工作记录11之python操作数据库

2016-09-18 14:57:26

python 对数据库的操作是通过MySQLdb模块，MySQLdb is an thread-compatible interface to the popular MySQL database server that provides the Python database API. 使用这种数据库接口的方法一般是执行连接数据库->执行query->提取数据->关闭连接这几个步骤，MySQLdb提供比较关键的对象，分别是Connection、Cursor、Result。 在MySQLdb.Connect(host , user , passw , db)函数中，经常使用的是这几个参数，但是其实还有很多重要参数如字符集、线程安全、ssl等，使用时要注意。 python操作数据库实例： 1.获得mysql版本： # -*- coding: UTF-8 -*- #安装 MYSQL DB for python import MySQLdb as mdb con = None try: #连接 mysql 的方法： connect('ip','user','password','dbname') con = mdb.connect('localhost', 'root','root', 'test'); #所有的查询，都在连接 con 的一个模块 cursor 上面运行的 cur = con.cursor() #执行一个查询 cur.execute("SELECT VERSION()") #取得上个查询的结果，是单个结果 data = cur.fetchone() print "Database version : %s " % data finally: if con: #无论如何，连接记得关闭 con.close() 2.创建表并插入数据 import MySQLdb as mdb import sys #将 con 设定为全局连接 con = mdb.connect('localhost', 'root', 'root', 'test'); with con: #获取连接的 cursor，只有获取了 cursor，我们才能进行各种操作 cur = con.cursor() #创建一个数据表 writers(id,name) cur.execute("CREATE TABLE IF NOT EXISTS \ Writers(Id INT PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(25))") #以下插入了 5 条数据 cur.execute("INSERT INTO Writers(Name) VALUES('Jack London')") cur.execute("INSERT INTO Writers(Name) VALUES('Honore de Balzac')") cur.execute("INSERT INTO Writers(Name) VALUES('Lion Feuchtwanger')") cur.execute("INSERT INTO Writers(Name) VALUES('Emile Zola')") cur.execute("INSERT INTO Writers(Name) VALUES('Truman Capote')") 3.使用select获取数据并遍历 import MySQLdb as mdb import sys #连接 mysql，获取连接的对象 con = mdb.connect('localhost', 'root', 'root', 'test'); with con: #仍然是，第一步要获取连接的 cursor 对象，用于执行查询 cur = con.cursor() #类似于其他语言的 query 函数， execute 是 python 中的执行查询函数 cur.execute("SELECT * FROM Writers") #使用 fetchall 函数，将结果集（多维元组）存入 rows 里面 rows = cur.fetchall() #依次遍历结果集，发现每个元素，就是表中的一条记录，用一个元组来显示 for row in rows: print row 运行结果为： 1 Jack London 2 Honore de Balzac 3 Lion Feuchtwanger 4 Emile Zola 5 Truman Capote 4.使用cursor取得结果集 import MySQLdb as mdb import sys #获得 mysql 查询的链接对象 con = mdb.connect('localhost', 'root', 'root', 'test') with con: #获取连接上的字典 cursor，注意获取的方法， #每一个 cursor 其实都是 cursor 的子类 cur = con.cursor(mdb.cursors.DictCursor) #执行语句不变 cur.execute("SELECT * FROM Writers") #获取数据方法不变 rows = cur.fetchall() #遍历数据也不变（比上一个更直接一点） for row in rows: #这里，可以使用键值对的方法，由键名字来获取数据 print "%s %s" % (row["Id"], row["Name"]) 5.获取单个表的字段信息 import MySQLdb as mdb import sys #获取数据库的链接对象 con = mdb.connect('localhost', 'root', 'root', 'test') with con: #获取普通的查询 cursor cur = con.cursor() cur.execute("SELECT * FROM Writers") rows = cur.fetchall() #获取连接对象的描述信息 desc = cur.description print 'cur.description:',desc #打印表头，就是字段名字 print "%s %3s" % (desc[0][0], desc[1][0]) for row in rows: #打印结果 print "%2s %3s" % row 运行结果： cur.description: ((‘Id', 3, 1, 11, 11, 0, 0), (‘Name', 253, 17, 25, 25, 0, 1)) Id Name 1 Jack London 2 Honore de Balzac 3 Lion Feuchtwanger 4 Emile Zola 5 Truman Capote 6.使用prepared statement执行查询 import MySQLdb as mdb import sys con = mdb.connect('localhost', 'root', 'root', 'test') with con: cur = con.cursor() #我们看到，这里可以通过写一个可以组装的 sql 语句来进行 cur.execute("UPDATE Writers SET Name = %s WHERE Id = %s", ("Guy de Maupasant", "4")) #使用 cur.rowcount 获取影响了多少行 print "Number of rows updated: %d" % cur.rowcount 7.使用事物 import MySQLdb as mdb import sys try: #连接 mysql，获取连接的对象 conn = mdb.connect('localhost', 'root', 'root', 'test'); cursor = conn.cursor() #如果某个数据库支持事务，会自动开启 #这里用的是 MYSQL，所以会自动开启事务（若是 MYISM 引擎则不会） cursor.execute("UPDATE Writers SET Name = %s WHERE Id = %s", ("Leo Tolstoy", "1")) cursor.execute("UPDATE Writers SET Name = %s WHERE Id = %s", ("Boris Pasternak", "2")) cursor.execute("UPDATE Writer SET Name = %s WHERE Id = %s", ("Leonid Leonov", "3")) #事务的特性 1、原子性的手动提交 conn.commit() cursor.close() conn.close() except mdb.Error, e: #如果出现了错误，那么可以回滚，就是上面的三条语句要么执行，要么都不执行 conn.rollback() print "Error %d: %s" % (e.args[0],e.args[1]) 8.数据库图片转换为二进制的存取 CREATE TABLE Images(Id INT PRIMARY KEY AUTO_INCREMENT, Data MEDIUMBLOB); 存入： import MySQLdb as mdb import sys try: #用读文件模式打开图片 fin = open("../web.jpg") #将文本读入 img 对象中 img = fin.read() #关闭文件 fin.close() except IOError, e: #如果出错，打印错误信息 print "Error %d: %s" % (e.args[0],e.args[1]) sys.exit(1) try: #链接 mysql，获取对象 conn = mdb.connect(host='localhost',user='root',passwd='root', db='test') #获取执行 cursor cursor = conn.cursor() #直接将数据作为字符串，插入数据库 cursor.execute("INSERT INTO Images SET Data='%s'" % mdb.escape_string(img)) #提交数据 conn.commit() #提交之后，再关闭 cursor 和链接 cursor.close() conn.close() except mdb.Error, e: #若出现异常，打印信息 print "Error %d: %s" % (e.args[0],e.args[1]) sys.exit(1) 读取： import MySQLdb as mdb import sys try: #连接 mysql，获取连接的对象 conn = mdb.connect('localhost', 'root', 'root', 'test'); cursor = conn.cursor() #执行查询该图片字段的 SQL cursor.execute("SELECT Data FROM Images LIMIT 1") #使用二进制写文件的方法，打开一个图片文件，若不存在则自动创建 fout = open('image.png','wb') #直接将数据如文件 fout.write(cursor.fetchone()[0]) #关闭写入的文件 fout.close() #释放查询数据的资源 cursor.close() conn.close() except IOError, e: #捕获 IO 的异常， 主要是文件写入会发生错误 print "Error %d: %s" % (e.args[0],e.args[1]) sys.exit(1) python操作数据库时的错误处理有： 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。