Python SMTP发送邮件 →

# Python 操作 MySQL 数据库

Python 标准数据库接口为 Python DB-API, Python DB-API为开发人员提供了数据库应用编程接口。

Python 数据库接口支持非常多的数据库,你可以选择适合你项目的数据库:

- GadFly
- mSQL
- MySQL
- PostgreSQL
- Microsoft SQL Server 2000
- Informix
- Interbase
- Oracle
- Sybase

你可以访问Python数据库接口及API查看详细的支持数据库列表。

不同的数据库你需要下载不同的DB API模块,例如你需要访问Oracle数据库和Mysql数据,你需要下载Oracle和MySQL数据库模块。

DB-API 是一个规范. 它定义了一系列必须的对象和数据库存取方式, 以便为各种各样的底层数据库系统和多种多样的数据库接口程序提供一致的访问接口。

Python的DB-API,为大多数的数据库实现了接口,使用它连接各数据库后,就可以用相同的方式操作各数据库。

Python DB-API使用流程:

- 引入 API 模块。
- 获取与数据库的连接。
- 执行SQL语句和存储过程。
- 关闭数据库连接。

# 什么是MySQLdb?

MySQLdb 是用于Python链接Mysql数据库的接口,它实现了 Python 数据库 API 规范 V2.0,基于 MySQL C API 上建立的。

# 如何安装MySQLdb?

为了用DB-API编写MySQL脚本,必须确保已经安装了MySQL。复制以下代码,并执行:

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
import MySQLdb
```

如果执行后的输出结果如下所示,意味着你没有安装 MySQLdb 模块:

```
Traceback (most recent call last):
   File "test.py", line 3, in <module>
    import MySQLdb
ImportError: No module named MySQLdb
```

安装MySQLdb,请访问 <a href="http://sourceforge.net/projects/mysql-python">http://sourceforge.net/projects/mysql-python</a>, (Linux平台可以访问: <a href="https://pypi.python.org/pypi/MySQL-python">https://pypi.python.org/pypi/MySQL-python</a>)从这里可选择适合您的平台的安装包,分为预编译的二进制文件和源代码安装包。

如果您选择二进制文件发行版本的话,安装过程基本安装提示即可完成。如果从源代码进行安装的话,则需要切换到MySQLdb发行版本的顶级目录,并键入下列命令:

```
$ gunzip MySQL-python-1.2.2.tar.gz
$ tar -xvf MySQL-python-1.2.2.tar
$ cd MySQL-python-1.2.2
$ python setup.py build
$ python setup.py install
```

注意:请确保您有root权限来安装上述模块。

# 数据库连接

连接数据库前,请先确认以下事项:

- 您已经创建了数据库 TESTDB.
- 在TESTDB数据库中您已经创建了表 EMPLOYEE
- EMPLOYEE表字段为 FIRST\_NAME, LAST\_NAME, AGE, SEX 和 INCOME。
- 连接数据库TESTDB使用的用户名为 "testuser", 密码为 "test123",你可以可以自己设定或者直接使用root用户名及其密码, Mysql数据库用户授权请使用Grant命令。
- 在你的机子上已经安装了 Python MySQLdb 模块。
- 如果您对sql语句不熟悉,可以访问我们的SQL基础教程

### 实例:

以下实例链接Mysql的TESTDB数据库:

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import MySQLdb

# 打开数据库连接
db = MySQLdb.connect("localhost", "testuser", "test123", "TESTDB", charset='utf8')

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# 使用execute方法执行SQL语句
cursor.execute("SELECT VERSION()")

# 使用 fetchone() 方法获取一条数据
data = cursor.fetchone()

print "Database version: %s " % data

# 关闭数据库连接
db.close()
```

#### 执行以上脚本输出结果如下:

Database version : 5.0.45

# 创建数据库表

如果数据库连接存在我们可以使用execute()方法来为数据库创建表,如下所示创建表EMPLOYEE:

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import MySQLdb

# 打开数据库连接
db = MySQLdb.connect("localhost", "testuser", "test123", "TESTDB", charset='utf8')

# 使用cursor()方法获取操作游标
cursor = db.cursor()

# 如果数据表已经存在使用 execute() 方法删除表。
cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")

# 创建数据表SQL语句
sql = ""CREATE TABLE EMPLOYEE(
```

```
FIRST_NAME CHAR(20) NOT NULL,

LAST_NAME CHAR(20),

AGE INT,

SEX CHAR(1),

INCOME FLOAT )"""

cursor.execute(sq1)

