

LAB2 - The ORM Magic

Author: 占健豪, 王彦超, 陈致远, 汤佳伟

Date: 2021/5/26

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

```
1  from sqlalchemy import Table, MetaData, Column, Integer, String, Date,
2  ForeignKey
3  from sqlalchemy.orm import mapper, relationship
4
5  import model
6
7  metadata = MetaData()
8
9  articles = Table(
10     'articles',
11     metadata,
12     Column('article_id', Integer, primary_key=True, autoincrement=True),
13     Column('text', String(10000)),
14     Column('source', String(100)),
15     Column('date', String(10)),
16     Column('level', Integer, nullable=False),
17     Column('question', String(1000)),
18 )
19
20 users = Table(
21     'users',
22     metadata,
23     Column('username', String(100), primary_key=True),
24     Column('password', String(64)),
25     Column('start_date', String(10), nullable=False),
26     Column('expiry_date', String(10), nullable=False),
27 )
28
29 newwords = Table(
30     'newwords',
31     metadata,
32     Column('word_id', Integer, primary_key=True, autoincrement=True),
33     Column('username', String(100), ForeignKey('users.username')),
34     Column('word', String(20)),
35     Column('date', String(10)),
36 )
37
38 # ADDITION: add the reading part
39 readings = Table(
40     'readings',
41     metadata,
42     Column('id', Integer, primary_key=True, autoincrement=True),
43     Column('username', String(100), ForeignKey('users.username')),
44     Column('article_id', Integer, ForeignKey('articles.article_id')),
45 )
46
47 def start_mappers():
48     # ADDITION: implement the start_mapper()
49     lines_mapper = mapper(model.User, users)
50     lines_mapper = mapper(model.NewWord, newwords)
51     lines_mapper = mapper(model.Article, articles)
52     lines_mapper = mapper(model.Reading, readings)
53     # pass
```

2. model.py

```

1  from dataclasses import dataclass
2  from sqlalchemy import create_engine
3  from sqlalchemy.orm import sessionmaker
4
5
6  # ADDITION: just for convenience
7  engine = create_engine(
8      r'sqlite:///D:\newDesktop\大三下
9  courses\SADP\lab2\test\EnglishPalDatabase.db')
10 get_session = sessionmaker(bind=engine)
11 session = get_session()
12
13 @dataclass
14 class Article:
15     article_id: int
16     text: str
17     source: str
18     date: str
19     level: int
20     question: str
21
22
23 class NewWord:
24     def __init__(self, username, word='', date='yyyy-mm-dd'):
25         self.username = username
26         self.word = word
27         self.date = date
28
29
30 class User:
31     def __init__(self, username, password='12345',
32 start_date='2021-05-19', expiry_date='2031-05-19'):
33         self.username = username
34         self.password = password
35         self.start_date = start_date
36         self.expiry_date = expiry_date
37         self._read = []
38
39     def read_article(self, article):
40         # ADDITION: implement the action
41         session.add(article)
42         reading = Reading(self.username, article.article_id)
43         session.add(reading)
44         session.commit()
45         # pass
46
47     # ADDITION: use property to achieve list(user.newwords)
48     @property
49     def newwords(self):
50         words = session.query(NewWord).filter(NewWord.username ==
51 self.username).all()
52         # test code
53         # for w in words:
54         #     print(w.word)
55         return words
56
57

```

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',
45 date='2021-05-15'))
46 session.add(model.NewWord(username='mrlan', word='timhortons',
47 date='2021-05-15'))
48 session.commit()
49 session.close()
50
51 # add a few articles
52
53 session = get_session()
54 article = model.Article(article_id=1,
55                          text='THE ORIGIN OF SPECIES BY MEANS OF NATURAL
56 SELECTION, OR THE PRESERVATION OF FAVOURED RACES IN THE STRUGGLE FOR
57 LIFE',

```

Results

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

1. After running **app.py**:/imgs/apppy_res.png
2. Inside **EnglishPalDatabase.db**(Open with Navicat Premium):
 - a. list of tables:

