数据库ORM操作

1. 数据库ORM操作实现

为了实现ORM方式操作数据库，而不是使用SQL语句，就需要通过安装 `Flask-SQLAlchemy` 扩展库来实现。同时通过安装 `Flask-Migrate` 扩展库实现数据库更新的跟踪，并实现数据迁移。

```
pip install Flask-SQLAlchemy
pip install Flask-Migrate
```

2. 配置sqlite数据库

对于小型应用来说，使用sqlite数据库更为便利，每一个数据库都单独存放在一个文件中。使用sqlite数据库需要在 `app/config.py` 配置文件进行相关配置。

```
# 配置SQLITE数据库信息
BASE_DIR = os.path.abspath(os.path.dirname(__file__))
# 数据库文件存放路径
SQLALCHEMY_DATABASE_URI = 'sqlite:///' + os.path.join(BASE_DIR, '..', 'instance',
'flask.sqlite')
# Flask-SQLAlchemy 是否需要追踪对象的修改并且发送信号。
# 这需要额外的内存， 如果不必要的可以禁用它。
SQLALCHEMY_TRACK_MODIFICATIONS = False
```

3. 初始化数据库

修改`app/__init__.py`初始化文件，初始化数据库

```
......
import os
from flask import Flask
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate


# 实例化flask_sqlalchemy
db = SQLAlchemy()
# 实例化flask_migrate
migrate = Migrate()


def create_app():
    """应用工厂函数"""
    application = Flask(__name__)
    # 加载config配置文件
    application.config.from_pyfile('config.py', silent=True)

    # 初始化数据库flask_sqlalchemy
    db.init_app(application)
    # 初始化数据库flask_migrate
    import app.models
    migrate.init_app(application, db)
......
```

4. 创建数据库模型

新建`app/models.py`文件，创建用户信息的数据库模型。

```
from app import db


class User(db.Model):
```

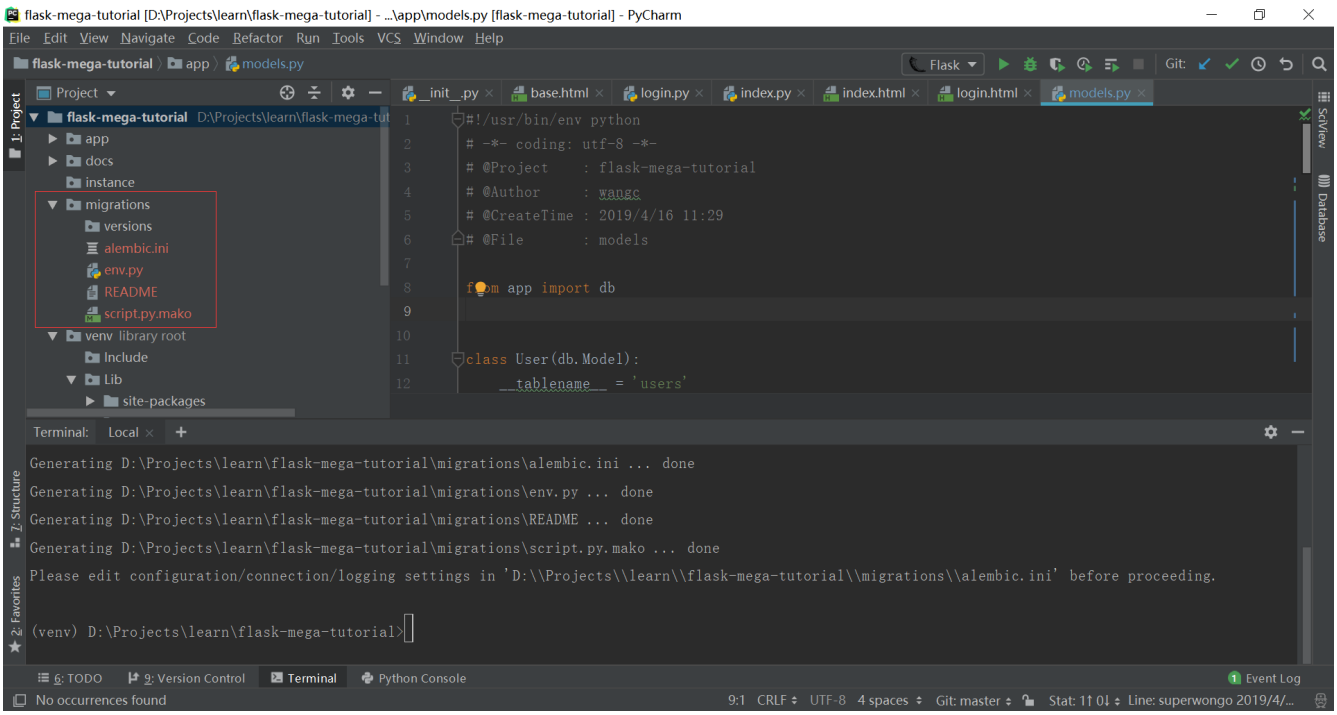
```
    __tablename__ = 'users'
    id = db.Column(db.Integer, primary_key=True)
    # index用于添加索引
    # unique用于设置唯一索引
    username = db.Column(db.String(64), index=True, unique=True)
    email = db.Column(db.String(120), index=True, unique=True)
    password_hash = db.Column(db.String(128))

    def __repr__(self):
        """打印类对象时的展示方式"""
        return '<User %r>' % self.username
```

