1.数据管理

Flask-SQLAlchemy中，插入、修改、删除操作，均由数据库会话db.session管理。

在准备把数据写入数据库前需要db.session.add()将数据添加到会话。

数据添加到会话中后调用db.session.commit()方法提交会话保存数据到数据库。

如果已经有数据添加到会话，但是中间有异常不提交数据，db.session.rollback()方法回滚会话。

```
# -*- encoding: utf-8 -*-

# -*- encoding: utf-8 -*-
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)  # type:Flask

# 设置连接的数据库  注意采用pymysql连接 mysql+pymysql
app.config['SQLALCHEMY_DATABASE_URI'] =
'mysql+pymysql://root:qwe123@127.0.0.1:3306/py'
# 创建数据库对象
db = SQLAlchemy(app)


class Father(db.Model):
    """创建一个父亲模型类"""
    # 表名
```

```
    __tablename__ = 'father'
    # id主键列，整数类型，自增
    id = db.Column(db.Integer,primary_key=True)
    # name，可变长字符串类型
    name = db.Column(db.String(20))
    # 关系字段，不是数据库中真实存在的字段，而是为了方便查询添加的
属性
    son = db.relationship('Son', backref='father')


class Son(db.Model):
    # 表名
    __tablename__ = 'son'
    # id主键列，整数类型，自增
    id = db.Column(db.Integer,primary_key=True)
    # name，可变长字符串类型
    name = db.Column(db.String(20))
    # 外键，需要指定字段类型，以及关联那个表的那个字段
    father_id = db.Column(db.Integer,
db.ForeignKey('father.id'))



if __name__ == '__main__':
    # 创建手动应用上下文（激活） --- 避免异步处理 RuntimeError:
Working outside of application context
    with app.app_context():  # 应用上下文app context，存储的是应
用级别的信息，比如数据库连接信息
        # 删除所有表，注意这条是危险命令，会将模型类对应数据库中的
表物理删除。在实际生产环境下勿用。
        db.drop_all()
        # 创建所有表
        db.create_all()

        # 创建一个 mother
        m = Father(name='围裙妈妈')
        db.session.add(m)
        # 创建了一个妈妈，但是我们这里是Father类，需要回滚会话
        db.session.rollback()
        # 创建一个father对象
```

```
        f = Father(name='小头爸爸')
        # 将数据添加到会话
        db.session.add(f)
        # 提交会话数据
        db.session.commit()

        # 创建两个Son对象,这里利用 到father对象id,需要先将Father对象提交才能拿到id
        s = Son(name='大头儿子', father_id=f.id)
        s1 = Son(name='中头儿子', father_id=f.id)
        # 同时添加多个数据，接收一个列表做为参数
        db.session.add_all([s, s1])
        # 提交会话数据
        db.session.commit()
```