SQLAlchemy学习教程

 (2016-05-02 19:26:04)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
|  |  |

最近用到比较多的Python，当然访问数据库不能用原生的模块直接写Sql语句，这样太累。找了一本SQLAlchemy的教材来研究，记下学习笔记把。

相关的代码的例子在：

[**https://github.com/DoubleSpout/SQLAlchemy\_Demo**](https://github.com/DoubleSpout/SQLAlchemy_Demo)

**一、安装：**

先安装pip工具，然后设置好源服务器，用豆瓣的比较好，国外的源被墙了。

pip install MySQL-python

pip install SQLAlchemy

**二、Engine 数据库引擎**

想要连接使用SQLAlchemy连接数据库，必须先创建一个引擎来连接，下面是创建引擎的相关代码：

# -\*- coding: utf-8 -\*-

from sqlalchemy import \*

\_\_engine = create\_engine('mysql://root:@192.168.150.3/test',convert\_unicode=True, echo=True)

#conn 是DB-API连接对象

conn = \_\_engine.connect()

#result是数据库游标

result = conn.execute('select \* from tf\_user')

for row in result:

    print row

conn.close()

create\_engine 类接受的参数如下：

1、connect\_args 数据库链接参数，一般使用中文需要传递

{'charset':'utf8'}

2、convert\_unicode 将保存的unicode字符串数据转为2进制存入数据库，取出时也将2进制取出专程unicode，主要用于数据库不支持Unicode编码使用，默认 False

3、creator 一个可以被调用的对象（带有\_\_call\_\_的对象），返回DB-API的链接对象，默认None

4、echo 是否输出SQLAlchemy日志，包括拼接的sql语句，默认None

5、 echo\_pool 是否输出连接池日志，从连接池取出或放入，默认None

6、 encoding 设置传输数据的编码，默认是False

7、module 设置使用这个数据库哪一个连接模块，比如某些数据库安装了几个连接模块，默认False

8、pool 如果设定，则使用一个已经存在的连接池，不设定表示重新创建一个，默认None

9、 poolclass 表示这个引擎是否自己实现连接池类，否则会使用 sqlalchemy.pool.QueuePool ，而SQLite会使用 sqlalchemy.pool.SingletonThreadPool， 默认 None

10、 max\_overflow 表示连接池允许超出的连接数，默认是10

11、pool\_size 表示连接池数量，默认是5

12、pool\_recycle 单位秒，表示将闲置的连接释放掉，对于mysql数据库会自动释放闲置连接，有必要对这个值进行设置

13、pool\_timeout 从连接池中获取连接的超时时间，单位秒，默认是30

14、strategy 为这个连接引擎选择一个别用的策略，当前备用策略包括"threadlocal"和"plain"

15、"threadlocal" 在一个线程中重用一个连接，执行多条语句

16、"plain" （默认） 对每一条语句使用一个连接

17、 threaded 仅在Oracle数据库使用，默认False

18、 use\_ansi 仅在Oracle数据库使用

19、 use\_oids 仅在PostgreSQL数据库使用

此句返回的result是一个ResultProxy的实例

result = conn.execute('select \* from tf\_user')

具有如下的属性和方法：

1、\_\_iter\_\_() 可迭代

2、fetchone() 获取第一行数据，返回类型 RowProxy

3、fetchall() 获取所有的数据，返回类型  RowProxy 数组

4、scalar() ，获取第一列名的第一条数据，然后关闭游标

5、keys 属性，返回列明的数组

6、rowcount 属性，返回记录数

7、close() 关闭连接，将连接返回连接池

8、fetchmany(size=None)，根据数量返回RowProxy类型数组

此句迭代的row是RowProxy实例，具有如下属性和方法：

for row in result:

    print row

RowProxy 类型

1、\_\_getattr\_\_() 可以通过 object.column\_name 获取值

2、\_\_getitem\_\_() 可以通过 object[column\_name] 或者 object[column\_position] 获取值

3、keys() 提供所有 column\_name 的数组

4、values() 提供所有 value 值的数组

5、items() 提供一个 元组(column\_name,value)的数组

**三、MetaData 类**

MetaData主要用于保存表结构，连接字符串等数据，是一个多表共享的对象

metadata = MetaData() #生成一个未绑定数据源的metadata

bound\_meta = MetaData('sqlite:///test2.db') #绑定了数据源的metadata

我们可以利用meatadata.bind来绑定上面生成的数据源引擎

metadata = MetaData()

\_\_engine = create\_engine('mysql://root:@192.168.150.3/test', connect\_args={'charset': 'UTF8'},echo=True,

                         encoding='UTF-8')

metadata.bind = \_\_engine #将metadata绑定数据源

执行create\_all方法来创建表，这个操作是安全的操作，会先判断表是否存在

metadata.create\_all()