5. 数据库模型初始化

5.1. 迁移目录初始化

通过`flask db init`命令，初始化数据库模型迁移目录`migrations`。

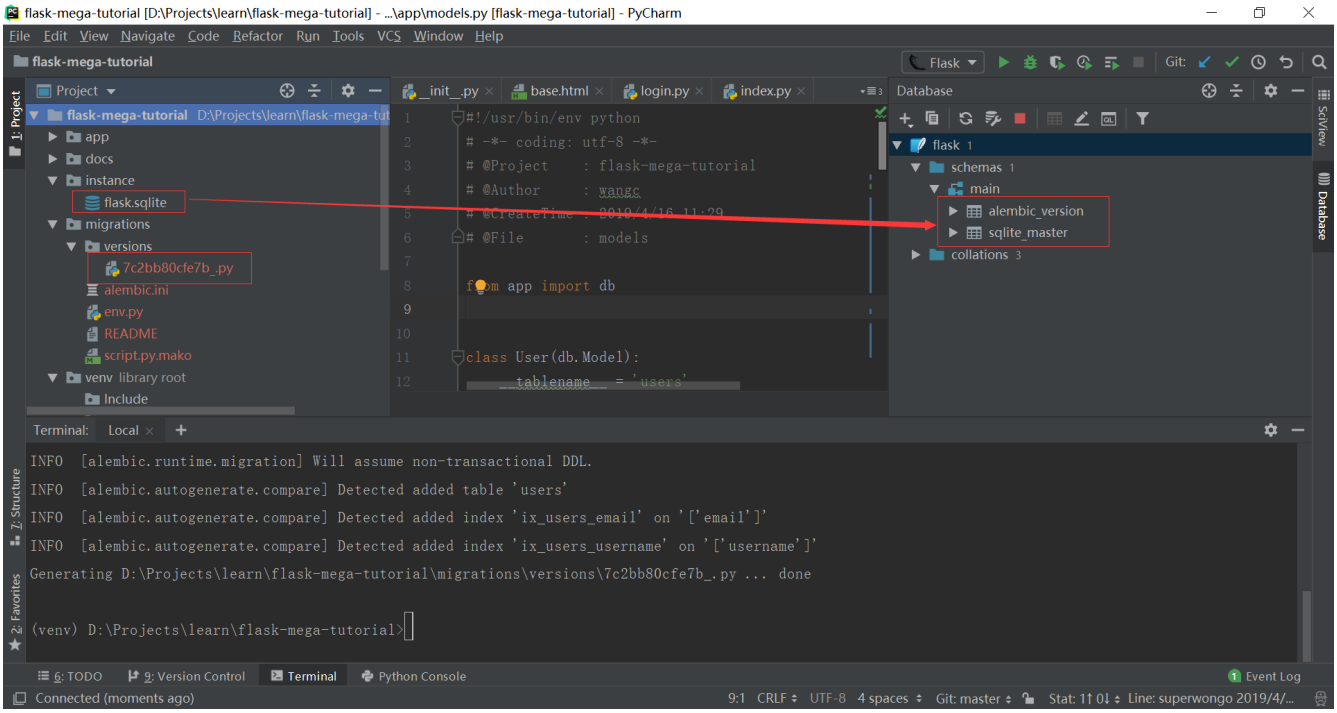
```
(venv) D:\Projects\learn\flask-mega-tutorial>flask db init
Creating directory D:\Projects\learn\flask-mega-tutorial\migrations ... done
Creating directory D:\Projects\learn\flask-mega-tutorial\migrations\versions ... done
Generating D:\Projects\learn\flask-mega-tutorial\migrations\alembic.ini ... done
Generating D:\Projects\learn\flask-mega-tutorial\migrations\env.py ... done
Generating D:\Projects\learn\flask-mega-tutorial\migrations\README ... done
Generating D:\Projects\learn\flask-mega-tutorial\migrations\script.py.mako ... done
Please edit configuration/connection/logging settings in 'D:\\Projects\\learn\\flask-mega-tutorial\\migrations\\alembic.ini' before proceeding.
```



