


# 超详细Django+vue+vscode前后端分离搭建

原创 唐僧骑白马 已于 2023-04-16 23:24:28 修改 阅读量8.8k 收藏 199 点赞数 29 版权

分类专栏: Django 文章标签: django vue.js vscode

 Django 专栏收录该内容

6 订阅 24 篇文章 订阅专栏

文章目录

- 一、Django后端搭建
    - 1.1 创建项目和app
    - 1.2 注册app
    - 1.3 运行项目
    - 1.4 配置mysql数据库
    - 1.5 创建数据库类
    - 1.6 使用Django后台进行数据管理
  - 2、Django rest framework配置
    - 2.1 序列化
    - 2.2 添加视图
    - 2.3 添加路由
    - 2.4 在项目根目录下的urls中加入如下代码
    - 2.5 api测试
    - 2.6 筛选和搜索功能配置
    - 2.7 分页设置
  - 3、自动生成api文档
- 二、vue前端搭建
    - 1、前端工具及框架
    - 2、在Django项目的根目录下创建前端文件
    - 3、修改src/components/HelloWorld.vue中的代码如下
    - 4、前后端联调
    - 5、前端打包
- 三、总结

一、Django后端搭建

1.1 创建项目和app

```
1 | django-admin startproject tman
2 | python manage.py startapp tadmin
```



## 1.2 注册app

```
1 INSTALLED_APPS = [  
2     'tadmin',  
3 ]
```

## 1.3 运行项目

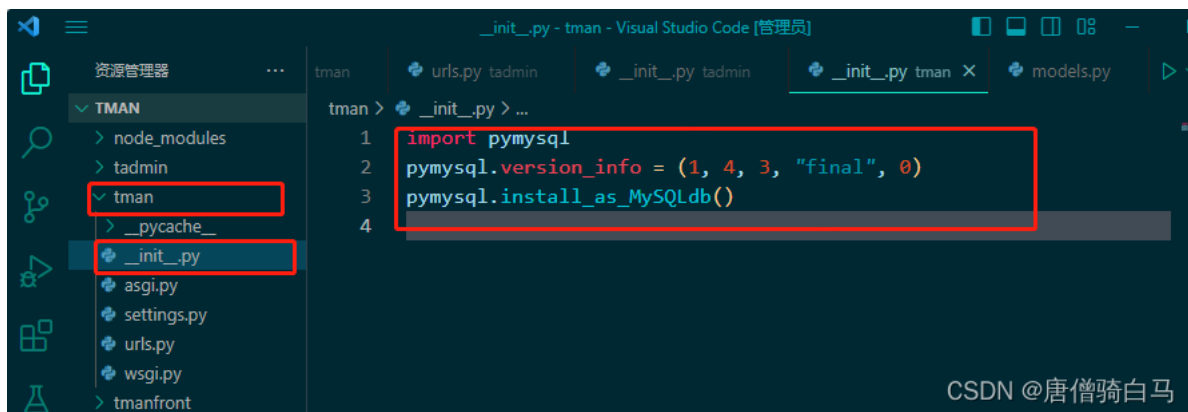
```
1 python manage.py runserver
```

## 1.4 配置mysql数据库

```
1 DATABASES = {  
2     'default': {  
3         'ENGINE': 'django.db.backends.mysql',  
4         'NAME': 'taskmanage',  
5         'USER': 'root',  
6         'PASSWORD': '密码',  
7         'HOST': '192.168.75.132',  
8         'PORT': '3306',  
9     }  
10 }
```

在项目tman项目下的init.py中加入如下代码

```
1 pip install pymysql  
  
1 import pymysql  
2 pymysql.version_info = (1, 4, 3, "final", 0)  
3 pymysql.install_as_MySQLdb()
```



## 1.5 创建数据库类

在tadmin的model.py中加入如下代码

```
1 from django.db import models  
2  
3  
4 class UserInfo(models.Model):  
5     username = models.CharField('用户名', max_length=128)  
6     password = models.CharField('密码', max_length=128)  
7  
8     class Meta:  
9         verbose_name = '用户信息'  
10        verbose_name_plural = '用户信息'  
11  
12    def __str__(self):  
13        return self.username
```

