**6.1数据库模型层·使用ORM模型操作数据库**

当我们使用SQLAlchemy这类ORM框架后，我们可以有一个python的实体类与数据库相对应，这样操作数据库会更加方便一些，下面我们来体验一下：

# 创建模型模块models

mkdir models

# 新建models/\_\_init\_\_.py

from flask\_sqlalchemy import SQLAlchemy  
  
db = SQLAlchemy()

# 新建一个模型类models/user.py

from datetime import datetime  
  
from models import db  
  
  
class User(db.Model):  
 \_\_tablename\_\_ = "t\_user"  
 \_\_table\_args\_\_ = ({"comment": "用户"})  
 id = db.Column(db.Integer, primary\_key=True, comment="主键")  
 user\_name = db.Column(db.String(32), name="user\_name", unique=True, nullable=False, comment="用户名")  
 real\_name = db.Column(db.String(32), name="real\_name", unique=False, nullable=False, comment="姓名")  
 password = db.Column(db.String(64), name="password", unique=False, nullable=False, comment="密码")  
 create\_time = db.Column(db.DateTime, name="create\_time", onupdate=datetime.now, comment="创建时间")  
 update\_time = db.Column(db.DateTime, name="update\_time", default=datetime.now, onupdate=datetime.now, comment="更新时间")  
 is\_deleted = db.Column(db.Boolean, name="is\_deleted", default=False, comment="逻辑删除:0=未删除,1=删除")

# 修改controllers/user\_controller.py

from flask import Blueprint  
  
from controllers import R  
from models import db  
from models.user import User  
from validators import BasePageForm  
from validators.id\_validator import IdForm, IdsForm  
from validators.user\_validator import UserForm  
  
user = Blueprint('user', \_\_name\_\_, url\_prefix="/user")  
  
  
@user.route("/get", methods=['POST'])  
def user\_get():  
 """  
 通过id获取用户信息  
 :return:  
 """  
 form = IdForm()  
 form.validate\_for\_api()  
 # 可通过form.data获取所有提交参数  
 # 或者直接拿id值 id=form.id.data  
 # u = User.query.filter\_by(id=form.id.data).first()  
 # 通过主键查询  
 u = User.query.get(form.id.data)  
 if u is not None:  
 return R.data({  
 "id": u.id,  
 "userName": u.user\_name,  
 "realName": u.real\_name  
 })  
 else:  
 return R.fail("该记录不存在")  
  
  
@user.route("/list", methods=['POST'])  
def user\_list():  
 """  
 分页查询用户列表  
 :return:  
 """  
 form = BasePageForm()  
 form.validate\_for\_api()  
 # 可通过form.data获取所有提交参数  
 # 可通过form.pageNum.data获取pageNum  
 # 可通过form.pageSize.data获取pageSize  
 user\_obj=User.query.filter().paginate(form.pageNum.data, form.pageSize.data,False)  
 # print(user\_obj.page) # 当前页码-从1开始  
 # print(user\_obj.per\_page) # 每页大小  
 # print(user\_obj.total) # 总记录数  
 # print(user\_obj.items) # 数据集  
 rows = []  
 for u in user\_obj.items:  
 rows.append({  
 "id": u.id,  
 "userName": u.user\_name,  
 "realName": u.real\_name  
 })  
 return R.data({  
 "recordCount": user\_obj.total,  
 "totalPage": int((user\_obj.total-1)/user\_obj.per\_page)+1,  
 "pageSize": user\_obj.per\_page,  
 "pageNum": user\_obj.page,  
 "rows": rows  
 })  
  
  
@user.route("/save", methods=['POST'])  
def user\_save():  
 """  
 添加用户  
 :return:  
 """  
 form = UserForm()  
 form.validate\_for\_api()  
 # 可通过form.data获取所有提交参数  
 # print(form.data)  
 u = User()  
 u.user\_name = form.data.get("userName")  
 u.real\_name = form.data.get("realName")  
 u.password = form.data.get("password")  
 db.session.add(u)  
 db.session.commit()  
 return R.success("添加用户成功")  
  
