test_example(loop):
    pool = await aiomysql.create_pool(host='', port=3306,
                                      user='', password='',
                                      db='', loop=loop)
    core = AioMysqlCore(pool=pool)
    rows = await core.query('select * from users where uid=%s', 113)
    print(rows)
    pool.close()
    await pool.wait_closed()


loop = asyncio.get_event_loop()
loop.run_until_complete(test_example(loop))
```

2.3 Gener

Execute a query and return an generator for the number of affected rows looks like this:

```
import asyncio
import aiomysql
from aiomysql_core import AioMysqlCore


async def test_example(loop):
    pool = await aiomysql.create_pool(host='', port=3306,
                                      user='', password='',
                                      db='', loop=loop)
    core = AioMysqlCore(pool=pool)
    rows = await core.gener('select * from users limit 100')
    async for row in rows:
        print(row.id)
    pool.close()
    await pool.wait_closed()


loop = asyncio.get_event_loop()
loop.run_until_complete(test_example(loop))
```

2.4 Get

Execute a query and return number of affected first row looks like this:

```
import asyncio
import aiomysql
from aiomysql_core import AioMysqlCore


async def test_example(loop):
    pool = await aiomysql.create_pool(host='', port=3306,
                                      user='', password='',
                                      db='', loop=loop)
    core = AioMysqlCore(pool=pool)
    row = await core.get('select * from users where uid=%s', 113)
    print(row)
    pool.close()
    await pool.wait_closed()


loop = asyncio.get_event_loop()
loop.run_until_complete(test_example(loop))
```

2.5 Execute

Execute a query and return lastrowid looks like this:

```
import asyncio
import aiomysql
from aiomysql_core import AioMysqlCore


async def test_example(loop):
    pool = await aiomysql.create_pool(host='', port=3306,
                                      user='', password='',
                                      db='', loop=loop)
    core = AioMysqlCore(pool=pool)
    lastrowid = await core.execute('update users set name=%(name)s where uid=%(uid)s',
↪ {'name': 'core', 'uid': 113})
    print(lastrowid)
    pool.close()
    await pool.wait_closed()


loop = asyncio.get_event_loop()
loop.run_until_complete(test_example(loop))
```

2.6 Execute_lastrowid

Execute a query and return lastrowid looks like this:

```
import asyncio
import aiomysql
from aiomysql_core import AioMysqlCore


async def test_example(loop):
    pool = await aiomysql.create_pool(host='', port=3306,
                                      user='', password='',
                                      db='', loop=loop)
    core = AioMysqlCore(pool=pool)
    lastrowid = await core.execute_lastrowid('update users set name=%(name)s where␣
↪uid=%(uid)s', {'name': 'core', 'uid': 113})
    print(lastrowid)
    pool.close()
    await pool.wait_closed()


loop = asyncio.get_event_loop()
loop.run_until_complete(test_example(loop))
```

2.7 Execute_rowcount

Execute a query and return rowcount looks like this:

```
import asyncio
import aiomysql
from aiomysql_core import AioMysqlCore


async def test_example(loop):
    pool = await aiomysql.create_pool(host='', port=3306,
                                      user='', password='',
                                      db='', loop=loop)
    core = AioMysqlCore(pool=pool)
    lastrowid = await core.execute_rowcount('update users set name=%(name)s where uid=
↪%(uid)s', {'name': 'core', 'uid': 113})
    print(lastrowid)
    pool.close()
    await pool.wait_closed()


loop = asyncio.get_event_loop()
loop.run_until_complete(test_example(loop))
```

2.8 Executemany

Run several data against one query and return lastrowid looks like this:

```
import asyncio
import aiomysql
from aiomysql_core import AioMysqlCore
```

(continues on next page)

(continued from previous page)

```
async def test_example(loop):
    pool = await aiomysql.create_pool(host='', port=3306,
                                      user='', password='',
                                      db='', loop=loop)
    core = AioMysqlCore(pool=pool)
    data = [(0, "bob", 21, 123), (1, "jim", 56, 45), (2, "fred", 100, 180)]
    lastrowid = await core.executemany("insert into users (id, name, age, height)␣
↪values (%s,%s,%s,%s)", data)
    print(lastrowid)
    pool.close()
    await pool.wait_closed()


loop = asyncio.get_event_loop()
loop.run_until_complete(test_example(loop))
```

2.9 Executemany_lastrowid

Run several data against one query and return lastrowid looks like this:

```
import asyncio
import aiomysql
from aiomysql_core import AioMysqlCore


async def test_example(loop):
    pool = await aiomysql.create_pool(host='', port=3306,
                                      user='', password='',
                                      db='', loop=loop)
    core = AioMysqlCore(pool=pool)
    data = [(0, "bob", 21, 123), (1, "jim", 56, 45), (2, "fred", 100, 180)]
    lastrowid = await core.executemany_lastrowid("insert into users (id, name, age,␣
↪height) values (%s,%s,%s,%s)", data)
    print(lastrowid)
    pool.close()
    await pool.wait_closed()


loop = asyncio.get_event_loop()
loop.run_until_complete(test_example(loop))
```

2.10 Executemany_rowcount

Run several data against one query and return rowcount looks like this:

```
import asyncio
import aiomysql
from aiomysql_core import AioMysqlCore
```

(continues on next page)

(continued from previous page)

```
async def test_example(loop):
    pool = await aiomysql.create_pool(host='', port=3306,
                                      user='', password='',
                                      db='', loop=loop)
    core = AioMysqlCore(pool=pool)
    data = [(0, "bob", 21, 123), (1, "jim", 56, 45), (2, "fred", 100, 180)]
    rowcount = await core.executemany_rowcount("insert into users (id, name, age,␣
↪height) values (%s,%s,%s,%s)", data)
    print(rowcount)
    pool.close()
    await pool.wait_closed()


loop = asyncio.get_event_loop()
loop.run_until_complete(test_example(loop))
```

2.11 SQLAlchemy

looks like this:

```
import asyncio
from aiomysql.sa import create_engine
from aiomysql_core import AioMysqlAlchemyCore
from sqlalchemy import Column, Integer, String, MetaData, Table

metadata = MetaData()

Test = Table('test', metadata,
             Column('id', Integer, primary_key=True),
             Column('content', String(255), server_default="")
             )


async def test_example(loop):
    config = {'user': '', 'password': '', 'db': '',
              'host': '', 'port': 3306, 'autocommit': True, 'charset': 'utf8mb4'}
    engine = await create_engine(loop=loop, **config)
    core = AioMysqlAlchemyCore(engine=engine)
    # insert
    doc = {'content': 'insert'}
    clause = Test.insert().values(**doc)
    rowcount = await core.execute_rowcount(clause)
    print(rowcount)
    # search
    clause = Test.select().where(Test.c.id == 1).limit(1)
    row = await core.get(clause)
    print(row.id, row.content)
    clause = Test.select().where(Test.c.id > 1)
    rows = await core.query(clause)
    async for row in rows:
        print(row.id, row.content)
    # update
    doc = {'content': 'update'}
    clause = Test.update().values(**doc).where(Test.c.id == 1)
```

(continues on next page)

(continued from previous page)

```
    rowcount = await core.execute_rowcount(clause)
    print(rowcount)
    # delete
    clause = Test.delete().where(Test.c.id == 1)
    rowcount = await core.execute_rowcount(clause)
    print(rowcount)
    await engine.wait_closed()


loop = asyncio.get_event_loop()
loop.run_until_complete(test_example(loop))
```