

公告

wiki和教程：[www.pythonav.com](http://www.pythonav.com)

免费教学视频：[B站](#)：[凸头统治地球](#)

高级专题教程：[网易云课堂](#)：[武沛齐](#)

聊技术，加武Sir微信



武沛齐



扫一扫上面的二维码图案，加我微信



Python技术交流群：737658057

软件测试开发交流群：721023555

昵称：武沛齐  
园龄：8年1个月  
粉丝：9974  
关注：44  
[+加关注](#)

我的标签

- Python(17)
- ASP.NET MVC(15)
- python之路(7)
- Tornado源码分析(5)
- 每天一道Python面试题(5)
- crm项目(4)
- 面试都在问什么？(2)
- Python开源组件 - Tyrion(1)
- Python面试315题(1)
- Python企业面试题讲解(1)

积分与排名

积分 - 426642  
排名 - 785

随笔分类

- JavaScript(1)
- MVC(15)
- Python(17)
- 面试都在问什么系列？【图】(2)
- 其他(37)

随笔 - 140 文章 - 164 评论 - 893

Python开发【第十九篇】：Python操作MySQL

本篇对于Python操作MySQL主要使用两种方式：

- 原生模块 pymysql
- ORM框架 SQLAlchemy

pymysql

pymysql是Python中操作MySQL的模块，其使用方法和MySQLdb几乎相同。

下载安装

```
1 | pip3 install pymysql
```

使用操作

1、执行SQL

```
1 | #!/usr/bin/env python
2 | # -*- coding:utf-8 -*-
3 | import pymysql
4 |
5 | # 创建连接
6 | conn = pymysql.connect(host='127.0.0.1', port=3306, user='root', passwd='12
7 | # 创建游标
8 | cursor = conn.cursor()
9 |
10 | # 执行SQL，并返回回收影响行数
11 | effect_row = cursor.execute("update hosts set host = '1.1.1.2'")
12 |
13 | # 执行SQL，并返回受影响行数
14 | #effect_row = cursor.execute("update hosts set host = '1.1.1.2' where nid >
15 |
16 | # 执行SQL，并返回受影响行数
17 | #effect_row = cursor.executemany("insert into hosts(host,color_id)values(%s
18 |
19 |
20 | # 提交，不然无法保存新建或者修改的数据
21 | conn.commit()
22 |
23 | # 关闭游标
24 | cursor.close()
25 | # 关闭连接
26 | conn.close()
```

2、获取新创建数据自增ID

```
1 | #!/usr/bin/env python
2 | # -*- coding:utf-8 -*-
3 | import pymysql
4 |
5 | conn = pymysql.connect(host='127.0.0.1', port=3306, user='root', passwd='12
6 | cursor = conn.cursor()
7 | cursor.executemany("insert into hosts(host,color_id)values(%s,%s)", [(1.1.
8 | conn.commit()
9 | cursor.close()
10 | conn.close()
11 |
12 | # 获取最新自增ID
```

企业面试题及答案(1)

请求响应(6)

设计模式(9)

微软C#(34)

**随笔档案**

2020年6月(1)

2020年5月(1)

2019年11月(1)

2019年10月(1)

2019年9月(4)

2018年12月(1)

2018年8月(1)

2018年5月(2)

2018年4月(1)

2017年8月(1)

2017年5月(1)

2017年3月(1)

2016年10月(1)

2016年7月(1)

2015年10月(1)

2015年8月(1)

2015年7月(1)

2015年6月(2)

2015年4月(2)

2014年3月(3)

2014年1月(3)

2013年12月(2)

2013年11月(2)

2013年10月(7)

2013年8月(17)

2013年7月(1)

2013年6月(14)

2013年5月(23)

2013年4月(3)

2013年3月(13)

2013年2月(1)

2012年11月(26)

**相册**

git(14)

**最新评论**

1. Re:Python开发【第十九篇】：Python操作MySQL

执行本网页中第一段SQL语句，其中执行语句改成： cursor.execute('insert into part(caption) values("AB") ') 报错 unable to reso...

--serene1979

2. Re:python 面向对象 (进阶篇)

class Foo:

pass

这个类不是由object.new(cls)创建的吗

13 new\_id = cursor.lastrowid

## 3、获取查询数据

```

1  #!/usr/bin/env python
2  # -*- coding:utf-8 -*-
3  import pymysql
4
5  conn = pymysql.connect(host='127.0.0.1', port=3306, user='root', passwd='12
6  cursor = conn.cursor()
7  cursor.execute("select * from hosts")
8
9  # 获取第一行数据
10 row_1 = cursor.fetchone()
11
12 # 获取前n行数据
13 # row_2 = cursor.fetchmany(3)
14 # 获取所有数据
15 # row_3 = cursor.fetchall()
16
17 conn.commit()
18 cursor.close()
19 conn.close()

```

注：在fetch数据时按照顺序进行，可以使用cursor.scroll(num,mode)来移动光标位置，如：

- cursor.scroll(1,mode='relative') # 相对当前位置移动
- cursor.scroll(2,mode='absolute') # 相对绝对位置移动

## 4、fetch数据类型

关于默认获取的数据是元组类型，如果想要或者字典类型的数据，即：

```

1  #!/usr/bin/env python
2  # -*- coding:utf-8 -*-
3  import pymysql
4
5  conn = pymysql.connect(host='127.0.0.1', port=3306, user='root', passwd='12
6
7  # 游标设置为字典类型
8  cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)
9  r = cursor.execute("call p1()")
10
11 result = cursor.fetchone()
12
13 conn.commit()
14 cursor.close()
15 conn.close()

```



作业：

参考表结构：

用户类型

用户信息

权限

用户类型@权限

功能：

# 登陆、注册、找回密码

# 用户管理

--bajie\_new

### 3. Re:MySQL练习题参考答案

14、查询和“002”号的同学学习的课程完全相同的其他同学学号和姓名； select sid,sname from (select sid,sname from student where sid i...

--BobAyIn

### 4. Re:MySQL练习题参考答案

13、查询至少学过学号为“001”同学所有课的其他同学学号和姓名； 需要考虑这种情况，001同学选了1、2、4，002同学选了1、2、3。这种情况，两位同学选课有交集，count(选课数)也一样，然是...

--BobAyIn

5. Re:人生没有白走的路，每一步都算数 后续来了吗

--小鱼仔。

```
# 用户类型
# 权限管理
# 分配权限
```

特别的：程序仅一个可执行文件

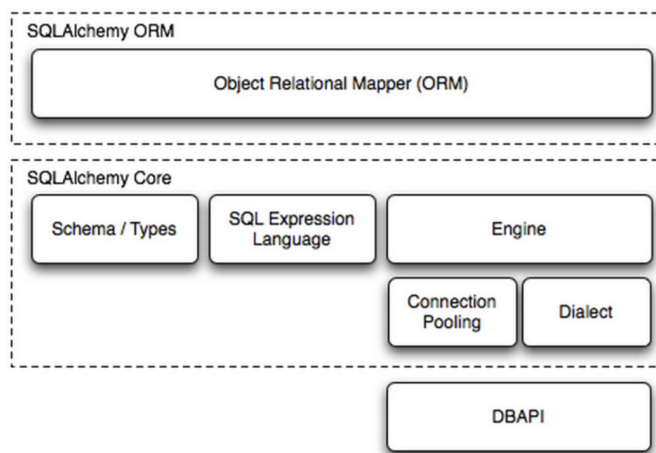


## SQLAlchemy

SQLAlchemy是Python编程语言下的一款ORM框架，该框架建立在数据库API之上，使用关系对象映射进行数据库操作，简言之便是：将对象转换成SQL，然后使用数据API执行SQL并获取执行结果。

安装：

```
1 | pip3 install SQLAlchemy
```



SQLAlchemy本身无法操作数据库，其必须以来pymysql等第三方插件，Dialect用于和数据API进行交流，根据配置文件的不同调用不同的数据库API，从而实现对数据库的操作，如：

```
1 MySQL-Python
2     mysql+mysqldb://<user>:<password>@<host>[:<port>]/<dbname>
3
4 pymysql
5     mysql+pymysql://<username>:<password>@<host>/<dbname>[?<options>]
6
7 MySQL-Connector
8     mysql+mysqlconnector://<user>:<password>@<host>[:<port>]/<dbname>
9
10 cx_Oracle
11     oracle+cx_oracle://<user>:pass@<host>:port/<dbname>[?key=value&key=value...]
12
13 更多详见: http://docs.sqlalchemy.org/en/latest/dialects/index.html
```

## 一、内部处理

使用 Engine/ConnectionPooling/Dialect 进行数据库操作，Engine使用 ConnectionPooling连接数据库，然后再通过Dialect执行SQL语句。

```
1 #!/usr/bin/env python
2 # -*- coding:utf-8 -*-
3 from sqlalchemy import create_engine
4
5
6 engine = create_engine("mysql+pymysql://root:123@127.0.0.1:3306/t1", max_ov
7
8 # 执行SQL
9 # cur = engine.execute(
```

```

10 # "INSERT INTO hosts (host, color_id) VALUES ('1.1.1.22', 3)"
11 # )
12
13 # 新插入行自增ID
14 # cur.lastrowid
15
16 # 执行SQL
17 # cur = engine.execute(
18 # "INSERT INTO hosts (host, color_id) VALUES(%, %s)", [('1.1.1.22', 3),
19 # )
20
21
22 # 执行SQL
23 # cur = engine.execute(
24 # "INSERT INTO hosts (host, color_id) VALUES (%(host)s, %(color_id)s)",
25 # host='1.1.1.99', color_id=3
26 # )
27
28 # 执行SQL
29 # cur = engine.execute('select * from hosts')
30 # 获取第一行数据
31 # cur.fetchone()
32 # 获取第n行数据
33 # cur.fetchmany(3)
34 # 获取所有数据
35 # cur.fetchall()

```

## 二、ORM功能使用

使用 ORM/Schema Type/SQL Expression

Language/Engine/ConnectionPooling/Dialect 所有组件对数据进行操作。根据类创建对象，对象转换成SQL，执行SQL。

### 1、创建表

```

1 #!/usr/bin/env python
2 # -*- coding:utf-8 -*-
3 from sqlalchemy.ext.declarative import declarative_base
4 from sqlalchemy import Column, Integer, String, ForeignKey, UniqueConstraint
5 from sqlalchemy.orm import sessionmaker, relationship
6 from sqlalchemy import create_engine
7
8 engine = create_engine("mysql+pymysql://root:123@127.0.0.1:3306/t1", max_ov
9
10 Base = declarative_base()
11
12 # 创建单表
13 class Users(Base):
14     __tablename__ = 'users'
15     id = Column(Integer, primary_key=True)
16     name = Column(String(32))
17     extra = Column(String(16))
18
19     __table_args__ = (
20         UniqueConstraint('id', 'name', name='uix_id_name'),
21         Index('ix_id_name', 'name', 'extra'),
22     )
23
24
25 # 一对多
26 class Favor(Base):
27     __tablename__ = 'favor'
28     nid = Column(Integer, primary_key=True)
29     caption = Column(String(50), default='red', unique=True)
30
31

```

```

32 class Person(Base):
33     __tablename__ = 'person'
34     nid = Column(Integer, primary_key=True)
35     name = Column(String(32), index=True, nullable=True)
36     favor_id = Column(Integer, ForeignKey("favor.nid"))
37
38
39 # 多对多
40 class Group(Base):
41     __tablename__ = 'group'
42     id = Column(Integer, primary_key=True)
43     name = Column(String(64), unique=True, nullable=False)
44     port = Column(Integer, default=22)
45
46
47 class Server(Base):
48     __tablename__ = 'server'
49
50     id = Column(Integer, primary_key=True, autoincrement=True)
51     hostname = Column(String(64), unique=True, nullable=False)
52
53
54 class ServerToGroup(Base):
55     __tablename__ = 'servertogroup'
56     nid = Column(Integer, primary_key=True, autoincrement=True)
57     server_id = Column(Integer, ForeignKey('server.id'))
58     group_id = Column(Integer, ForeignKey('group.id'))
59
60
61 def init_db():
62     Base.metadata.create_all(engine)
63
64
65 def drop_db():
66     Base.metadata.drop_all(engine)

