Mysql驱动对比

# MySQL-python

MySQL-python 又叫 MySQLdb，是 Python 连接 MySQL 最流行的一个驱动，很多框架都也是基于此库进行开发，遗憾的是它只支持 Python2.x，而且安装的时候有很多前置条件，因为它是基于C开发的库，在 Windows 平台安装非常不友好，经常出现失败的情况，现在基本不推荐使用，取代的是它的衍生版本。

# Mysqlclient

由于 MySQL-python 年久失修，后来出现了它的 Fork 版本 mysqlclient，完全兼容 MySQLdb，同时支持 Python3.x，是 Django ORM的依赖工具，如果你想使用原生 SQL 来操作数据库，那么推荐此驱动。安装方式和 MySQLdb 是一样的，Windows 可以在 <https://www.lfd.uci.edu/~gohlke/pythonlibs/#mysqlclient> 网站找到 对应版本的 whl 包下载安装。

# PyMySQL

PyMySQL 是纯 Python 实现的驱动，速度上比不上 MySQLdb，最大的特点可能就是它的安装方式没那么繁琐，同时也兼容 MySQL-python。性能比MySQL-python。

# 为了兼容mysqldb，只需要加入

pymysql.install\_as\_MySQLdb()

# peewee

写原生 SQL 的过程非常繁琐，代码重复，没有面向对象思维，继而诞生了很多封装 wrapper 包和 ORM 框架，ORM 是 Python 对象与数据库关系表的一种映射关系，有了 ORM 你不再需要写 SQL 语句。提高了写代码的速度，同时兼容多种数据库系统，如sqlite, mysql、postgresql，付出的代价可能就是性能上的一些损失。如果你对 Django 自带的 ORM 熟悉的话，那么 peewee的学习成本几乎为零。它是 Python 中是最流行的 ORM 框架。

# SQLAlchemy

如果想找一种既支持原生 SQL，又支持 ORM 的工具，那么 SQLAlchemy 是最好的选择，它非常接近 Java 中的 Hibernate 框架。

1. **用 Python 连接 MySQL 的几种方式详解**

 更新时间：2018年04月04日 10:26:57   作者：FOOFISH-PYTHON之禅   [[https://www.jb51.net/skin/2018/images/text-message.png](https://www.jb51.net/article/137722.htm#comments) 我要评论](https://www.jb51.net/article/137722.htm#comments)

这篇文章主要介绍了用 Python 连接 MySQL 的几种方式，大家可以根据实际情况选择合理的连接方式,需要的朋友可以参考下

尽管很多 NoSQL 数据库近几年大放异彩，但是像 MySQL 这样的关系型数据库依然是互联网的主流数据库之一，每个学 Python 的都有必要学好一门数据库，不管你是做数据分析，还是网络爬虫，Web 开发、亦或是机器学习，你都离不开要和数据库打交道，而 MySQL 又是最流行的一种数据库，这篇文章介绍 Python 操作 MySQL 的几种方式，你可以在实际开发过程中根据实际情况合理选择。

**1、MySQL-python**

MySQL-python 又叫 MySQLdb，是 Python 连接 MySQL 最流行的一个驱动，很多框架都也是基于此库进行开发，遗憾的是它只支持 Python2.x，而且安装的时候有很多前置条件，因为它是基于C开发的库，在 Windows 平台安装非常不友好，经常出现失败的情况，现在基本不推荐使用，取代的是它的衍生版本。

# 前置条件

sudo apt-get install python-dev libmysqlclient-dev # Ubuntu

sudo yum install python-devel mysql-devel # Red Hat / CentOS

# 安装

pip install MySQL-python

Windows 直接通过下载 exe 文件安装，公众号回复「win」获取下载链接

#!/usr/bin/python

import MySQLdb

db = MySQLdb.connect(

host="localhost", # 主机名

user="john", # 用户名

passwd="megajonhy", # 密码

db="jonhydb") # 数据库名称

# 查询前，必须先获取游标

cur = db.cursor()

# 执行的都是原生SQL语句

cur.execute("SELECT \* FROM YOUR\_TABLE\_NAME")