# 关闭数据库连接

db.close()
```

# 数据库插入操作

以下实例使用执行 SQL INSERT 语句向表 EMPLOYEE 插入记录:

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
import MySQLdb
# 打开数据库连接
db = MySQLdb.connect("localhost", "testuser", "test123", "TESTDB", charset='utf8' )
# 使用cursor()方法获取操作游标
cursor = db.cursor()
# SQL 插入语句
sql = """INSERT INTO EMPLOYEE(FIRST_NAME,
        LAST_NAME, AGE, SEX, INCOME)
        VALUES ('Mac', 'Mohan', 20, 'M', 2000)"""
try:
  # 执行sql语句
 cursor.execute(sql)
  # 提交到数据库执行
  db.commit()
except:
  # Rollback in case there is any error
  db.rollback()
# 关闭数据库连接
db.close()
```

#### 以上例子也可以写成如下形式:

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
```

```
import MySQLdb
# 打开数据库连接
db = MySQLdb.connect("localhost", "testuser", "test123", "TESTDB", charset='utf8' )
# 使用cursor()方法获取操作游标
cursor = db.cursor()
# SQL 插入语句
sql = "INSERT INTO EMPLOYEE(FIRST_NAME, \
      LAST NAME, AGE, SEX, INCOME) \
      VALUES (%s, %s, %s, %s, %s)" % \
      ('Mac', 'Mohan', 20, 'M', 2000)
try:
  # 执行sql语句
  cursor.execute(sql)
  # 提交到数据库执行
  db.commit()
except:
  # 发生错误时回滚
  db.rollback()
# 关闭数据库连接
db.close()
```

## 实例:

以下代码使用变量向SQL语句中传递参数:

# 数据库查询操作

Python查询Mysql使用 fetchone() 方法获取单条数据, 使用fetchall() 方法获取多条数据。

- fetchone(): 该方法获取下一个查询结果集。结果集是一个对象
- fetchall():接收全部的返回结果行。
- rowcount: 这是一个只读属性,并返回执行execute()方法后影响的行数。

#### 实例:

查询EMPLOYEE表中salary (工资)字段大于1000的所有数据:

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
import MySQLdb
# 打开数据库连接
db = MySQLdb.connect("localhost", "testuser", "test123", "TESTDB", charset='utf8' )
# 使用cursor()方法获取操作游标
cursor = db.cursor()
# SQL 查询语句
sql = "SELECT * FROM EMPLOYEE \
      WHERE INCOME > %s" % (1000)
try:
  # 执行SQL语句
  cursor.execute(sql)
  # 获取所有记录列表
  results = cursor.fetchall()
  for row in results:
     fname = row[0]
     lname = row[1]
     age = row[2]
     sex = row[3]
     income = row[4]
     # 打印结果
     print "fname=%s,lname=%s,age=%s,sex=%s,income=%s" % \
            (fname, lname, age, sex, income)
except:
   print "Error: unable to fecth data"
# 关闭数据库连接
db.close()
```

#### 以上脚本执行结果如下:

```
fname=Mac, lname=Mohan, age=20, sex=M, income=2000
```

## 数据库更新操作

更新操作用于更新数据表的的数据,以下实例将 EMPLOYEE 表中的 SEX 字段为 'M' 的 AGE 字段递增 1:

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
```

```
import MySQLdb
# 打开数据库连接
db = MySQLdb.connect("localhost", "testuser", "test123", "TESTDB", charset='utf8' )
# 使用cursor()方法获取操作游标
cursor = db.cursor()
# SQL 更新语句
sql = "UPDATE EMPLOYEE SET AGE = AGE + 1 WHERE SEX = '%c'" % ('M')
try:
  # 执行SQL语句
cursor.execute(sql)
  # 提交到数据库执行
db.commit()
except:
 # 发生错误时回滚
  db.rollback()
# 关闭数据库连接
db.close()
```

# 删除操作

删除操作用于删除数据表中的数据,以下实例演示了删除数据表 EMPLOYEE 中 AGE 大于 20 的所有数据:

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
import MySQLdb
# 打开数据库连接
db = MySQLdb.connect("localhost", "testuser", "test123", "TESTDB", charset='utf8' )
# 使用cursor()方法获取操作游标
cursor = db.cursor()
# SQL 删除语句
sql = "DELETE FROM EMPLOYEE WHERE AGE > %s" % (20)
try:
  # 执行SQL语句
 cursor.execute(sql)
  # 提交修改
 db.commit()
except:
  # 发生错误时回滚
```

```
db.rollback()

# 关闭连接
db.close()
```