执行如下命令创建数据库

```
1 python manage.py makemigrations  
2 python manage.py migrate
```

1.6 使用Django后台进行数据管理

在tadmin应用目录下加入如下代码

```
1 | from django.contrib import admin
2 | from tadmin.models import UserInfo
3 |
4 | admin.site.site_header = '任务管理系统'
5 |
6 |
7 | class UserInfoAdmin(admin.ModelAdmin):
8 |     list_display = ('id', 'username', 'password',)
9 |     list_display_links = ('username',)
10 |    list_per_page = 50
11 |
12 |
13 | admin.site.register(UserInfo, UserInfoAdmin)
```

创建后台管理员用户

```
1 | python manage.py createsuperuser
```



2、Django rest framework配置

```
1 | pip install djangorestframework
2 | # 暂时不装也可以
3 | pip install markdown
4 | # 用于数据筛选
5 | pip install django-filter
```

在settings中注册framework

```
1 | INSTALLED_APPS = [
2 |     'rest_framework',
3 |     'django_filters',
4 | ]
```

2.1 序列化

在app目录下创建serializer.py，添加如下代码

```
1 | from tadmin.models import UserInfo
2 | from rest_framework import serializers
3 |
4 |
5 | class UserInfoSerializer(serializers.ModelSerializer):
6 |     class Meta:
7 |         model = UserInfo
8 |         fields = "__all__"
```

2.2 添加视图

在app目录下的view.py中加入如下代码：

```
1 from rest_framework.viewsets import ModelViewSet
2 from tadmin.models import UserInfo
3 from tadmin.serializer import UserInfoSerializer
4 from tadmin.filter import UserInfoFilter
5 from django_filters.rest_framework import DjangoFilterBackend
6
7
8 class UserInfoViewSet(ModelViewSet):
9     queryset = UserInfo.objects.all()
10    serializer_class = UserInfoSerializer
11
12    filter_class = UserInfoFilter
13    filter_fields = ['username',]
14    search_fields = ('username',)
```

### 2.3 添加路由

在app目录下创建urls.py文件：

```
1 from django.urls import path, include
2 from rest_framework.routers import DefaultRouter
3 from tadmin.views import UserInfoViewSet
4
5 router = DefaultRouter()
6 router.register('UserInfo', UserInfoViewSet, basename='UserInfo')
7
8 urlpatterns = [
9 ]
10
11 urlpatterns += [
12     path('', include(router.urls)),
13 ]
```

### 2.4 在项目根目录下的urls中加入如下代码

```
1 from django.contrib import admin
2 from django.urls import path, include
3
4 urlpatterns = [
5     path('admin/', admin.site.urls),
6     path('api/v1/', include('tadmin.urls')),
7 ]
```

### 2.5 api测试

<http://127.0.0.1:8000/api/v1/UserInfo/>

Django REST frameworkadmin

Api Root / User Info List

# User Info List

Filters OPTIONS GET

GET /api/v1/UserInfo/

HTTP 200 OK  
Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{
  "count": 4,
  "next": null,
  "previous": null,
  "results": [
    {
      "id": 1,
      "username": "test1",
      "password": "test1"
    },
    {
      "id": 2,
      "username": "test2",
      "password": "test2"
    },
    {
      "id": 4,
      "username": "test4",
      "password": "test4"
    },
    {
      "id": 5,
      "username": "test5",
      "password": "test5"
    }
  ]
}
```

Raw data 唐僧骑白马

## 2.6 筛选和搜索功能配置

在app根目录下创建filter.py文件