  
@user.route("/update", methods=['POST'])  
def user\_update():  
 """  
 修改用户  
 :return:  
 """  
 form = UserForm()  
 form.validate\_for\_api()  
 # 可通过form.data获取所有提交参数  
 # print(form.data)  
 User.query.filter\_by(id=form.id.data).update({  
 User.user\_name: form.userName.data,  
 User.real\_name: form.realName.data,  
 User.password: form.password.data  
 })  
 db.session.commit()  
 return R.success("修改用户成功")  
  
  
@user.route("/delete", methods=['POST'])  
def user\_delete():  
 """  
 删除用户  
 :return:  
 """  
 form = IdsForm()  
 form.validate\_for\_api()  
 # 可通过form.data获取所有提交参数  
 # print(form.data)  
 User.query.filter(User.id.in\_(form.ids.data)).delete()  
 db.session.commit()  
 return R.success("删除用户成功")

# 运行Flask服务

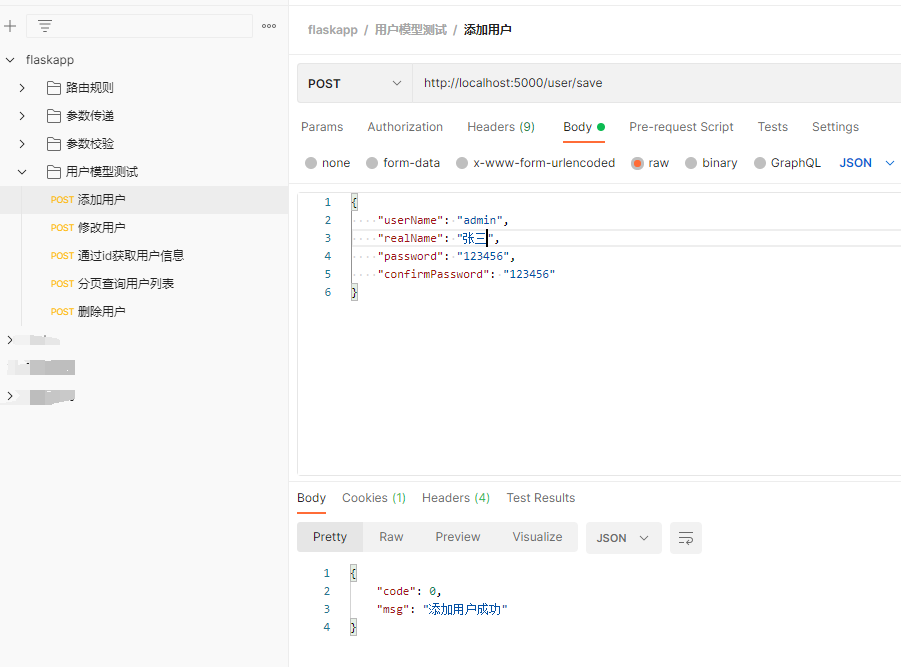
flask run

# 测试前重新建表

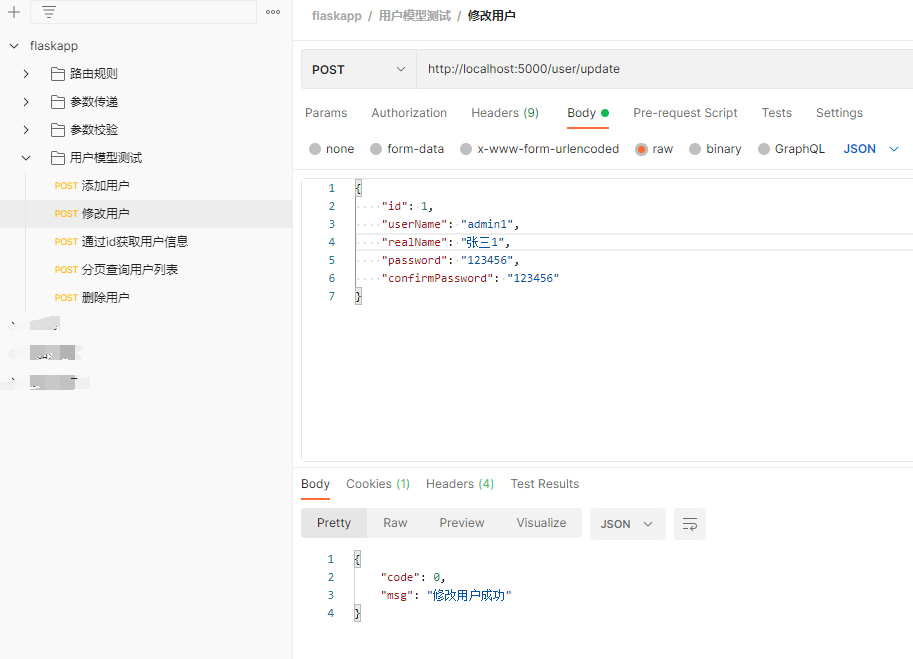
drop table if exists t\_user;  
CREATE TABLE `t\_user` (  
 `id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT '主键',  
 `user\_name` varchar(32) NOT NULL COMMENT '用户名',  
 `real\_name` varchar(32) NOT NULL COMMENT '姓名',  
 `password` varchar(64) NOT NULL COMMENT '密码',  
 `create\_time` datetime DEFAULT NULL COMMENT '创建时间',  
 `update\_time` datetime DEFAULT NULL COMMENT '更新时间',  
 `is\_deleted` tinyint(11) DEFAULT NULL COMMENT '逻辑删除:0=未删除,1=删除',  
 PRIMARY KEY (`id`),  
 UNIQUE KEY `user\_name` (`user\_name`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='用户';