```

注：设置外检的另一种方式 `ForeignKeyConstraint(['other_id'], ['othertable.other_id'])`

## 2、操作表

```

#!/usr/bin/env python
# -*- coding:utf-8 -*-

from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import Column, Integer, String, ForeignKey, UniqueConstraint
from sqlalchemy.orm import sessionmaker, relationship
from sqlalchemy import create_engine

engine = create_engine("mysql+pymysql://root:123@127.0.0.1:3306/t1", max_ove

Base = declarative_base()

# 创建单表
class Users(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    name = Column(String(32))
    extra = Column(String(16))

    __table_args__ = (
        UniqueConstraint('id', 'name', name='uix_id_name'),
        Index('ix_id_name', 'name', 'extra'),
    )

```

```

def __repr__(self):
    return "%s-%s" %(self.id, self.name)

# 一对多
class Favor(Base):
    __tablename__ = 'favor'
    nid = Column(Integer, primary_key=True)
    caption = Column(String(50), default='red', unique=True)

    def __repr__(self):
        return "%s-%s" %(self.nid, self.caption)

class Person(Base):
    __tablename__ = 'person'
    nid = Column(Integer, primary_key=True)
    name = Column(String(32), index=True, nullable=True)
    favor_id = Column(Integer, ForeignKey("favor.nid"))
    # 与生成表结构无关, 仅用于查询方便
    favor = relationship("Favor", backref='pers')

# 多对多
class ServerToGroup(Base):
    __tablename__ = 'servertogroup'
    nid = Column(Integer, primary_key=True, autoincrement=True)
    server_id = Column(Integer, ForeignKey('server.id'))
    group_id = Column(Integer, ForeignKey('group.id'))
    group = relationship("Group", backref='s2g')
    server = relationship("Server", backref='s2g')

class Group(Base):
    __tablename__ = 'group'
    id = Column(Integer, primary_key=True)
    name = Column(String(64), unique=True, nullable=False)
    port = Column(Integer, default=22)
    # group = relationship('Group', secondary=ServerToGroup, backref='host_list')

class Server(Base):
    __tablename__ = 'server'

    id = Column(Integer, primary_key=True, autoincrement=True)
    hostname = Column(String(64), unique=True, nullable=False)

def init_db():
    Base.metadata.create_all(engine)

def drop_db():
    Base.metadata.drop_all(engine)

Session = sessionmaker(bind=engine)
session = Session()

```

#### • 增



```

obj = Users(name="alex0", extra='sb')
session.add(obj)

```

```

session.add_all([
    Users(name="alex1", extra='sb'),
    Users(name="alex2", extra='sb'),
])
session.commit()

```



#### • 删

```

session.query(Users).filter(Users.id > 2).delete()
session.commit()

```

#### • 改

```

session.query(Users).filter(Users.id > 2).update({"name" : "099"})
session.query(Users).filter(Users.id > 2).update({Users.name: Users.name})
session.query(Users).filter(Users.id > 2).update({"num": Users.num + 1},
session.commit()

```



#### • 查

```

ret = session.query(Users).all()
ret = session.query(Users.name, Users.extra).all()
ret = session.query(Users).filter_by(name='alex').all()
ret = session.query(Users).filter_by(name='alex').first()

ret = session.query(Users).filter(text("id<:value and name=:name")).para
ret = session.query(Users).from_statement(text("SELECT * FROM users wher

