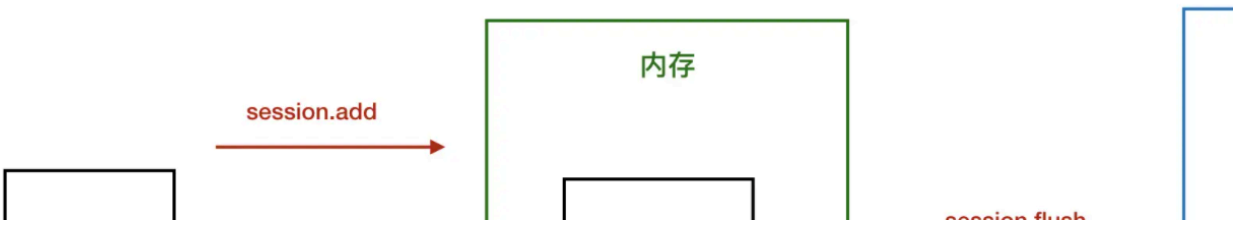
SQLAlchemy

1. 数据刷新

- 有两种情况下会 **隐式执行刷新操作**:
    - 提交会话 db.session.commit()
    - 执行查询操作 (包括 `update` 和 `delete` 子查询)
- 开发者也可以 **手动执行刷新操作** `session.flush()`



2. 外键关联

数据关联

```python
# TODO:
# 1. 定义用户模型类
class User(db.Model):
    __tablename__ = 'tb_user'
    id = Column(Integer, primary_key=True)
    name = Column(String(32))
    age = Column(Integer)


# 2. 定义用户地址模型类
class Address(db.Model):
    __tablename__ = 'tb_addr'
    id = Column(Integer, primary_key=True)
    detail = Column(String(128))
    user_id = Column(Integer, nullable=True)
```

```python
#     创建新的用户
#     创建用户地址(注意: 在创建地址之前, 需要先手动进行 flush, 否则 user_id 获取的是错误的数据 None)
@app.route('/create_user')
def create_user():
    user = User(name='zhangsan', age=100)
    db.session.add(user)  # 用户数据在内存中, 不在数据库中
    db.session.flush() # 手动同步数据到数据库

    addr1 = Address(detail='shanghai', user_id=user.id)
    addr2 = Address(detail='beijing', user_id=user.id)
    db.session.add_all([addr1, addr2])

    db.session.commit()
    return 'create user'
```

关联查询

```python
# TODO:
# 定义接口
#     1. 通过用户模型类查询特定的用户
#     2. 通过地址模型类查询用户的地址, filter(Address.user_id=用户 id)
@app.route('/read_user_address')
def read_user_address():
    user = User.query.filter(User.name == 'zhangsan').first()
    if user:
        addresses = Address.query.filter(Address.user_id == user.id).all()
        for address in addresses:
            print(address.id, address.detail, address.user_id)

    return 'read addresses'
```

3. 关系属性

```python
# 3. 定义用户模型类
#    定义 关系属性名 = relationship(地址模型类名字符串)
class User(db.Model):
    __tablename__ = 'tb_user'
    id = Column(Integer, primary_key=True)
    name = Column(String(32))
    age = Column(Integer)
    addresses = relationship('Address')


# 4. 定义地址模型类
#    定义 user_id = Column(Integer, ForeignKey('用户表名.主键'))
class Address(db.Model):
    __tablename__ = 'tb_addr'
    id = Column(Integer, primary_key=True)
    detail = Column(String(128))
    user_id = Column(Integer, ForeignKey('tb_user.id'))
```

```python
# 6. 定义接口
#    1. 查询用户
#    2. 通过用户.关系属性名 读取地址信息
@app.route('/read_user_address')
def read_user_address():
    user = User.query.filter(User.name == 'zhangsan').first()
    if user:
        for address in user.addresses:
            print(address.id, address.detail, address.user_id)

    return 'read user address'


if __name__ == '__main__':
```

4. 连接查询

```python
# TODO
# join 的语法
# db.session.query(模型类1的字段,...,模型2的字段,...).join(模型类2, 模型类的字段==模型类2的字段)
@app.route('/join_query')
def join_query():
    data = db.session.query(User.name, Address.detail) \
        .join(Address, User.id == Address.user_id) \
        .filter(User.name == 'zhangsan').all()
    for i in data:
        print(i.name, i.detail)

    return 'join query'


if __name__ == '__main__':
    db.drop_all()
```

5. session 机制介绍

session 机制演示

```python
# 3. 定义接口
#     1. 创建一个 username 相同的用户,
#         这段代码包装在 try...except 中
#         在 except 中手动调 db.session.rollback() 进行回滚
#         否则无法创建新的事务
#     2. 创建一个 username 不同的用户, 提交数据
@app.route('/create_user')
def create_user():
    # 事务 1
    try:
        user = User(username='zhangsan')
        db.session.add(user)
        db.session.commit()
    except Exception:
        # db.session.rollback()
        pass

    # 事务 2
    user = User(username='lisi')
    db.session.add(user)
    db.session.commit()

    return 'create user'
```

6. 数据迁移

```python
# 迁移组件初始化
Migrate(app, db)


# 构建模型类
class User(db.Model):
    __tablename__ = 't_user'
    id = Column(Integer, primary_key=True)
    name = Column('username', String(20), unique=True)
    age = Column(Integer, default=0, index=True)
    street = Column(String(128))


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8000, debug=True)
```

- 执行迁移命令

```
export FLASK_APP=main.py  # 设置环境变量指定启动文件
flask db init  # 生成迁移文件夹
flask db migrate  # 生成迁移版本, 保存到迁移文件夹中
flask db upgrade  # 执行迁移
```

项目

产品介绍

项目架构

需求分析

数据模型介绍

http://git-meiduo.itheima.net//sh-python39/tianzhichao-topnews.git

0. 综合练习题

模型定义

商品模型类、订单模型类、订单详情模型类、购物车模型类、购物车详情模型类、用户模型类

模型关系: 每个订单包含多个订单详情、每个订单详情关联一个商品、每个用户有多个订单、每个用户只有一个购物车、每个购物车包含多个购物车详情

商品字段: 商品名、介绍、价格、库存、类别

订单字段: 订单号、总价、创建时间、订单状态、关联的用户

订单详情:字段 关联的订单、关联的商品、商品数量、总价

购物车字段: 关联的用户、购物车中商品总价、总共包含的商品数量

购物车详情字段: 关联的购物车、关联的商品、关联的商品数量、总价

用户字段: 手机号码、密码

用户蓝图模块构建

要求:

1. 路由前缀 '/user'
2. 定义用户类视图
    1. get 获取当前用户信息
    2. post 注册用户
    3. delete 退出登录
    4. put 用户登录

订单蓝图模块构建

要求:

1. 路由前缀 '/order'

2. 定义订单类视图
    1. get 方获取当前用户所有的订单
    2. post 新建订单
    3. delete 用于删除订单

购物车蓝图模块构建

要求:

1. 路由前缀 '/order'
2. 定义购物车类视图
    1. get 获取当前用户的购物车
    2. post 向购物车添加商品
    3. delete 从购物车移除商品