The ORM Magic

Author: 占健豪, 王彦超, 陈致远, 汤佳伟

Date: 2021/5/25

Location: 22-206

Introduction

In this lab, we are going to learn the object-relational mapper (ORM) provided by SQLAlchemy.

With ORM, we can map a class to a database table, and map an object of that class to a row in

the database table. With SQLAlchemy's ORM, we can avoid directly using any raw SQL statements.

More important,we will be able to follow the principle of dependency inversion – let ORM depend

on the domain model,but not the other way around.

We will create 3 files:

> - model.py
> - orm.py
> - app.py

Here app.py imports the above two python modules and generates an SQLite database exactly

like EnglishPalDatabase.db

Materials and Methods

Work Flow

> 1. Review and analyze the requirements in lab2.pdf.
> 2. Learn about the relative knowledges in Chapter 2 of the course text book.
> 3. Start with the code.
> 4. Search for the coding techniques required online.
> 5. Finish the coding process.
> 6. Summarize and Write the document.

Source Codes

For this part, We implemented the incomplete function, class and used property to achieve the

requirements. See the source codes and comments for detail.

1. orm.py

```python
from sqlalchemy import Table, MetaData, Column, Integer, String, Date,
ForeignKey
from sqlalchemy.orm import mapper, relationship

import model

metadata = MetaData()

articles = Table(
    'articles',
    metadata,
    Column('article_id', Integer, primary_key=True,
autoincrement=True),
    Column('text', String(10000)),
    Column('source', String(100)),
    Column('date', String(10)),
    Column('level', Integer, nullable=False),
    Column('question', String(1000)),
)

users = Table(
    'users',
    metadata,
    Column('username', String(100), primary_key=True),
    Column('password', String(64)),
    Column('start_date', String(10), nullable=False),
    Column('expiry_date', String(10), nullable=False),
)

newwords = Table(
    'newwords',
    metadata,
    Column('word_id', Integer, primary_key=True, autoincrement=True),
    Column('username', String(100), ForeignKey('users.username')),
    Column('word', String(20)),
    Column('date', String(10)),
)

# ADDITION: add the reading part
readings = Table(
    'readings',
    metadata,
    Column('id', Integer, primary_key=True, autoincrement=True),
    Column('username', String(100), ForeignKey('users.username')),
    Column('article_id', Integer, ForeignKey('articles.article_id')),
)
def start_mappers():
    # ADDITION: implement the start_mapper()
    lines_mapper = mapper(model.User, users)
    lines_mapper = mapper(model.NewWord, newwords)
    lines_mapper = mapper(model.Article, articles)
    lines_mapper = mapper(model.Reading,readings)
    # pass
```

2. model.py

```python
from dataclasses import dataclass
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker


# ADDITION: just for convenience
engine = create_engine(
    r'sqlite:///D:\newDesktop\大三下
courses\SADP\lab2\test\EnglishPalDatabase.db')
get_session = sessionmaker(bind=engine)
session = get_session()

@dataclass
class Article:
    article_id: int
    text: str
    source: str
    date: str
    level: int
    question: str


class NewWord:
    def __init__(self, username, word='', date='yyyy-mm-dd'):
        self.username = username
        self.word = word
        self.date = date


class User:
    def __init__(self, username, password='12345', start_date='2021-
05-19', expiry_date='2031-05-19'):
        self.username = username
        self.password = password
        self.start_date = start_date
        self.expiry_date = expiry_date
        self._read = []

    def read_article(self, article):
        # ADDITION: implement the action
        session.add(article)
        reading = Reading(self.username, article.article_id)
        session.add(reading)
        session.commit()
        # pass

    # ADDITION: use property to achieve list(user.newwords)
    @property
    def newwords(self):
        words = session.query(NewWord).filter(NewWord.username ==
self.username).all()
        # test code
        # for w in words:
        #     print(w.word)
        return words
```

```
56
57
58  # ADDITION: implement the Reading class
59  class Reading:
        def __init__(self, username, article_id):
            self.username = username
            self.article_id = article_id
```

3. app.py

```
1   from sqlalchemy import create_engine
2   from sqlalchemy.orm import sessionmaker
3
4   import model
5   import orm
6
7   orm.start_mappers()
8   engine = create_engine(
9       r'sqlite:///D:\newDesktop\大三下
10  courses\SADP\lab2\test\EnglishPalDatabase.db') # modify the path
11  orm.metadata.drop_all(engine)
12  orm.metadata.create_all(engine)
13  get_session = sessionmaker(bind=engine)
14
15  # add two users
16
17  session = get_session()
18
19  try:
20      session.add(model.User(username='mrlan', password='12345',
21  start_date='2021-05-14'))
22      session.add(model.User(username='lanhui', password='Hard2Guess!',
23  start_date='2021-05-15'))
24      session.commit()
25  except:
26      print('Duplicate insertions.')
27
28  print(session.query(model.User).count())
29
30  for u in session.query(model.User).all():
31      print(u.username)
32
33  session.close()
34
35  # add a few new words
36
37  session = get_session()
38  session.add(model.NewWord(username='lanhui', word='starbucks',
39  date='2021-05-15'))
40  session.add(model.NewWord(username='lanhui', word='luckin',
41  date='2021-05-15'))
42  session.add(model.NewWord(username='lanhui', word='secondcup',
43  date='2021-05-15'))
44  session.add(model.NewWord(username='mrlan', word='costa', date='2021-
45  05-15'))
46  session.add(model.NewWord(username='mrlan', word='timhortons',
47  date='2021-05-15'))
```

```
session.commit()
session.close()

# add a few articles

session = get_session()
article = model.Article(article_id=1,
                        text='THE ORIGIN OF SPECIES BY MEANS OF
NATURAL SELECTION, OR THE PRESERVATION OF FAVOURED RACES IN THE
STRUGGLE FOR LIFE',
                        source='CHARLES DARWIN, M.A.', date='1859-01-
01', level=5,
                        question='Are humans descended from monkeys?')
session.add(article)
session.commit()
session.close()

# query user and let him read something

session = get_session()
user = session.query(model.User).filter_by(username='lanhui').one()

for item in list(user.newwords):
    print(item.word)

user.read_article(article)  # this method call will add a row to table
readings

print('-----')

user = session.query(model.User).filter_by(username='mrlan').one()
for item in list(user.newwords):
    print(item.word)

user.read_article(article)  # this method call will add a row to table
readings

session.commit()
session.close()
```