# 使用Postman接口测试工具访问

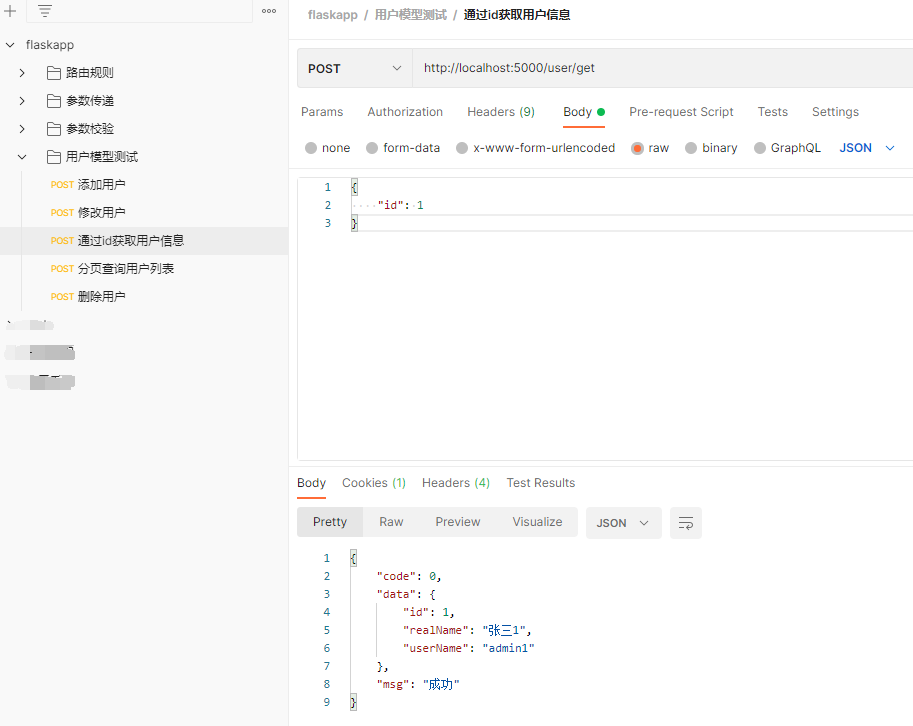
## 访问/user/save添加用户



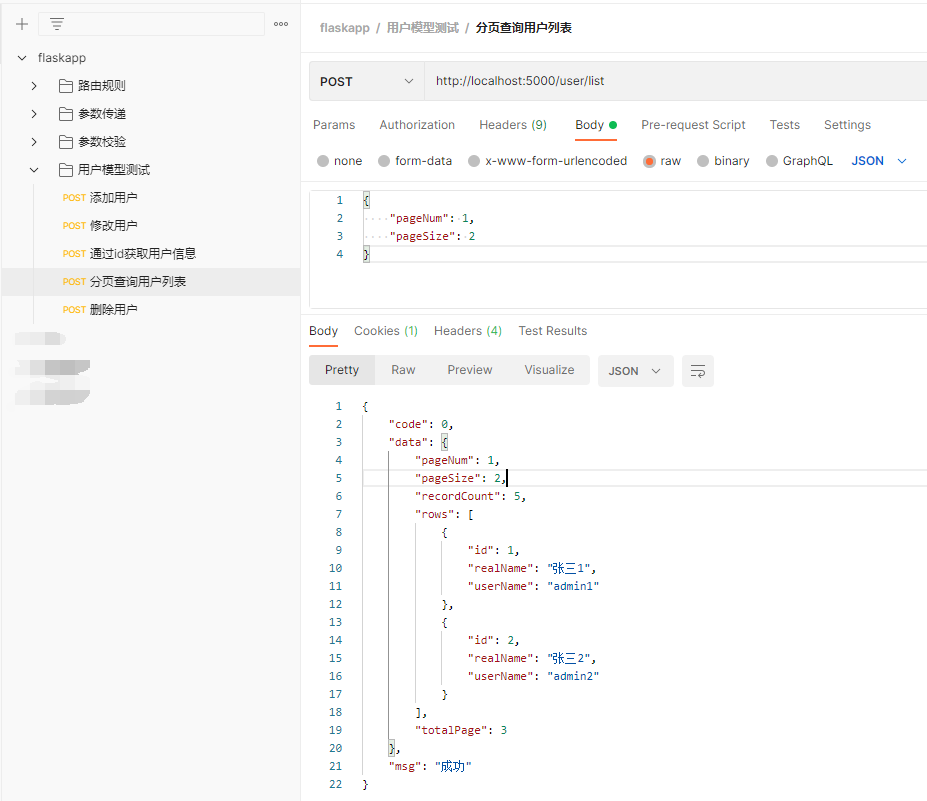
## 访问/user/update修改用户