```



#### • 其他

```

# 条件
ret = session.query(Users).filter_by(name='alex').all()
ret = session.query(Users).filter(Users.id > 1, Users.name == 'eric').al
ret = session.query(Users).filter(Users.id.between(1, 3), Users.name ==
ret = session.query(Users).filter(Users.id.in_([1,3,4])).all()
ret = session.query(Users).filter(~Users.id.in_([1,3,4])).all()
ret = session.query(Users).filter(Users.id.in_(session.query(Users.id).f
from sqlalchemy import and_, or_
ret = session.query(Users).filter(and_(Users.id > 3, Users.name == 'eric'
ret = session.query(Users).filter(or_(Users.id < 2, Users.name == 'eric'
ret = session.query(Users).filter(
    or_(
        Users.id < 2,
        and_(Users.name == 'eric', Users.id > 3),
        Users.extra != ""
    )).all()

# 通配符
ret = session.query(Users).filter(Users.name.like('%e%')).all()
ret = session.query(Users).filter(~Users.name.like('%e%')).all()

# 限制
ret = session.query(Users)[1:2]

# 排序

```

```

ret = session.query(Users).order_by(Users.name.desc()).all()
ret = session.query(Users).order_by(Users.name.desc(), Users.id.asc()).all()

# 分组
from sqlalchemy.sql import func

ret = session.query(Users).group_by(Users.extra).all()
ret = session.query(
    func.max(Users.id),
    func.sum(Users.id),
    func.min(Users.id)).group_by(Users.name).all()

ret = session.query(
    func.max(Users.id),
    func.sum(Users.id),
    func.min(Users.id)).group_by(Users.name).having(func.min(Users.id) >

# 连表

ret = session.query(Users, Favor).filter(Users.id == Favor.nid).all()

ret = session.query(Person).join(Favor).all()

ret = session.query(Person).join(Favor, isouter=True).all()

# 组合
q1 = session.query(Users.name).filter(Users.id > 2)
q2 = session.query(Favor.caption).filter(Favor.nid < 2)
ret = q1.union(q2).all()

q1 = session.query(Users.name).filter(Users.id > 2)
q2 = session.query(Favor.caption).filter(Favor.nid < 2)
ret = q1.union_all(q2).all()

```

更多功能参见文档，[猛击这里](#)下载PDF



作者：武沛齐

出处：<http://www.cnblogs.com/wupeiqi/>

本文版权归作者和博客园共有，欢迎转载，但未经作者同意必须保留此段声明，且在文章页面明显位置给出原文连接。

好文要顶

关注我

收藏该文



武沛齐

关注 - 44

粉丝 - 9974

+加关注

14

2

posted @ 2016-07-28 07:24 武沛齐 阅读(26993) 评论(3) 编辑 收藏

## 评论列表

#1楼 2020-03-22 11:31 mlwtc

[回复](#) [引用](#)

学习 好难

支持(0) 反对(0)

#2楼 2020-07-21 16:13 Xiyue666

[回复](#) [引用](#)

武大大 我学到这里想放弃了 怎么办？学不进去了！



支持(0) 反对(0)

#3楼 2020-08-07 10:04 serene1979

回复 引用





执行本网页中第一段SQL语句，其中执行语句改成：  
cursor.execute('insert into part(caption) values("AB") ')  
报错  
unable to resolve table 'part'  
unable to resolve column 'caption'  
unable to resolve column 'AB'  
麻烦问下是什么原因？

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

发表评论

编辑 预览

B    

支持 Markdown

提交评论 退出 订阅评论

[Ctrl+Enter快捷键提交]

- 【推荐】超50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】了不起的开发者，挡不住的华为，园子里的品牌专区
- 【推荐】如何在面试中成长？来看阿里前端终面官的面试心得

相关博文：

- Python开发【第十九篇】：Python操作MySQL
- Python开发【第十九篇】：Python操作MySQL
- Python开发【第十四篇】：Python操作MySQL
- 【Python之路】第十九篇--Python操作MySQL
- Python开发【第十九篇】：Python操作MySQL
- » 更多推荐...

最新 IT 新闻：

- 免费战场失利、新丽亏损 阅文是否能用换帅赢得未来？
- 小米的下一个十年 “互联网+制造”将如何落地？
- 大型科技股齐创新高 分析师为何独看好谷歌亚马逊？
- 苹果在华或遭滑铁卢：微信或被禁 iPhone12无亮点
- 马斯克的Neuralink 是先知的指引还是无知的妄想？
- » 更多新闻...