5.2. 迁移文件创建

通过`flask db migrate`命令，将数据库迁移文件生成到迁移目录`versions`中，同时也初始化创建了`Flask-Migrate`扩展库依赖的数据库表。

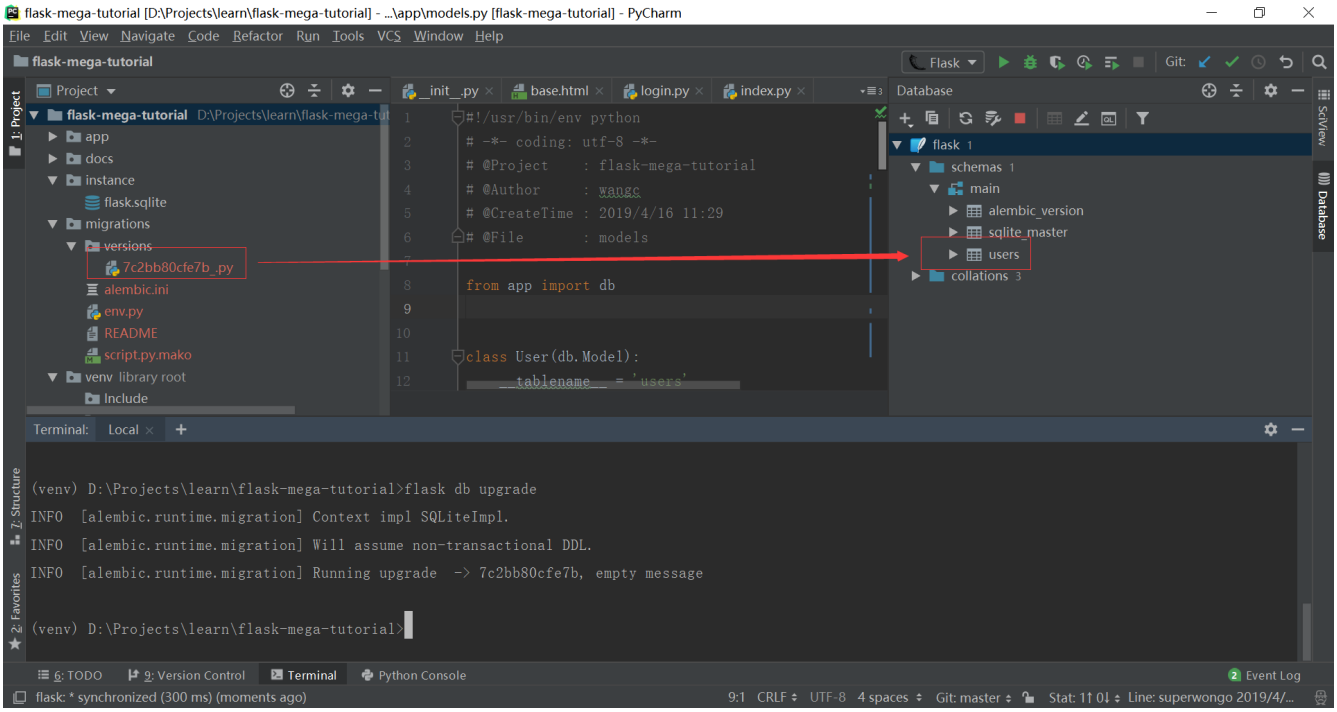
```
(venv) D:\Projects\learn\flask-mega-tutorial>flask db migrate
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.autogenerate.compare] Detected added table 'users'
INFO  [alembic.autogenerate.compare] Detected added index 'ix_users_email' on '['email']'
INFO  [alembic.autogenerate.compare] Detected added index 'ix_users_username' on
'['username']'
Generating D:\Projects\learn\flask-mega-tutorial\migrations\versions\7c2bb80cfe7b_.py ...
done
```



5.3. 数据库同步

通过`flask db upgrade`命令，将数据库迁移文件的变化同步到数据库中。

```
(venv) D:\Projects\learn\flask-mega-tutorial>flask db upgrade
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade  -> 7c2bb80cfe7b, empty message
```



6. 新增数据库模型、数据库同步

6.1. 新增数据库模型定义

编辑`app/models.py`文件，新增posts表的数据库模型定义

```
from app import db


class User(db.Model):
    __tablename__ = 'users'
    id = db.Column(db.Integer, primary_key=True)
    # index用于添加索引
    # unique用于设置唯一索引
    username = db.Column(db.String(64), index=True, unique=True)
    email = db.Column(db.String(120), index=True, unique=True)
    password_hash = db.Column(db.String(128))
    # 初始化db.relationship
    # 并非实际的数据库字段，只是创建一个虚拟的列，该列会与 Post.user_id (db.ForeignKey) 建立联系
    # 第一个参数表示关联的模型类名；backref用于指定表之间的双向关系；lazy用于定义加载关联对象的方式
    posts = db.relationship('Post', backref='author', lazy='dynamic')

    def __repr__(self):
        """打印类对象时的展示方式"""
        return '<User %r>' % self.username


class Post(db.Model):
    __tablename__ = 'posts'
    id = db.Column(db.Integer, primary_key=True)
```

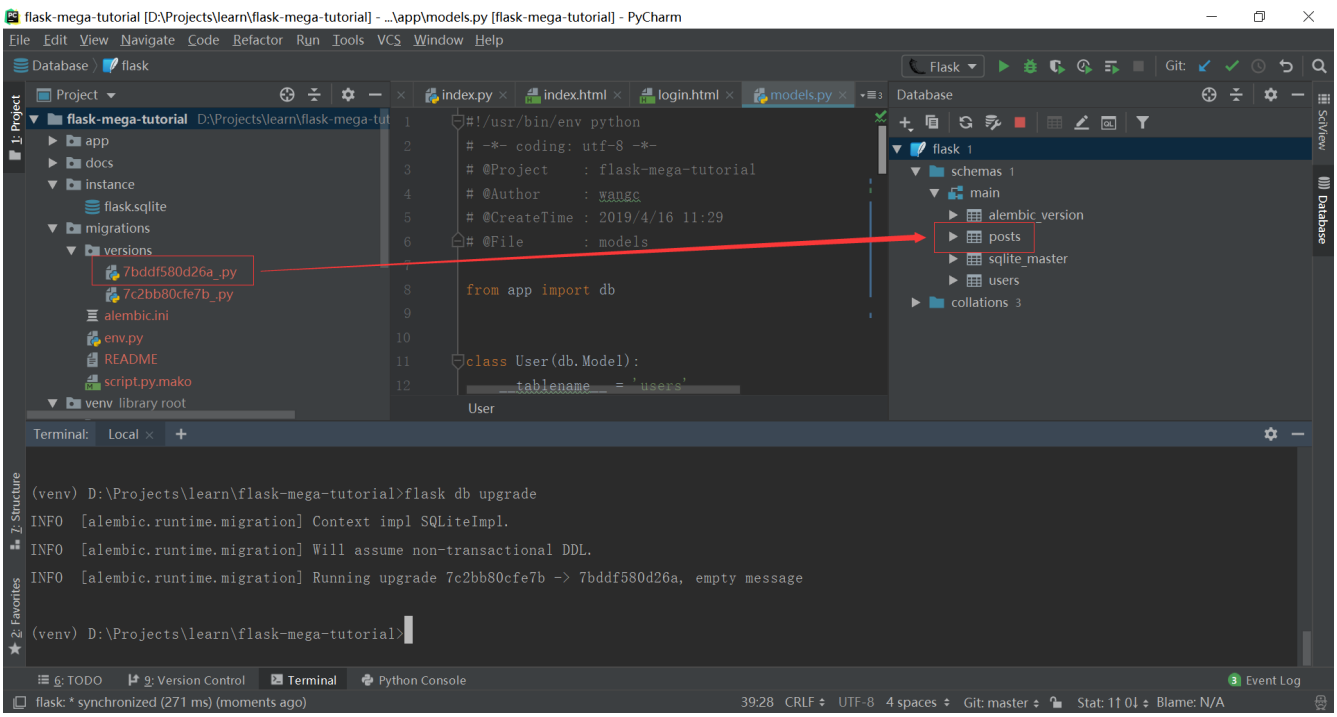
```
    body = db.Column(db.String(140))
    timestamp = db.Column(db.DateTime)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'))

    def __repr__(self):
        return '<Post %r>'.format(self.body)
```

6.2. 迁移文件创建

```
(venv) D:\Projects\learn\flask-mega-tutorial>flask db migrate
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.autogenerate.compare] Detected added table 'posts'
Generating D:\Projects\learn\flask-mega-tutorial\migrations\versions\7bddf580d26a_.py ...
done
```

6.3. 数据库同步

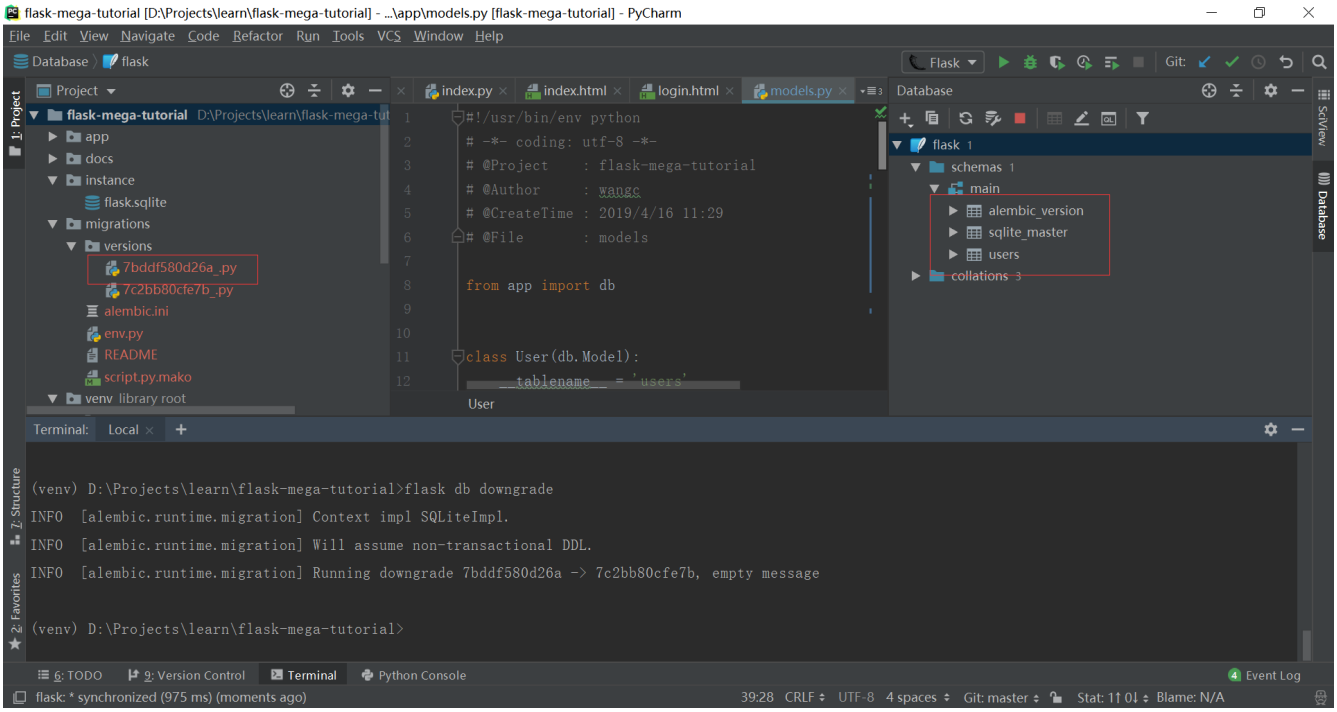
```
(venv) D:\Projects\learn\flask-mega-tutorial>flask db upgrade
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade 7c2bb80cfe7b -> 7bddf580d26a, empty
message
```



7. 数据库操作回退

通过`flask db downgrade`命令，对已同步的数据库进行回退处理，此操作仅限于**数据库同步**的回退，但是迁移文件还在。

```
(venv) D:\Projects\learn\flask-mega-tutorial>flask db downgrade
INFO  [alembic.runtime.migration] Context impl SQLiteImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.runtime.migration] Running downgrade 7bddf580d26a -> 7c2bb80cfe7b, empty
message
```



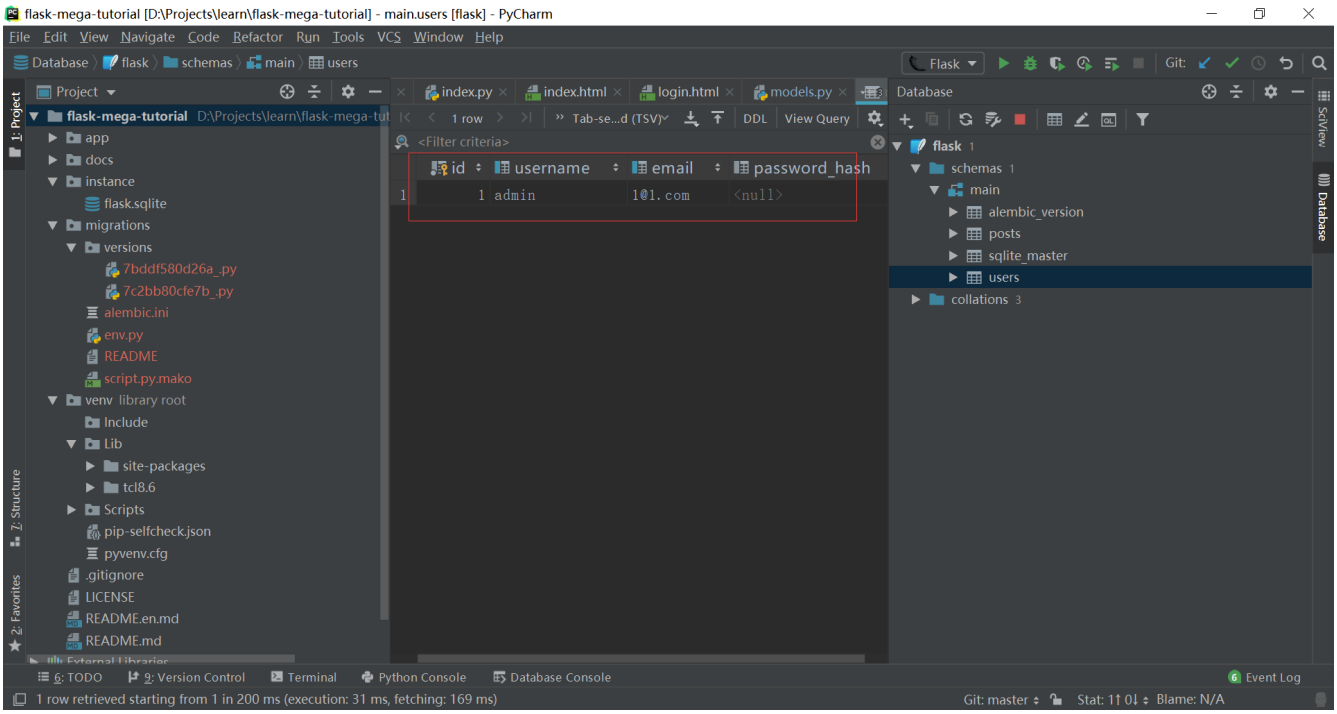
8. 数据库操作测试

8.1. 数据查询

```
>>> from app import create_app, db
>>> app = create_app()
>>> app.app_context().push()
>>> with app.app_context():
...     from app.models import User
...     user = User.query.all()
...     print(user)
...
[]
```

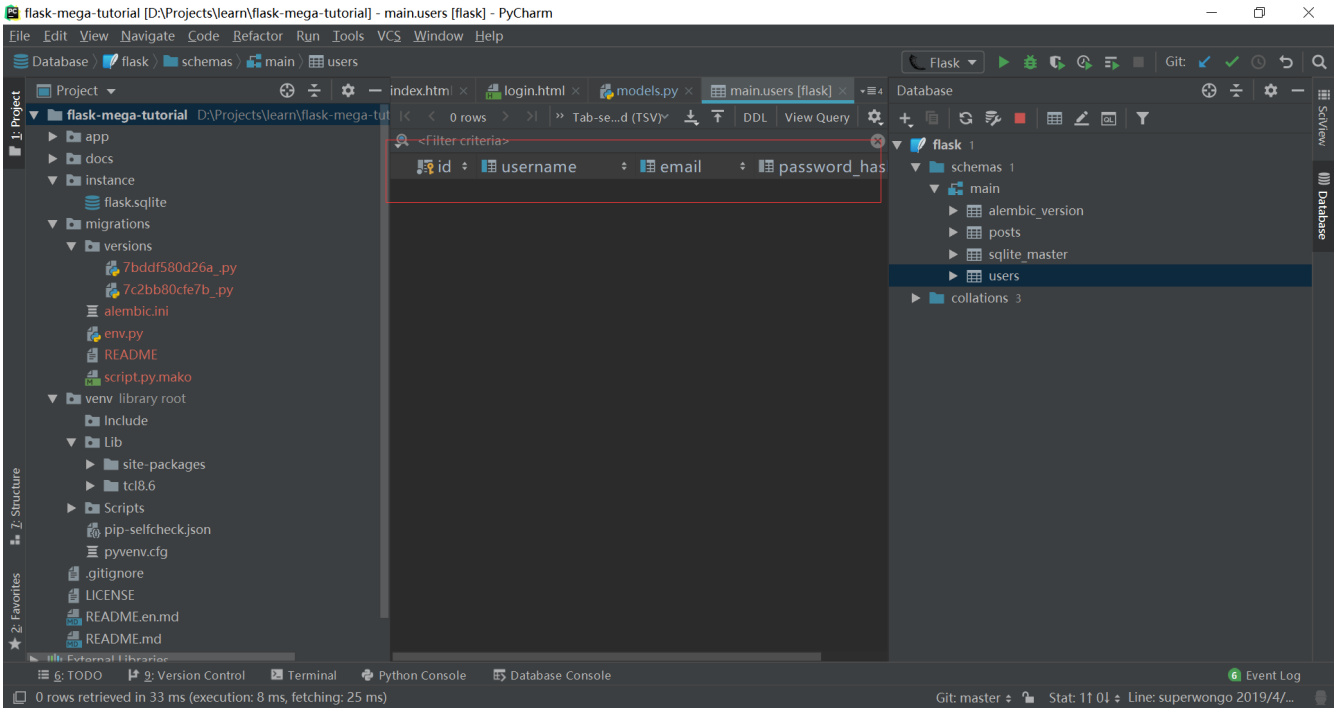
8.2. 数据插入

```
>>> from app import create_app, db
>>> app = create_app()
>>> app.app_context().push()
>>> with app.app_context():
...     from app.models import User
...     user = User(username='admin', email='1@1.com')
...     db.session.add(user)
...     db.session.commit()
...     db.session.close()
...     users = User.query.all()
...     print(users)
...
[<User 'admin'>]
```



8.3. 数据删除

```
>>> with app.app_context():
...     from app.models import User
...     db.session.delete(User.query.filter(User.username == 'admin').one())
...     db.session.commit()
...     db.session.close()
...     user = User.query.all()
...     print(user)
...
[]
```


flask-mega-tutorial [D:\Projects\learn\flask-mega-tutorial] - main.users [flask] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Database flask schemas main users
Flask
Git:
Project
flask-mega-tutorial D:\Projects\learn\flask-mega-tut
app
docs
instance
flask.sqlite
migrations
versions
7bddf580d26a_.py
7c2bb80cfe7b_.py
alembic.ini
env.py
README
script.py.mako
venv library root
Include
Lib
site-packages
tcl8.6
Scripts
pip-selfcheck.json
pyvenv.cfg
.gitignore
LICENSE
README.en.md
README.md
index.html
login.html
models.py
main.users [flask]
0 rows
Tab-se...d (TSV)
DDL
View Query
<Filter criteria>
id
username
email
password_has
Database
flask 1
schemas 1
main
alembic_version
posts
sqlite_master
users
collations 3
SciView
Database
1: Project
7: Structure
2: Favorites
6: TODO
9: Version Control
Terminal
Python Console
Database Console
Event Log
0 rows retrieved in 33 ms (execution: 8 ms, fetching: 25 ms)
Git: master
Stat: 1↑ 0↓
Line: superwongo 2019/4/...