

mysql-flask-vue-schoolProjectStudentNet

1. 项目简介

学生校园网

vue+flask 前后分离式

介绍思路和整体要求 该项目为针对校园学生的网站，设计学生共同关注的信息服务及论坛。 业务场景：描述相关的真实企业业务背景。从真实场景中，适当简化或者提炼出适合项目场景 学生校园网是以全国各类学校为信息节点的校园社区平台，关注学生创业、就业、生活、情感、心理、学习、留学。提供服务及论坛。

功能性需求

- 1.实现基本公共信息服务；
- 2.实现学生论坛服务；
- 3.实现学生二手市场服务；

开发语言

javascript

html

python

mysql

2. 项目结构

- 目录

```
└─back 后端项目
| |
| | myapp.py 启动程序
| |
| | └─app 项目代码文件夹
| | | config.py 配置文件
| | | models.py 模型
| | | └─__init__.py 构造文件
| | |
| | | └─api 接口文件夹
| | | | communityApi.py 用户交流平台接口
| | | | loginRegistApi.py 登录注册接口
| | | | productApi.py 二手市场接口
| | | | └─__init__.py 构造文件
| | |
| | | └─main 主文件夹
| | | | views.py 视图页面
| | | | └─__init__.py 构造文件
| |
| | └─venv 虚拟环境
|
└─front
| communityHt.html 评论页面
| cs.html 测试页面
| login.html 登录页面
| lunIndex.html 论坛主页
| myZhuye.html 个人主页
| productHt.html 二手市场
| regist.html 注册
| single.html 页面
| single1.html 页面
| single2.html 页面
| single3.html 页面
| style2.css
|
| └─CS
| | chuan.html
| | communityHt.html
| | index.html
| | login.html
| | productHt.html
| | regist.html
```

3. 数据库配置

1. 账号密码配置

```
mysql -u root -p
```

账号: root

密码: 123456


```
# 获取数据
@api.route('/todo/api/communityApi', methods=['GET'])
# 添加数据
@api.route('/todo/api/addCommunityApi', methods=['POST'])
```

二手市场接口

```
# 获取数据
@api.route('/todo/api/Productions', methods=['GET'])
# 添加数据
@api.route('/todo/api/addProduction', methods=['POST'])
# 删除数据
@api.route('/todo/api/deleteProduction', methods=['POST'])
```

登录注册接口

```
# 获取登录状态
@api.route('/todo/api/getLoginApi', methods=['GET'])
# 注册接口
@api.route('/todo/api/addRegistApi', methods=['POST'])
# 登录接口
@api.route('/todo/api/loginApi', methods=['GET', 'POST'])
```

5. 后端部分说明

myapp.py

```
# 屏蔽JINJA2 防止和VUE冲突
app.jinja_env.variable_start_string = '{{'
app.jinja_env.variable_end_string = '}}'
```

app/models.py

```
# 将数据库数据转换为JSON格式
class JSONEncoder(_JSONEncoder):
    def default(self, o):
        if hasattr(o, 'keys') and hasattr(o, '__getitem__'):
            return dict(o)
        if isinstance(o, date):
            return o.strftime('%Y-%m-%d %H:%M:%S')
        return json.JSONEncoder.default(self, o)

# 重构Flask
class Flask(_Flask):
    json_encoder = JSONEncoder

# 设置用户模型
class User(db.Model, UserMixin):
    __tablename__ = 'users'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(64))
```

```
password = db.Column(db.String(128))
# 判断登录状态
def is_authenticated(self):
    return True
```

1. 从数据库中获取数据然后转换为JSON格式

```
# 将数据发送到URL地址中
@api.route('/todo/api/Productions', methods=['GET'])
def getTasks():
    # 获取Product表中的所有数据
    test_data = Product.query.all()
    # 将数据库的数据转换为JSON格式
    Production = json.loads(json.dumps(test_data, cls=JSONEncoder))
    # 将数据返回到URL地址中
    return jsonify({'productH': Production})
```

