数据库编程

mariadb安装与使用

```
[root@server0 ~]# yum  install mariadb-server -y

[root@server0 ~]# systemctl  start mariadb

[root@server0 ~]# mysql_secure_installation

[root@server0 ~]# mysql -uroot -pwestos
```

数据库基本操作

```
[root@server0 ~]# mysql -uroot -pwestos
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 12

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)

MariaDB [(none)]> create database westos;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> use westos;
Database changed

MariaDB [westos]> show tables;
Empty set (0.00 sec)

MariaDB [westos]> create table userinfo( username varchar(10), password varchar(10));
```

```
Query OK, 0 rows affected (0.16 sec)

MariaDB [westos]> desc userinfo;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| username | varchar(10) | YES  |     | NULL    |       |
| password | varchar(10) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

MariaDB [westos]> insert into userinfo values("user1", "123");
Query OK, 1 row affected (0.02 sec)

MariaDB [westos]> select * from userinfo;
+----------+----------+
| username | password |
+----------+----------+
| user1    | 123      |
+----------+----------+
1 row in set (0.00 sec)

MariaDB [westos]> insert into userinfo values("程序员1", "123");
Query OK, 1 row affected, 1 warning (0.02 sec)

MariaDB [westos]> select * from userinfo;
+----------+----------+
| username | password |
+----------+----------+
| user1    | 123      |
| ???1     | 123      |
+----------+----------+
2 rows in set (0.00 sec)

MariaDB [westos]> update userinfo set password="345" where username
="user1";
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [westos]> select * from userinfo;+----------+----------+
| username | password |
+----------+----------+
| user1    | 345      |
| ???1     | 123      |
+----------+----------+
2 rows in set (0.00 sec)
```

```
MariaDB [westos]> delete from userinfo where username="user1";
Query OK, 1 row affected (0.06 sec)

MariaDB [westos]> select * from userinfo;
+----------+----------+
| username | password |
+----------+----------+
| ???1     | 123      |
+----------+----------+
1 row in set (0.00 sec)

MariaDB [westos]> drop table userinfo;
Query OK, 0 rows affected (0.04 sec)

MariaDB [westos]> drop database westos;
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.01 sec)

MariaDB [(none)]> quit
```

mariadb中文编码问题

当使用mysql数据库时，作为纯正的爱国者(实质是英语看不懂)，当然会使用中文存储一些数据，但是默认情况下存储的中文是 `??????` 这种外星文文字，这是要搞什么?

因此如何修改数据库中的编码格式设置呢?

数据库编码格式的查看

- 登陆数据库

  ```
  $ mysql -uroot -p
  ```

- 通过下面命令查看mysql的编码格式;

```
MariaDB [(none)]> show variables like 'character_set_%';
+--------------------------+----------------------------+
| Variable_name            | Value                      |
+--------------------------+----------------------------+
| character_set_client     | utf8                       |
| character_set_connection | utf8                       |
| character_set_database   | latin1                     |
| character_set_filesystem | binary                     |
| character_set_results    | utf8                       |
| character_set_server     | latin1                     |
| character_set_system     | utf8                       |
| character_sets_dir       | /usr/share/mysql/charsets/ |
+--------------------------+----------------------------+
```

临时修改数据库的编码格式

- 修改除了character_set_filesystem之外的其他变量的编码格式为utf8;

```
MariaDB [(none)]> set character_set_server='utf8';
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]> set character_set_database = 'utf8';
Query OK, 0 rows affected (0.00 sec)
```

- 但是，很坑的是，当重启服务后，刚才的设置就直接失效，也是挺崩溃的;大招在下面.

永久修改数据库的编码格式

- 配置mysql的配置文件，配置文件在/etc/my.cnf.d/目录，设置编码格式：
- 第一个文件修改： /etc/my.cnf.d/client.cnf

```
[client]
    default-character-set=utf8
```

- 第二个文件修改： /etc/my.cnf.d/server.cnf

```
# this is only for the mysqld standalone daemon
[mysqld]
    character-set-server=utf8
```

- 上面设置完成，看似很完美，但是重启mariadb服务后，character_set_database并没有进行修改，因此，为了一步到位，直接设置character_set_database编码格式为utf8，如下:

```
MariaDB [(none)]> set character_set_database = 'utf8';
Query OK, 0 rows affected (0.00 sec)
```

一切ok，终于搞定!