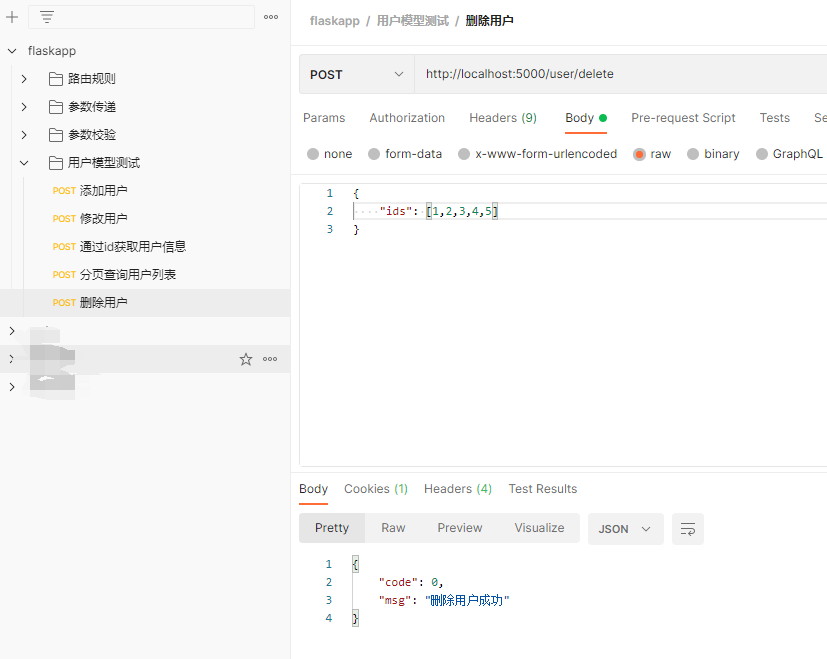
## 访问/user/get通过id获取用户信息



## 访问/user/list分页查询用户列表



## 访问/user/delete删除用户



# postman导出文件

文件，去掉.txt