**四、Table类**

构造函数：

Table.\_\_init\_\_(self, name, metadata,\*args, \*\*kwargs)

1、name 表名

2、metadata 共享的元数据

3、\*args Column 是列定义，详见下一节

下面是可变参数 \*\*kwargs 定义

4、schema 此表的结构名称，默认None

5、autoload 自动从现有表中读入表结构，默认False

6、autoload\_with 从其他engine读取结构，默认None，例：

db = create\_engine('sqlite:///devdata.sqlite')

brand\_table = Table('brand', metadata, autoload=True, autoload\_with=db)

7、include\_columns 如果autoload设置为True，则此项数组中的列明将被引用，没有写的列明将被忽略，None表示所有都列明都引用，默认None

8、mustexist 如果为True，表示这个表必须在其他的python应用中定义，必须是metadata的一部分，默认False

9、useexisting 如果为True，表示这个表必须被其他应用定义过，将忽略结构定义，默认False

10、owner 表所有者，用于Orcal，默认None

11、quote 设置为True，如果表明是SQL关键字，将强制转义，默认False

12、quote\_schema  设置为True，如果列明是SQL关键字，将强制转义，默认False

13、mysql\_engine  mysql专用，可以设置'InnoDB'或'MyISAM'

**五、Column类**

构造函数：

Column.\_\_init\_\_(self,  name,  type\_,  \*args,  \*\*kwargs)

1、name 列名

2、type\_ 类型，更多类型 sqlalchemy.types

3、\*args Constraint（约束）,  ForeignKey（外键）,  ColumnDefault（默认）, Sequenceobjects（序列）定义

4、key 列明的别名，默认None

下面是可变参数 \*\*kwargs

5、primary\_key 如果为True，则是主键

6、nullable 是否可为Null，默认是True

7、default 默认值，默认是None

8、index 是否是索引，默认是True

9、unique 是否唯一键，默认是False

10、onupdate 指定一个更新时候的值，这个操作是定义在SQLAlchemy中，不是在数据库里的，当更新一条数据时设置，大部分用于updateTime这类字段

11、autoincrement 设置为整型自动增长，只有没有默认值，并且是Integer类型，默认是True

12、quote 如果列明是关键字，则强制转义，默认False

一个简单的表定义代码：

user\_table = Table(

    'tf\_user2', metadata,

    Column('id', Integer, primary\_key=True),

    Column('user\_name', Unicode(16, collation='utf8\_bin'), unique=True, nullable=False),

    Column('email\_address', Unicode(255, collation='utf8\_bin'), unique=True, nullable=False),

    Column('password', Unicode(40, collation='utf8\_bin'), nullable=False),

    Column('first\_name', Unicode(255, collation='utf8\_bin'), default=u'发'),

    Column('last\_name', Unicode(255, collation='utf8\_bin'), default=u'发'),

    Column('created', DateTime, default=datetime.now)

)

**六、约束定义：**

可以同时在列或者表中进行定义，相关代码如下：

product\_table = Table(

 'product', metadata,

 Column('brand\_id', Integer, ForeignKey('brand.id'),primary\_key=True),

 Column('sku', Unicode(80), primary\_key=True))

或者

product\_table = Table(

  'product', metadata,

  Column('brand\_id', Integer, ForeignKey('brand.id')), Column('sku', Unicode(80)),

  PrimaryKeyConstraint('brand\_id', 'sku', name='prikey'))

**七、外键的定义**

ForeignKey类构造函数如下：

ForeignKey.\_\_init\_\_(  self,  col-umn,  constraint=None,  use\_alter=False,  name=None,  onupdate=None,  ondelete=None)

**八、唯一定义**

Unique类，相关代码如下，当然也可以在列中定义，下面这种是复合唯一约束

product\_table = Table(

'product', metadata,

Column('id', Integer, primary\_key=True),

Column('brand\_id', Integer, ForeignKey('brand.id')),

Column('sku', Unicode(80)),

UniqueConstraint('brand\_id', 'sku'))

**九、检查约束**

例子：

payment\_table = Table(

 'payment', metadata,

 Column('amount', Numeric(10,2), CheckConstraint('amount > 0')))

 Column('original', Numeric(10,2), CheckConstraint('original> 0')),

 Column('discounted', Numeric(10,2),

 CheckConstraint('discounted > 0')),

 CheckConstraint('discounted < original', name='check\_constraint\_1'))

**十、index索引对象**

可以在列中定义，一般用到索引对象有下面几种情况：

1、定义多列复合索引

2、对索引命名

3、独立的对表创建索引，一般用于已经存在的表，增加索引

 使用例子

 i = Index('idx\_name', user\_table.c.first\_name,

 user\_table.c.last\_name, unique=True)

 i.create(bind=e)