多条语句执行

```
In [2]: import MySQLdb as mysql

In [4]: conn = mysql.connect(user='root', passwd='westos', db='user
s', charset='utf8')

In [5]: cur = conn.cursor()


# 方法一: 多次执行，但效率低；

In [6]: cur.execute('insert into userinfo values("user1", "123");')
Out[6]: 1L

In [7]: cur.execute('insert into userinfo values("user2", "123");')
Out[7]: 1L


# 方法二: 选择传入变量， 然后for循环执行;

In [11]: sqli = 'insert into userinfo values("%s", "%s");' %("user3",
 "123")

In [12]: cur.execute(sqli)
Out[12]: 1L

# 方法三: executemany方法， 直接传入列表， 实现数据多行添加；

In [13]: li = [("user1", "wdwef"), ("user2", "124213"), ("user3", "de
fergf")]

In [14]: sqli = 'insert into userinfo values("%s", "%s");'

In [15]: for i in li:
    ...:     cur.execute(sqli, i)
    ...:

In [17]: cur.executemany(sqli, li)
Out[17]: 3L
```

查看表内容

```
# 1. fetchone()查看一条信息， 指针向后移动;
In [26]: cur.execute('select * from userinfo where username="user
1";')
Out[26]: 5L

In [27]: cur.fetchone()
Out[27]: (u'user1', u'123')

In [28]: cur.fetchone()
Out[28]: (u'user1', u'123')

In [29]: cur.fetchone()
Out[29]: (u'user1', u'123')

In [30]: cur.fetchone()
Out[30]: (u'user1', u'123')

In [31]: cur.fetchone()
Out[31]: (u'user1', u'123')

In [32]: cur.fetchone()



# 2. scroll()移动指针;
In [41]: cur.scroll(0, 'absolute')

In [42]: cur.fetchone()
Out[42]: (u'user1', u'123')



# 3. fetchmany(n)显示n条信息
In [49]: cur.fetchmany(3)
Out[49]:
((u'user1', u'123'),
 (u'user1', u'123'),
 (u'user1', u'123'))

# 4. fetchall()显示所有信息
In [49]: cur.fetchall()
Out[49]:
```

```
((u'user1', u'123'),
 (u'user1', u'123'),
 (u'user1', u'123'),
 (u'user1', u'123'),
 (u'user1', u'123'))
```

数据库应用

- 随机生成100个人名和对应的密码;
- 人名由三个汉字或者2个汉字组成，
    - 姓 = [许, 张， 赵， 钱， 孙， 李， 朱， 杨]
    - 名 = [彬， 群， 宁， 盼， 龙， 欢， 丹]
- 密码统一6位， 由字母和字符组成;

- 存储上述用户信息到数据库中， 保存在数据库users中的userinfo表中;

In [ ]:

```
#!/usr/bin/env pyhton
# coding:utf-8

"""
@Name: mysql01.py
@Author: lvah
@Date:1/10/18
@Connect: xc_guofan@163.com
@Desc:




"""

import random
from random import choice as choice
import string
import MySQLdb as mysql


# 生成指定位数密码， 前n-1位为数字， 最后一位为密码;
def create_passwd(count=6):
    nums = random.sample(string.digits, count - 1)
    letters = random.sample(string.letters, 1)
    return "".join(nums + letters)

# 生成随机的姓名， 有两个或三个汉字组成;
def create_name():
    first = ['许', '张', '赵', '钱', '孙', '李', '朱', '杨']
    second = ['彬', '群', '宁', '盼', '龙', '欢', '丹']
    last = ['彬', '群', '宁', '盼', '龙', '欢', '丹', ' ' ]
    name = choice(first) + choice(second)+ choice(last)
    return  name.rstrip()

def main():

    # 连接数据库
    conn = mysql.connect(user='root', passwd='westos', charset='utf8')
    cur = conn.cursor()
    conn.select_db('users')

    n = input("生成数据数:")

    # 往数据库表中插入n条随机数据;
    for i in range(n):
        # cur.execute('insert into userinfo values("user1", "123");')
        sqli = 'insert into userinfo values ("%s", "%s");' %(create_name(), c
        cur.execute(sqli)

    # 提交数据，并关闭连接;
    conn.commit()
    cur.close()
    cur.close()
    conn.close()

main()
```