```
1 from django_filters import FilterSet, filters
2 from tadmin.models import UserInfo
3
4
5 class UserInfoFilter(FilterSet):
6     name = filters.CharFilter(field_name='username', lookup_expr='icontains')
7
8     class Meta:
9         model = UserInfo
10        fields = ('username',)
```

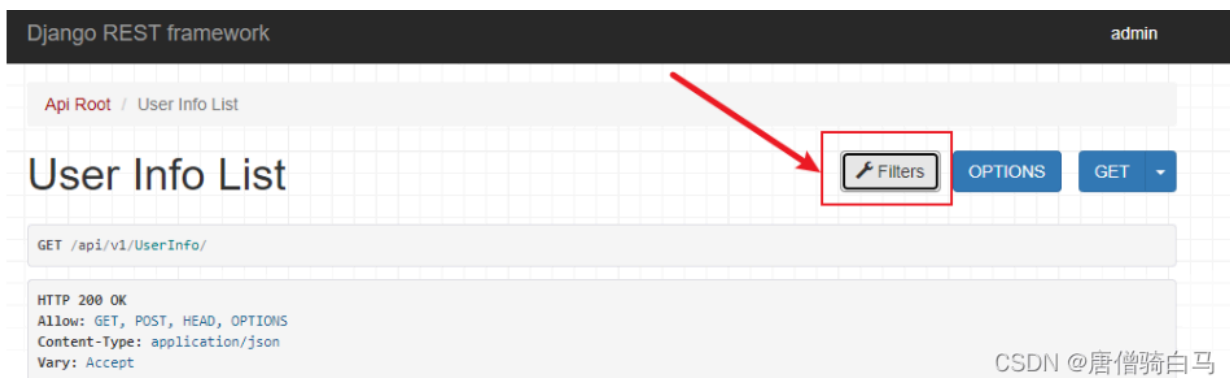
修改app目录下的view文件： [在这里插入代码片](#)

```
1 from django.shortcuts import render
2
3 from rest_framework.viewsets import ModelViewSet
4 from tadmin.models import UserInfo
5 from tadmin.serializer import UserInfoSerializer
6 from tadmin.filter import UserInfoFilter
7 from django_filters.rest_framework import DjangoFilterBackend
8
9
10 class UserInfoViewSet(ModelViewSet):
11     queryset = UserInfo.objects.all()
12     serializer_class = UserInfoSerializer
13
14     filter_class = UserInfoFilter
15     filter_fields = ['username']
16     search_fields = ('username',)
```

在settings中注册django\_filters:

```
1 INSTALLED_APPS = [  
2     'django_filters',  
3 ]  
4  
5 # REST_FRAMEWORK增加全局过滤配置  
6 REST_FRAMEWORK = {  
7     'DEFAULT_FILTER_BACKENDS': [  
8         'django_filters.rest_framework.DjangoFilterBackend',  
9         'rest_framework.filters.SearchFilter',  
10    ],  
11 }  
12 # 如果可以实现模糊查询, 则以下语句可省略  
13 FILTERS_DEFAULT_LOOKUP_EXPR = 'icontains'
```

Django Rest Framework页面出现Filters图标说明配置成功



## 2.7 分页设置

在settings.py中做如下修改

```
1 # REST_FRAMEWORK增加全局过滤配置  
2 REST_FRAMEWORK = {  
3     # 设置分页  
4     'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',  
5     'PAGE_SIZE': 10,  
6 }
```

```
# REST_FRAMEWORK增加全局过滤配置
```

```
REST_FRAMEWORK = {  
    'DEFAULT_FILTER_BACKENDS': [  
        'django_filters.rest_framework.DjangoFilterBackend',  
        'rest_framework.filters.SearchFilter',  
    ],  
    # 设置分页  
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',  
    'PAGE_SIZE': 5,  
}
```

CSDN @唐僧骑白马

Api Root / User Info List

## User Info List

Filters OPTIONS GET

« 1 2 »

GET /api/v1/UserInfo/?page=2

HTTP 200 OK  
Allow: GET, POST, HEAD, OPTIONS  
Content-Type: application/json  
Vary: Accept

```
{  
  "count": 8,  
  "next": null,  
  "previous": "http://127.0.0.1:8000/api/v1/UserInfo/",  
  "results": [  
    {  
      "id": 7,  
      "username": "test7",  
      "password": "test7"  
    },  
    {  
      "id": 8,  
      "username": "test7",  
      "password": "test7"  
    },  
    {  
      "id": 9,  
      "username": "test8",  
      "password": "test8"  
    }  
  ]  
}
```

CSDN @唐僧骑白马

### 3、自动生成api文档

