Python/MySQL

MySQLdb هى Interface بتسمحلك بالتعامل مع MySQL من خلال بايثون
اوكى القصة بدأت ان اتعمل wrap لل MySQL C APIs بصورة OO
فى امثلة لشكل ال APIs
http://mysql-python.sourceforge.net/MySQLdb.html#id5

جميل احنا تعاملنا كله من خلال ال MySQLdb وهى عملت wrap ل _mysql انترفيس لضمان التكافئ مع ال DB API specifications هتكون PEP 249

اولا ال connect

connect(...)

هى المسئولة عن انشاء الإتصال بقاعدة البيانات وبتعمل ريترن ب Connection Object لازم عشان ننشئ اتصال نحتاج شوية معلومات زى ال

- host

  ودا بيعبر عن الهوست اللى هيتم الإتصال عليه(الإفتراضى localhost)

- user

  اسم المستخدم (الإفتراضى المستخدم الحالى)

- passwd

  الباسورد الخاص بإسم المستخدم (افتراضى لايوجد)

- db

  قاعدة البيانات (الإفتراضى لأ)

- port

  زى مانت عارف ال MySQL ليها server ودا البورت بيعبر عن ال TCP Port اللى بيستخدمه السرفر وافتراضيا 3306 (عدله لو قمت بتغييره!)

- ssl

  لإنشاء SSL Connection (ملحوظة: throws exception لو غير مدعم!)

- compress

  لتفعيل ال compression (الإفتراضى لأ)

- connect_timeout

  تحدد زمن ال timeout

- charset

  اذا تم اضافتها هيتم تضمين use_unicode=True

- sqlmode

  لتحديد ال sqlmode (يفضل تراجع MySQL documentation)

-

تقدر تحدد الكثير من الإعدادات كل اللى عليك تراجع ال MySQL Documentation

apilevel

بتحدد اى DB API مدعمة ؟ الحالى 2.0

1- اعمل import ل MySQLdb كالتالى

```
>>> import MySQLdb as ms
```

--انا خليت ms ك alias طبعا انت حر فى كيفية الإستدعاء

```
>>> ms.apilevel
'2.0'
```

```
>>> ms.threadsafety
1
```

اوكى ايه معنى ال threadsafety اصلا ؟
هى عبارة عن رقم بين [0, 3]

0:

بيعنى ان ال threads مش تقدر تشارك فى ال module

1:

ان ال threads تقدر تشارك فى ال module ولكن مش ال connections

2-

ان ال threads تقدر تشارك فى ال module وال connections ولكن مش ال cursors (هنتكلم عنها)

3-

اعلى شئ وهى امكانية المشاركة الكاملة فى ال module, connections و ال cursors

paramstyle

سترينج بيعبر عن طريقة التعامل مع ال queries من خلال المدخلات يعنى مثلا احيانا فى ناس بتستخدم ؟ (علامة استفهام) او طريقة %s او حتى استخدام الأرقام :1 و :2 وهكذا (حسب الترتيب) فالديفولت هو format

```
>>> ms.paramstyle
'format'
```

ال Exceptions/Errors المرتبطة هنا هما Errors مشتقين من Error
ينقسمو الى
1- IntefaceError ودا بيعبر عن ايرور فى الإنترفيس المستخدمة مش ال داتابيز
2- DatabaseError بيعبر عن ايرور فى قاعدة البيانات وتنقسم لكذا شئ اهمهم
DataError مشاكل مع الداتا
OperationalError ايرور اثناء تنفيذ عملية معينة
ProgrammingError فشل فى تنفيذ sql command معين
NotSupportedError عملية غير مدعمة!

1- ال Connection Objects
هى اوبجكتس بيتم اعادتها عند الإتصال بقاعدة بيانات وليها عدة ميثودز

1- close()

لغلق الإتصال

2-commit()

لتنفيذ ال transaction الحالى (مش هتفرق فى حال مش فى تدعيم لل transactions اصلا او ال auto commit مفعلة)

3-rollback()

الغاء ال transaction الحالى (نفس الملحوظة السابقة ولكن لاحظ فى حال انهاء الإتصال وعدم ال commit هيتم عمل rollback اتوماتيك!)