Results

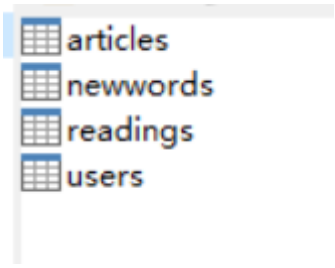
For this part we make **screenshots** to illustrate the results.

1. After running **app.py**:

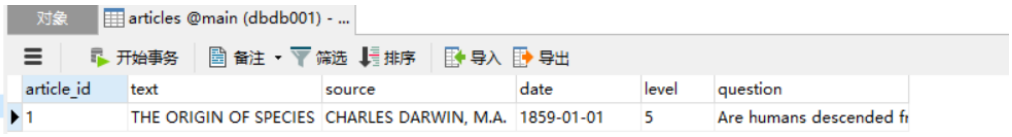
```
Run: app
D:\wangyc\Python\sadp_lab2\venv\Scripts\python.exe D:/wangyc/Python/sadp_lab2/app.py
2
mrlan
lanhui
starbucks
luckin
secondcup
-----
costa
timhortons

Process finished with exit code 0
```

2. Inside **EnglishPalDatabase.db(Open with Navicat Premium)**:

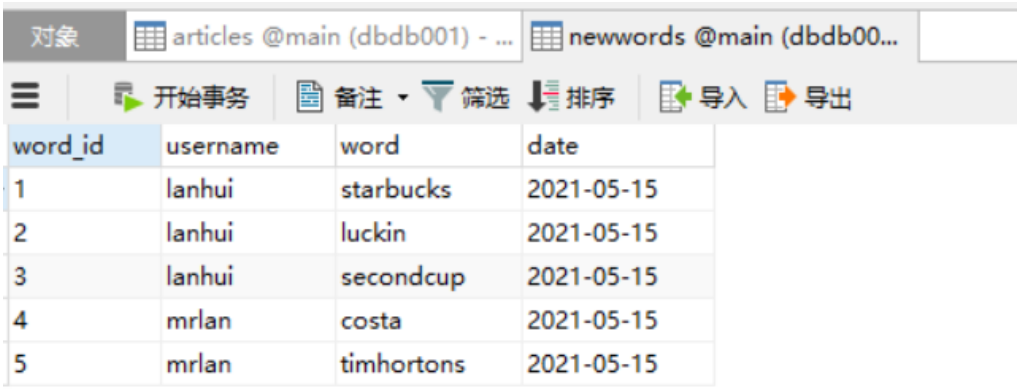
   - list of tables:



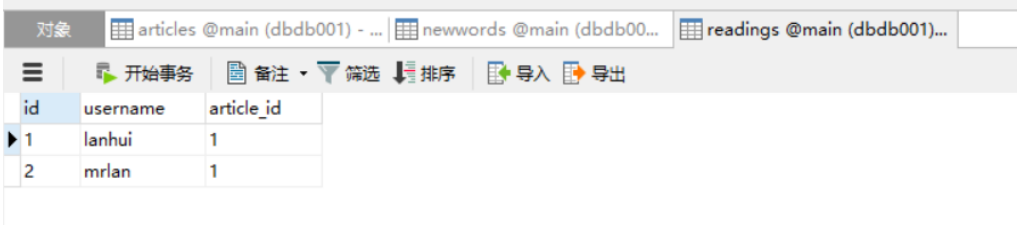
   - articles:



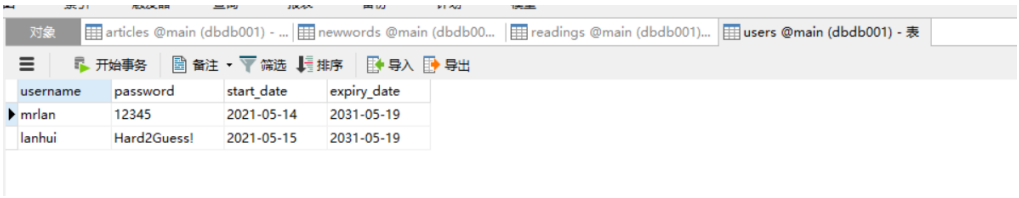
   - newwords:



   - readings:



- users:



Discussions

- For this lab we learnt about the way to manipulate database with SQLAlchemy's ORM (object-relational mapper) instead of raw SQL statement in web application, which will bring convenience while making the architecture more clear.
- We tried to understand dependency inversion.
- Also, we learnt to use Read the Docs combining with Sphinx to manage our lab report.

References

lab2.pdf