```
1 | pip install drf-yasg
```

在项目文件夹urls.py中做如下修改

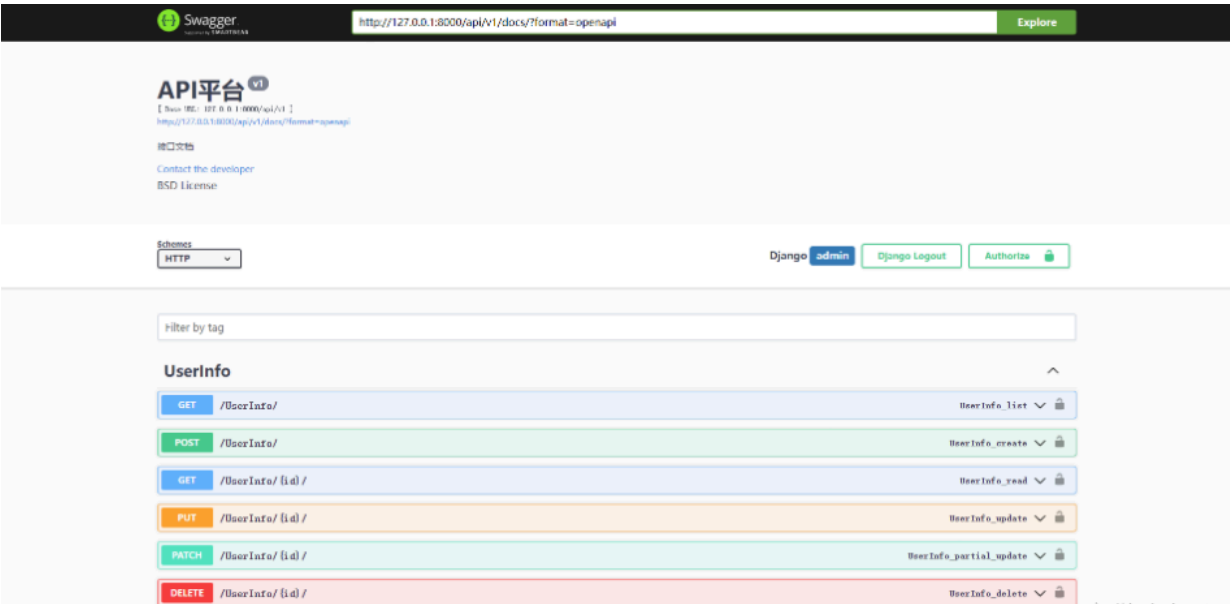
```
1 | INSTALLED_APPS = [  
2 |     'drf_yasg', # swagger  
3 | ]
```

在app的urls.py中做如下修改

```
1 | from drf_yasg.views import get_schema_view  
2 | from drf_yasg import openapi  
3 |  
4 | schema_view = get_schema_view(  
5 |     openapi.Info(  
6 |         title="API平台",  
7 |         default_version="v1",  
8 |         description="接口文档",  
9 |         terms_of_service="",  
10 |         contact=openapi.Contact(email='2495128088@qq.com'),  
11 |         license=openapi.License(name="BSD License"),  
12 |     ),  
13 |     public=True  
14 | )  
15 |  
16 |
```

```
16 router = DefaultRouter()
17 router.register('UserInfo', UserInfoViewSet, basename='UserInfo')
18
19 urlpatterns = [
20     path('docs/', schema_view.with_ui('swagger',cache_timeout=0), name='schema-swagger-ui'),
21 ]
```

文档查看文档是否成功， <http://127.0.0.1:8000/api/v1/docs/>



CSDN @唐僧骑白马

二、vue前端搭建

1、前端工具及框架

- node.js
- npm
- vue3
- axios
- Element plus
- 前端开发工具：VS Code

2、在Django项目的根目录下创建前端文件