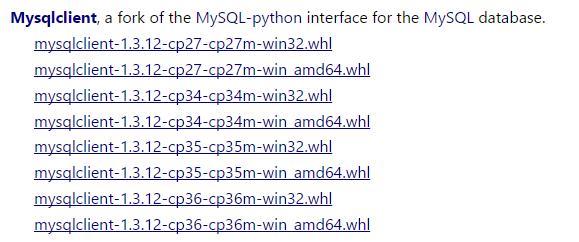
for row in cur.fetchall():

print(row[0])

db.close()

**2、mysqlclient**

由于 MySQL-python 年久失修，后来出现了它的 Fork 版本 mysqlclient，完全兼容 MySQLdb，同时支持 Python3.x，是 Django ORM的依赖工具，如果你想使用原生 SQL 来操作数据库，那么推荐此驱动。安装方式和 MySQLdb 是一样的，Windows 可以在 <https://www.lfd.uci.edu/~gohlke/pythonlibs/#mysqlclient> 网站找到 对应版本的 whl 包下载安装。



# Windows安装

pip install some-package.whl

# linux 前置条件

sudo apt-get install python3-dev # debian / Ubuntu

sudo yum install python3-devel # Red Hat / CentOS

brew install mysql-connector-c # macOS (Homebrew)

pip install mysqlclient

**3、PyMySQL**

PyMySQL 是纯 Python 实现的驱动，速度上比不上 MySQLdb，最大的特点可能就是它的安装方式没那么繁琐，同时也兼容 MySQL-python

pip install PyMySQL

# 为了兼容mysqldb，只需要加入

pymysql.install\_as\_MySQLdb()

一个例子

import pymysql

conn = pymysql.connect(host='127.0.0.1', user='root', passwd="xxx", db='mysql')

cur = conn.cursor()

cur.execute("SELECT Host,User FROM user")

for r in cur:

print(r)

cur.close()

conn.close()

**4、peewee**

写原生 SQL 的过程非常繁琐，代码重复，没有面向对象思维，继而诞生了很多封装 wrapper 包和 ORM 框架，ORM 是 Python 对象与数据库关系表的一种映射关系，有了 ORM 你不再需要写 SQL 语句。提高了写代码的速度，同时兼容多种数据库系统，如sqlite, mysql、postgresql，付出的代价可能就是性能上的一些损失。如果你对 Django 自带的 ORM 熟悉的话，那么 peewee的学习成本几乎为零。它是 Python 中是最流行的 ORM 框架。

pip install peewee

一个例子

import peewee

from peewee import \*

db = MySQLDatabase('jonhydb', user='john', passwd='megajonhy')

class Book(peewee.Model):

author = peewee.CharField()

title = peewee.TextField()

class Meta:

database = db

Book.create\_table()

book = Book(author="me", title='Peewee is cool')

book.save()

for book in Book.filter(author="me"):

print(book.title)

官方文档：<http://docs.peewee-orm.com/en/latest/peewee/installation.html>

**5、SQLAlchemy**

如果想找一种既支持原生 SQL，又支持 ORM 的工具，那么 SQLAlchemy 是最好的选择，它非常接近 Java 中的 Hibernate 框架。

from sqlalchemy import create\_engine

from sqlalchemy.orm import sessionmaker

from sqlalchemy\_declarative import Address, Base, Person

class Address(Base):

\_\_tablename\_\_ = 'address'

id = Column(Integer, primary\_key=True)

street\_name = Column(String(250))

engine = create\_engine('sqlite:///sqlalchemy\_example.db')

Base.metadata.bind = engine

DBSession = sessionmaker(bind=engine)

session = DBSession()

# Insert a Person in the person table

new\_person = Person(name='new person')

session.add(new\_person)

session.commit()

现在差不多搞明白了这几种数据库驱动的优劣，接下来你就可以选择其中的一个进行系统的学习再把它应用到项目中去了，祝你学习开心，不懂的可以咨询我哈。

**总结**

以上所述是小编给大家介绍的用 Python 连接 MySQL 的几种姿势详解，希望对大家有所帮助，如果大家有任何疑问请给我留言，小编会及时回复大家的。在此也非常感谢大家对脚本之家网站的支持！