oop_paper

February 23, 2022

1 A brief introduction to Python object-oriented programming and connecting to Postgresql database.

Krzysztof Kadowski

kkadowski@gmail.com

https://github.com/kkadowski/OOP_python_with_postgresql

1.1 Creating a new database

Before you start coding, you need to install the Postgresql server and create a users database using query:

CREATE DATABASE users;

1.2 Writing ini file

All the connection parameters to the database will be stored in the database.ini file in the form of section:

[postgresql]

host=localhost

database=users

user=postgres

password=postgres

1.3 Compatibility check

In the compat.py file we check compatibility with the Python language.

[ ]:
```
# -*- coding: utf-8 -*-
"""
    compat.py - Python compatibility check
"""

import sys

PY2 = sys.version_info[0] == 2
```

```
if not PY2:
    strtypes = (str, )
else:
    strtypes = (str, unicode)
```

1.4 Creating class dbConnection

[ ]:
```
"""
    db_con.py - dbConnection class:
    connection params from database.ini file
"""

from pickle import UNICODE
import sys
import psycopg2
from psycopg2 import connect
from psycopg2.extras import DictConnection
from compat import PY2
from configparser import ConfigParser

if PY2:
    from psycopg2.extras import register_type, unicode
    register_type(UNICODE)

class dbConnection:

    # reading connection params
    def __init__(self, filename='database.ini', section='postgresql'):
        self.parser = ConfigParser()
        self.parser.read(filename)
        self.db = {}

        if self.parser.has_section(section):
            self.params = self.parser.items(section)
            for param in self.params:
                self.db[param[0]] = param[1]
        else:
            raise Exception('Section {0} can\'t be found in {1} file.'.
 ↪format(section, filename))

    # connection to my database
    def connect(self):
        self.conn = None
        try:
            self.conn = psycopg2.connect(host = self.db['host'],
                                         database = self.db['database'],
                                         user = self.db['user'],
                                    password = self.db['password']
                                    )
        except (Exception, psycopg2.DatabaseError) as err:
            print(f"Database connection error:  {err}")

    # version of database
    def db_version(self):
        try:
            self.connect()
            self.cur = self.conn.cursor()
            self.cur.execute('SELECT version()')
            self.db_v = self.cur.fetchone()
            print(f"Database version: \n \t {self.db_v}")
            self.close()
        except (Exception, psycopg2.DatabaseError) as err:
            print(f"Database connection error: {err}")
        finally:
            self.close()

    # closing the connection
    def close(self):
        if self.conn and not self.conn.closed:
            self.conn.close()
        self.conn = None

    # commiting query
    def commit(self):
        self.conn.commit()

    # rollbacking query
    def rollback(self):
        self.conn.rollback()

    # executing query drop / create / insert...
    def execute(self, query, args=None):
        if self.conn is None or self.conn.closed:
            self.connect()
        curs = self.conn.cursor()
        try:
            curs.execute(query, args)
        except Exception as ex:
            self.conn.rollback()
            curs.close()
            raise ex
        return curs

    # executing query COUNT / SUM / MIN...
    def fetchone(self, query, args=None):
        curs = self.execute(query, args)
        row = curs.fetchone()
        curs.close()
        return row

    # executing query returning more rows than one
    def fetchall(self, query, args=None):
        curs = self.execute(query, args)
        rows = curs.fetchall()
        curs.close()
        return rows

    # copying records of the table to the file
    def copy_to(self, path_file, table_name, sep=','):
        if self.conn is None or self.conn.closed:
            self.connect()
        with open(path_file, 'w+') as f:
            curs = self.conn.cursor()
            try:
                curs.copy_to(f, table_name, sep)
            except:
                curs.close()
                raise Exception('Problem with writing to the file '.
↪format(path_file))

    # copying records from the file to the table
    def copy_from(self, path_file, table_name, sep=','):
        if self.conn is None or self.conn.closed:
            self.connect()
        with open(path_file, 'r') as f:
            curs = self.conn.cursor()
            try:
                curs.copy_from(f, table_name, sep)
            except:
                curs.close()
                raise Exception('Problem with copying from the file {0} to the␣
↪table {1}'.format(path_file, table_name))
```

1.5 Example of use

[ ]:
```
"""
app_test.py - example of use file
"""
import os
import pytest
from db_con import dbConnection
```

```python
# Test of reading params from file
db = dbConnection()

# Test of version of the database
db.db_version()

# Test of executing CREATE TABLE query
db.execute("CREATE TABLE IF NOT EXISTS users (id SERIAL PRIMARY KEY, f_name␣
 ↪VARCHAR(16) NOT NULL, l_name VARCHAR(32) NOT NULL) ")
db.commit()

# Test of inserting data to users table
db.execute("INSERT INTO users (f_name, l_name) VALUES ('John', 'Doe')")
db.execute("INSERT INTO users (f_name, l_name) VALUES ('Anna', 'Green')")
db.commit()

# Test of SELECT query
rows = db.fetchall("SELECT * FROM users")
for row in rows:
    print(row)

# Test of counting rows query
row = db.fetchone("SELECT COUNT(*) FROM users")
print("Num of records: ", row[0])

# Test of copy of the table to csv file
db.copy_to('users.csv', 'users', ':' )

# Test of copy records from file to new table
db.execute("CREATE TABLE IF NOT EXISTS new_users (id SERIAL PRIMARY KEY, f_name␣
 ↪VARCHAR(16), l_name VARCHAR(32))")
db.commit()
db.copy_from('userse.csv', 'new_users', ':' )
rows = db.fetchall("SELECT * FROM new_users")
for row in rows:
    print(row)

# Test of deleting tables
db.execute("DROP TABLE users")
db.commit()
db.execute("DROP TABLE new_users")
db.commit()

# Closing the connection
db.close()
```