# 执行事务

事务机制可以确保数据一致性。

事务应该具有4个属性:原子性、一致性、隔离性、持久性。这四个属性通常称为ACID特性。

- 原子性(atomicity)。一个事务是一个不可分割的工作单位,事务中包括的诸操作要么都做,要么都不做。
- 一致性(consistency)。事务必须是使数据库从一个一致性状态变到另一个一致性状态。一致性与原子性是密切相关的。
- 隔离性(isolation)。一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对并发的其他事务是隔离的,并发执行的各个事务之间不能互相干扰。
- 持久性(durability)。持续性也称永久性(permanence),指一个事务一旦提交,它对数据库中数据的改变就应该是永久性的。接下来的其他操作或故障不应该对其有任何影响。

Python DB API 2.0 的事务提供了两个方法 commit 或 rollback。

## 实例:

```
# SQL删除记录语句
sql = "DELETE FROM EMPLOYEE WHERE AGE > %s" % (20)
try:
    # 执行SQL语句
    cursor.execute(sql)
    # 向数据库提交
    db.commit()
except:
    # 发生错误时回滚
    db.rollback()
```

对于支持事务的数据库,在Python数据库编程中,当游标建立之时,就自动开始了一个隐形的数据库事务。commit()方法游标的所有更新操作,rollback()方法回滚当前游标的所有操作。每一个方法都开始了一个新的事务。

## 错误处理

DB API中定义了一些数据库操作的错误及异常,下表列出了这些错误和异常:

异常	描述
Warning	当有严重警告时触发,例如插入数据是被截断等等。必须是 StandardError 的子类。
Error	警告以外所有其他错误类。必须是 StandardError 的子类。
InterfaceError	当有数据库接口模块本身的错误(而不是数据库的错误)发生时触发。 必须是Error的子类。

DatabaseError	和数据库有关的错误发生时触发。 必须是Error的子类。
DataError	当有数据处理时的错误发生时触发,例如:除零错误,数据超范围等等。 必须是DatabaseError的子 类。
OperationalError	指非用户控制的,而是操作数据库时发生的错误。例如:连接意外断开、数据库名未找到、事务处理失败、内存分配错误等等操作数据库是发生的错误。必须是DatabaseError的子类。
IntegrityError	完整性相关的错误,例如外键检查失败等。必须是DatabaseError子类。
InternalError	数据库的内部错误,例如游标(cursor)失效了、事务同步失败等等。 必须是DatabaseError子类。
ProgrammingError	程序错误,例如数据表(table)没找到或已存在、SQL语句语法错误、参数数量错误等等。必须是DatabaseError的子类。
NotSupportedError	不支持错误,指使用了数据库不支持的函数或API等。例如在连接对象上 使用.rollback()函数,然而数据库并不支持事务或者事务已关闭。 必须是DatabaseError的子类。

◆ Python rstrip()方法

Python SMTP发送邮件 →



## 1 篇笔记





关于Mysql的连接,经过摸索,建议正文修改一下,使用mysql官方提供的连接器,我目前安装的mysql是8.0.12版本,数据库安装完成后,可以安装"mysql-connector-python-8.0.12-py2.7-windows-x86-64bit",当然了,要根据自己的操作系统和python版本以及位数进行选择,我是win10的64位,python2.7的64位,故选择的上述插件,安装完成后,直接使用以下代码进行测试:

# -\*- coding:utf-8 -\*
import mysql.connector

# 打开数据库连接(请根据自己的用户名、密码及数据库名称进行修改)
cnn = mysql.connector.connect(user='root',passwd='root',database='testdb')

# 使用cursor()方法获取操作游标
cursor = cnn.cursor()

# 使用execute方法执行SQL语句
cursor.execute("SELECT VERSION()")

# 使用 fetchone() 方法获取一条数据
data = cursor.fetchone()
print "Database version: %s " % data

# 执行sql语句

cnn.close()

显示的结果应该如下:

Database version: 8.0.12

### 显示的结果应该如下:

Database version : 8.0.12

### 相关资源下载链接如下:

Python2.7 (个人觉得这个版本语句兼容性高,适合入门):

https://blog.python.org/2018/05/python-2715-released.html

Mysql8.0.12(请根据自己需要选择版本):

https://dev.mysql.com/downloads/installer/

Mysql官方数据库连接器(请根据自己需要选择):

https://dev.mysql.com/downloads/connector/python/

备注:有些页面打开会很困难,原因不解释,如果实在打不开,那就自己搜一下吧。

fun\_zb 7个月前(08-25)