4- cursor()
5- set_sql_mode(sqlmode)

بتحدد ال sqlmode (يفضل تراجع ال MySQL documentation)

6- set_character_set(charset)

تحديد ال characterset

للحصول على cursor (هنتعرض ليه)

2- ال Cursor Objects

بكل بساطة طالما عندك اتصال بقاعدة بيانات يبقة انت محتاج * تتفاعل * معاها عن طريق تنفيذ SQL statements معينة فال cursor بيقوم بدور الوسيط بينكم بيسمحلك تنفذ SQL statements وبيسمحلك تتعامل مع ال rows الناتجة

ال Cursor ليه شوية fields و methods اهمهم

execute(sqlQuery, args)

بتقوم بتنفيذ Sql Statement على قاعدة البيانات وبيتم تجهيزها قبل التنفيذ ب args فى حال لو انت قررت تعمل late binding

where name=?

او

where name=:name

وهكذا

rowcount

عدد الصفوف الناتجة من تنفيذ اخر امر

callproc(proc, args)

لإستدعاء stored procedure!

fetchone()

الحصول على صف واحد من الناتج

fetchmany()

للحصول على عدد معين من الصفوف تم تحديده من خلال arraysize (تبع ال cursor وبتعبر عن عدد الصفوف)

arraysize

بتعبر عن رقم الصفوف اللى هيتم اعادته من خلال ال fetchmany method

fetchall()

للحصول على كل ال الصفوف الناتجة

rownumber

ال index الخاص ب ال cursor!

نختم بمثال (جزء من برنامج حالى استخدمت فيه MySQL ك backend)

```
CREATE TABLE `users` (
  `id` int(11) NOT NULL auto_increment,
  `username` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `state` tinyint(2) NOT NULL,
  PRIMARY KEY  (`id`),
  UNIQUE KEY `username` (`username`)
) ;
```

محتاجين نطبق crud على الجدول دا من خلال Python/MySQL

Create/Read/Updata/Delete

1- استدعى MySQLdb

```
import MySQLdb as ms
```

2- انشئ كلاس DBMan

```
class DBMan(object):

    def __init__(self, dbname="pyim"):


        self._dbname=dbname
        self._sqlconnection=ms.connect(host="localhost",
                                       user="root",
                                       passwd="",
                                       db="pyim")

        self._sqlcursor=self._sqlconnection.cursor()
```

لاحظ فى ال constructor محدين اسم ال database ك pyim
وحددنا البيانات وانشئنا Connection object بإسم self._sqlconnection
وحصلنا على cursor منه بإسم self._sqlcursor

إضافة مستخدم جديد

```
    def addUser(self, username, pwd, state=State.Offline):
        #State.Offline=1

        md5edpass=self._md5(pwd)
        sqlstmt="INSERT INTO users VALUES(NULL, '%s', '%s', %d)"%(username, md5edpass, state)
        try:
            self._sqlcursor.execute(sqlstmt)
            self._sqlconnection.commit()
        except Exception, e:
            print e.message
```

حذف مستخدم

```
    def deleteUser(self, username):

        sqlstmt="DELETE FROM users WHERE username='%s'"%username
        try:
            self._sqlcursor.execute(sqlstmt)
            self._sqlconnection.commit() #commit
        except Exception, e:
            print e.message
```

تسجيل دخول

```
    def login(self, username, pwd):

        md5edpass=self._md5(pwd)
        sqlstmt="SELECT username, password FROM users WHERE username='%s' AND
password='%s'"%(username, md5edpass)
```

```
    self._sqlcursor.execute(sqlstmt)
    if self._sqlcursor.fetchone():
        self.setState(State.Online, username)
```

تغيير الحالة

```
def setState(self, state, username):

    sqlstmt="UPDATE users SET state=%d WHERE username='%s'"%(state, username)
    try:
        self._sqlcursor.execute(sqlstmt)
    except Exception, e:
        print e.message
```

عرض الكل

```
def getAllUsers(self):

    sqlstmt="SELECT username, state FROM users"
    self._sqlcursor.execute(sqlstmt)
    for row in self._sqlcursor.fetchall():
        yield row[0], row[1]
```

ملحوظة: انا هنا ناقشت MySQLdb من خلال مفهوم ال DB API بمعنى ان نفس المبادئ هتلقيها ثابتة فى اى انترفيس هتستخدمها
ومازلنا منتظرين DB API 3

Refs:
[MySQLdb 1.2.2 docs]
[Python DB API Specifications v2]