2. 从前端获取数据 然后 添加到数据库中

```
# 获取数据 然后 添加到市场模型
tas = Product(id=Production[-1]['id'] + 1, name=name,
pname=request.json.get('pname', ""), description=request.json.get('description',
""), price=request.json.get('price', ""), )
# 添加数据到数据库中
db.session.add(tas)
db.session.commit()
# 将数据添加到json列表中
```

3. 从前端获取到需要删除的数据 然后删除该数据

```
# 获取前端中需要删除的数据
task_id = request.json['id']
for task in Production:
    if task['id'] == task_id:
        Production.remove(task)
    # 找到该条数据
    test_data = Product.query.filter(Product.id == task_id).first()
    # 删除该条数据
    db.session.delete(test_data)
    db.session.commit()
    return jsonify({'productH': Production})
```

4. 判断当前登录状态

```
loginnName = [{'lname': '游客'}]
# 如果已经登录, 那么获取登录用户名, 否则默认
if current_user.is_authenticated:
    loginnName[0]['lname'] = current_user.get_id()
```

判断从前端获取到的用户名和密码是否正确

```
# 登录接口
@api.route('/todo/api/loginApi', methods=['GET', 'POST'])
def login():
```

```

if request.method == 'POST':
    # 获取前端数据
    user_id = request.json.get('name')
    user_pw = request.json.get('password')
    # 将数据给到query_user方法找到该条记录
    user = query_user(user_id)
    # 如果user不为空且密码正确
    if user is not None and user_pw == user['password']:
        print('信息-----成功登陆-----')
        curr_user = User()
        curr_user.id = user_id
        # 通过Flask-Login的login_user方法登录用户
        login_user(curr_user)
        return jsonify({'code': 200, 'name': user_id})

# GET 请求
return 'cs %s' % current_user.get_id()

def query_user(user_name):
    for user in json.loads(json.dumps(User.query.all(), cls=JSONEncoder)):
        if user_name == user['name']:
            return user

```

6. 前端部分说明

获取到后端数据并打开网页是默认渲染

```

mounted: function() {
    this.gett()
},
methods: {
    gett: function() {
        var self = this;
        //在编译后即调用API接口取得服务器端数据
        self.$http.get('/todo/api/Productions').then(function(res) {
            self.productH = res.data.productH;
        });
    },
}
}

```

将前端获取到的数据返回给后端

```

// HTML
<p>商品名称: <input type="text" v-model="new_prod.pname"></p>
<p>商品描述: <input type="text" v-model="new_prod.description"></p>
<p>商品价格: <input type="number" v-model="new_prod.price"></p>
<button @click="addTask">确认发表商品</button>

```

```

// JS
<script>
    var vue = new Vue({
        el: '#app',
        data: {

```

```
        productH: [],
        new_prod: {
            name: '',
            pname: '',
            description: '',
            price: 0
        }
    },
    methods: {
        addTask: function() {
            var self = this;
            self.$http.post('/todo/api/addProduction', {
                pname: self.new_prod.pname,
                description: self.new_prod.description,
                price: self.new_prod.price
            }).then(function(res) {
                self.productH = res.data.productH;
            });
        },
    },
    })
</script>
```

7. 开发分工

组长->冯振熙



• 工作

后端项目框架搭建、配置文件编写、数据库配置、接口编写等

前端 vue-resource + axios 的数据接收和发送的开发 以及 登录、注册、二手市场、评论页面功能开发

组员->何梓浩



. 工作

后端

前端 个人主页开发、前端模板开发、个人主页页面设计、CSS样式开发

组员->岑梓峰



. 工作

后端

前端 论坛页面开发、前端模板开发、论坛模板设计

组员->潘康毅



. 工作

后端

前端 数据收集，文字图片素材提供

组员->刘展峰



• 工作

后端

前端 数据收集，文字图片素材提供