**Python学习总结笔记（9）-- MySQL数据库操作之mysql-connector**

2016年12月11日 01:36:13

阅读数：2771

MySQL数据库是使用最广泛的关系型数据库之一，其性能优良，能够承受高并发的访问，非常适合作为Web应用的后台数据库。

**0x01 安装MySQL访问驱动**

Python中操作MySQL数据库的适配器主要有两个，一是MySQLdb（即MySQL-Python）: 封装了MySQL C驱动的Python驱动器；另一个是mysql-connector：MySQL官方的纯Python驱动器。两者用法类似，但是在Python v3.x中，不再支持MySQLdb，但仍然支持mysql-connector，所以这里我们就以mysql-connector为例进行学习。

可以直接使用pip来安装mysql-connector模块：

pip install mysql-connector

* 1

**0x02 连接数据库**

#!/usr/bin/env python

#coding:utf-8

import mysql.connector

import time

\_\_author\_\_ = 'kikay'

try:

#连接数据库

con=mysql.connector.connect(host='localhost',port=3306,user='root',

password='root',database='test',charset='utf8')

print con.connection\_id

time.sleep(5)

#断开

con.close()

except mysql.connector.Error,e:

print e.message

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18

根据Python函数关键字参数的特性，可以这么改写：

#!/usr/bin/env python

#coding:utf-8

import mysql.connector

import time

\_\_author\_\_ = 'kikay'

try:

#配置信息

config={

'host':'localhost',

'port':3306,

'user':'root',

'password':'root',

'database':'test',

'charset':'utf8'

}

#连接数据库

# con=mysql.connector.connect(host='localhost',port=3306,user='root',

# password='root',database='test',charset='utf8')

con=mysql.connector.connect(\*\*config)

print con.connection\_id

time.sleep(5)

#断开

con.close()

except mysql.connector.Error,e:

print e.message

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28

数据库连接成功后，下面我们按照增删改查的顺序，依次介绍相关操作。

**0x03 插入操作**

插入数据有3种方式：

（1）通过字符串直接插入

#!/usr/bin/env python

#coding:utf-8

from mysql import connector

\_\_author\_\_ = 'kikay'

#连接

try:

#配置信息

config={

'host':'localhost',

'port':3306,

'user':'root',

'password':'root',

'database':'test',

'charset':'utf8'

}

#连接数据库

# con=mysql.connector.connect(host='localhost',port=3306,user='root',

# password='root',database='test',charset='utf8')

con=connector.connect(\*\*config)

cursor=con.cursor()

#通过字符串直接插入

insert1=("insert into user(name,age) values('Tom',20)")

cursor.execute(insert1)

#提交

con.commit()

#关闭

cursor.close()

con.close()

except connector.Error,e:

print e.message

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34

（2）通过tuple方式插入

#!/usr/bin/env python

#coding:utf-8

from mysql import connector

\_\_author\_\_ = 'kikay'

#连接

try:

#配置信息

config={

'host':'localhost',

'port':3306,

'user':'root',

'password':'root',

'database':'test',

'charset':'utf8'

}

#连接数据库

# con=mysql.connector.connect(host='localhost',port=3306,user='root',

# password='root',database='test',charset='utf8')

con=connector.connect(\*\*config)

cursor=con.cursor()

#通过tuple方式插入（利用%s作为占位符）

insert2=("insert into user(name,age) values(%s,%s)")

data=('Tom',20)

cursor.execute(insert2,data)

#提交

con.commit()

#关闭

cursor.close()

con.close()

except connector.Error,e:

print e.message

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36

（3）通过dict方式插入

#!/usr/bin/env python

#coding:utf-8

from mysql import connector

\_\_author\_\_ = 'kikay'

#连接

try:

#配置信息

config={

'host':'localhost',

'port':3306,

'user':'root',

'password':'root',

'database':'test',

'charset':'utf8'

}

#连接数据库

# con=mysql.connector.connect(host='localhost',port=3306,user='root',

# password='root',database='test',charset='utf8')

con=connector.connect(\*\*config)

cursor=con.cursor()

#通过dict方式插入（利用%(字段)s作为占位符）

insert3=("insert into user(name,age) values(%(name)s,%(age)s)")

data={

'name':'Tom',

'age':21

}

cursor.execute(insert3,data)

#提交

con.commit()

#关闭

cursor.close()

con.close()

except connector.Error,e:

print e.message

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39

（4）批量插入

此外，mysql-connector还支持批量插入：

#!/usr/bin/env python

#coding:utf-8

from mysql import connector

\_\_author\_\_ = 'kikay'

#连接

try:

#配置信息

config={

'host':'localhost',

'port':3306,

'user':'root',

'password':'root',

'database':'test',

'charset':'utf8'

}

#连接数据库

# con=mysql.connector.connect(host='localhost',port=3306,user='root',

# password='root',database='test',charset='utf8')

con=connector.connect(\*\*config)

cursor=con.cursor()

#批量插入

insertmany=("insert into user(name,age) values(%s,%s)")

data=[

('Tom1',20),

('Tom2',21),

('Tom3',22)

]

cursor.executemany(insertmany,data)

#提交

con.commit()

#关闭

cursor.close()

con.close()

except connector.Error,e:

print e.message

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40

上面的批量插入操作使用的是tuple方式，当然也可以使用dict方式，这里就不再赘述了。

**0x04 删除操作**

与插入操作的语法类似：

#!/usr/bin/env python

#coding:utf-8

\_\_author\_\_ = 'kikay'

from mysql import connector

try:

#配置信息

config={

'host':'localhost',

'port':3306,

'user':'root',

'password':'root',

'database':'test',

'charset':'utf8'

}

#连接数据库

con=connector.connect(\*\*config)

cursor=con.cursor()

#直接通过字符串方式

delete1=("delete from user where name='Tom'")

cursor.execute(delete1)

#通过tuple方式

delete2=("delete from user where name=%s and age=%s")

data=('Tom',20)

cursor.execute(delete2,data)

#通过dict方式

delete3=("delete from user where name=%(name)s and age=%(age)s")

data={

'name':'Tom',

'age':20

}

cursor.execute(delete3,data)

#提交

con.commit()

#关闭

cursor.close()

con.close

except connector.Error,e:

print e.message

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47

**0x05 更新操作**

类似的：

#!/usr/bin/env python

#coding:utf-8

from mysql import connector

\_\_author\_\_ = 'kikay'

try:

#配置信息

config={

'host':'localhost',

'port':3306,

'user':'root',

'password':'root',

'database':'test',

'charset':'utf8'

}

#连接数据库

con=connector.connect(\*\*config)

cursor=con.cursor()

#通过字符串方式直接更新

update1=("update user set name='Tom1',age=20 where Id=81")

cursor.execute(update1)

#通过tuple方式

update2=("update user set name=%s,age=%s where Id=%s")

data=('Tom2',21,81)

cursor.execute(update2,data)

#通过dict方式

update3=("update user set name=%(name)s,age=%(age)s where Id=%(Id)s")

data={

'name':'Tom3',

'age':29,

'Id':81

}

cursor.execute(update3,data)

#提交

con.commit()

#关闭

cursor.close()

con.close()

except connector.Error,e:

print e.message

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47

**0x06 查询操作**

也支持3种方式，下面以字符串方式为例：

#!/usr/bin/env python

#coding:utf-8

from mysql import connector

\_\_author\_\_ = 'kikay'

try:

#配置信息

config={

'host':'localhost',

'port':3306,

'user':'root',

'password':'root',

'database':'test',

'charset':'utf8'

}

#连接

con=connector.connect(\*\*config)

cursor=con.cursor()

#利用字符串方式查询

query1=("select Id,name,age from user where Id>10")

cursor.execute(query1)

#取出字段名称集合

columns=cursor.column\_names

#取出全部数据

result=cursor.fetchall()

print '数据表字段名称：{0}'.format(columns)

print '查询结果：{0}'.format(result)

#关闭

cursor.close()

con.close()

except connector.Error,e:

print e.message

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36

结果如下：

数据表字段名称：(u'Id', u'name', u'age')

查询结果：[(80, u'Tome', 30), (81, u'Tom3', 29), (82, u'Andy', 25)]

* 1
* 2

查询出来的字符串都是Unicode编码方式。另外，我们可以看到，查询的结果是一个list类型，其中每一项都是tuple，但是直观上不知道tuple中的每个值对应哪一个字段，为此，我们可以对cursor对象重新定义，让其输出dict:

#!/usr/bin/env python

#coding:utf-8

from mysql import connector

\_\_author\_\_ = 'kikay'

try:

#配置信息

config={

'host':'localhost',

'port':3306,

'user':'root',

'password':'root',

'database':'test',

'charset':'utf8'

}

#连接

con=connector.connect(\*\*config)

cursor=con.cursor(cursor\_class=connector.cursor.MySQLCursorDict)

#利用字符串方式查询

query1=("select Id,name,age from user where Id>10")

cursor.execute(query1)

#取出字段名称集合

columns=cursor.column\_names

#取出全部数据

result=cursor.fetchall()

print '数据表字段名称：{0}'.format(columns)

print '查询结果：{0}'.format(result)

#关闭

cursor.close()

con.close()

except connector.Error,e:

print e.message

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37

结果执行如下：

数据表字段名称：(u'Id', u'name', u'age')

查询结果：[{u'age': 30, u'Id': 80, u'name': u'Tome'}, {u'age': 29, u'Id': 81, u'name': u'Tom3'}, {u'age': 25, u'Id': 82, u'name': u'Andy'}]

* 1
* 2

两者的区别在于我们显示定义：

cursor\_class=connector.cursor.MySQLCursorDict

* 1

**0x07 后记**

此外，我们特别要注意中文字符的问题，比如需要插入含有中文的字符串，或者表名、字段名是中文等等，这里给出一个例子作为参考：

# /usr/bin/env python

# coding:utf-8

import mysql.connector

# 配置信息

config = {

'host': 'localhost',

'port': 3306,

'user': 'root',

'password': 'root',

'database': 'test',

'charset': 'utf8'

}

conn = mysql.connector.connect(\*\*config)

cursor = conn.cursor(cursor\_class=mysql.connector.cursor.MySQLCursorDict)

#插入数据

data = [

('Tom', '2010-01-01 00:00:00', '牛逼'),

('kikay', '2010-11-11 10:33:33', '相当牛逼'),

]

cursor.executemany(u'insert into member(name,bir,简介) values(%s,%s,%s)', data)

conn.commit()

# 查询

cursor.execute('select \* from member')

rows = cursor.rowcount

columns = cursor.column\_names

data = cursor.fetchall()

# data=cursor.fetchone()

# 处理含中文的字段

if data and len(data) > 0:

print data[0][u'简介']

cursor.close()

conn.close()

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39

上面简单总结了Python基于mysql-connector模块进行增删改查等相关操作，MySQLdb的操作过程是类似的。另外，在实际应用中，我们可能采用更高层的ORM方式，比如SQLAlchemy、SQLObject等等，在以后的博客中将会介绍。