应用之判断某个数据库表是否存在

In [ ]:

```
#coding:utf-8



import MySQLdb as mysql

def is_table_exist(db_name, table_name):
    conn  = mysql.connect(user="root", passwd="westos",db=db_name, charset='ut
    cur = conn.cursor()


    # 查看这个数据库的所有表名， 判断
    cur.execute('show tables;')

    # ((u'userinfo',), (u'userinfo1',))
    # print cur.fetchall()

    # i= (u'userinfo',)  (u'userinfo1',)
    for i  in cur.fetchall():
        if table_name ==  i[0]:
            return True
        else:
            return False

print is_table_exist("users", "userinfo3")
```

应用之封装MySQLdb

- 封装一个数据库类为MyMySQL；
- 实例化对象时， 自动连接数据库， 自动创建游标；
- 删除实例化对象时， 自动关闭游标， 断开数据库连接;
- 编写select_table(self, table_name, num)方法， 显示该表里面的n条数据;
- 编写create_table(self, table_name, `**attr`)方法， 实现创建表， 创建前判断表是否存在， **attr是表的属性名；

In [ ]:

```
#!/usr/bin/env pyhton
# coding:utf-8

"""
@Name: mysql01.py
@Author: lvah
@Date:1/10/18
@Connect: xc_guofan@163.com
@Desc:




"""

import MySQLdb as mysql

DB_CONFIG = {
    'name': 'root',
    'passwd': 'westos',
    'db': 'users',
    'host': 'localhost',
    'port': 3306,
    'charset': 'utf8',
}


class MyMySQL(object):
    # 构造函数， 实例化对象时，自动生成
    def __init__(self, **db_config):
        self.name = db_config['name']
        self.__passwd = db_config['passwd']
        self.db = db_config['db']
        self.host = db_config['host']
        self.port = db_config['port']
        self.charset = db_config['charset']

        # 连接数据库
        self.conn = mysql.connect(
            user=self.name, passwd=self.__passwd,
            db=db_config['db'], host=db_config['host'],
            port=db_config['port'], charset=db_config['charset']
        )

        # 创建游标
        self.cur = self.conn.cursor()

        print "数据库连接成功........."

    def is_table_exist(self, table_name, db_name=None):

        # 如果用户指定数据库名称， 先跳转到指定数据库， 再判断;
        if db_name != None:
```

```
        if db_name != None:
            self.conn.select_db(db_name)
            # print db_name

        # 查看这个数据库的所有表名， 判断
        self.cur.execute('show tables;')

        # i= (u'userinfo',)  (u'userinfo1',)
        for i in self.cur.fetchall():
            if table_name == i[0]:
                return True
        else:
            return False

    def select_table(self, table_name, num=None):
        if self.is_table_exist(table_name):
            self.cur.execute('select * from %s;' % table_name)
            if num == None:
                return self.cur.fetchall()
            else:
                return self.cur.fetchmany(num)
        else:
            print "%s 不存在" % (table_name)



    # attr  = {
    #   "name" : "varchar(20)",
    #    "age" : "int"
    # }


    def create_table(self, table_name, **attr):
        if self.is_table_exist(table_name):
            return  "%s 已经存在" %(table_name)
        else:
            # 生成value里面需要的字符串;
            s = ''
            for key, value in attr.items():
                # s = 'name varchar(20),'
                # s = 'name varchar(20),age int,'
                s = s + key + " " + value +","
            s = s.rstrip(",")

            sqli = 'create table %s (%s);' %(table_name, s)
            # print sqli
            self.cur.execute(sqli)
            print "%s 创建成功" %(table_name)


            # self.cur.execute('create table userinfo values(name varchar(20),

    def __del__(self):
        self.conn.commit()
        self.cur.close()
        self.conn.close()

        print "数据库断开成功.....欢迎下次连接"


a = MyMySQL(**DB_CONFIG)

print a.create_table('good', id = 'varchar(10)', age = "int")
#
# print a.is_table_exist('userinfo')

# print a.select_table('userinfo', 10)
```