```
1 | npm init webpack tmanfront
```

最终的文件目录如下：





### 3、修改src/components/HelloWorld.vue中的代码如下

```
1 <template>
2   <div class="hello">
3     <h1>{{ msg }}</h1>
4     <ul>
5       <li v-for="(user,index) in users" :key="index" style="display: block;">
6         {{ index }}--{{ user.username }}--{{ user.password }}
7       </li>
8     </ul>
9     <form action="">
10      用户名: <input type="text" placeholder="user name" v-model="inputUser.username"><br>
11
12      密码: <input type="text" placeholder="user password" v-model="inputUser.password"><br>
13      <button type="submit" @click="userSubmit()">提交</button>
14    </form>
15  </div>
16 </template>
17
18 <script>
19 import { getUsers,postUser } from '../api/api.js';
20 export default {
21   name:'hellouser',
22   data () {
23     return {
24       msg:'Welcome to Your Vue.js App',
25       users:[
26         {username:'test1',password:'test1'},
27         {username:'test2',password:'test2'}
28       ],
29       inputUser:{
30         "username":"","
31         "password":"","
32       }
33     }
34   },
35   methods:{
36     loadUsers(){},
37     userSubmit(){},
38   },
39   created: function(){
40     this.loadUsers()
41   }
42 }
```

```
42 |     }  
43 | }  
    </script>
```

启动前端项目，浏览器访问127.0.0.1:8080，可以看到刚写的页面已经更新上去了



## Welcome to Your Vue.js App

0--test1--test1  
1--test2--test2

用户名:

密码:

CSDN @唐僧骑白马

#### 4、前后端联调

利用django-cors-headers模块 [解决跨域问题](#)

```
1 | pip install django-cors-headers
```

然后在项目settings.py中添加该模块：

```
1 | INSTALLED_APPS = [  
2 |     'corsheaders',  
3 | ]  
4 |  
5 | MIDDLEWARE = [  
6 |     'corsheaders.middleware.CorsMiddleware', # 需注意与其他中间件顺序, 这里放在最前面即可  
7 |     ...  
8 | ]  
9 | # 支持跨域配置开始  
10 | CORS_ORIGIN_ALLOW_ALL = True  
11 | CORS_ALLOW_CREDENTIALS = True
```

后端部分告于段落，接下来需要补充一下前端的逻辑，[Vue框架](#) 现在一般都用axios模块进行网络请求，这里沿用这种方式，下面是在前端项目中操作：

首先命令行安装axios模块，如果没有安装cnpm就还是用npm安装：

```
1 | cnpm install axios  
2 | 或者  
3 | npm install axios
```

为了方便管理api请求的各种逻辑，在前端项目的src目录下创建api目录，然后创建api.js和index.js文件。index.js文件是对axios做配置：

[/src/api/index.js](#)

```
1 | import Vue from 'vue'  
2 | import Axios from 'axios'  
3 |  
4 | const axiosInstance=Axios.create({  
5 |     withCredentials:true  
6 | })  
7 |  
8 | axiosInstance.interceptors.request.use((config)=>{  
- |
```

```

9   config.headers['X-Requested-With'] = 'XMLHttpRequest'
10  const regex = /.csrftoken=([^;.]*).*$/
11  config.headers['X-CSRFToken'] = document.cookie.match(regex) === null ? null : document.cookie.match(regex)
12  return config
13 })
14
15 axiosInstance.interceptors.response.use(
16   response=>{
17     return response
18   },
19   error=>{
20     return Promise.reject(error)
21   }
22 )
23
24 Vue.prototype.axios=axiosInstance
25
26 export default axiosInstance

```

api.js文件是对后端进行请求，可以看到，获取books列表和添加一本书book各对应于一个请求：

```

1  import axiosInstance from "../index";
2
3  const axios = axiosInstance
4  export const getUsers = () => { return axios.get(`http://127.0.0.1:8000/api/v1/UserInfo/`) }
5
6  export const postUser = (username, password) => { return axios.post(`http://127.0.0.1:8000/api/v1/UserInfo/`, {

```

然后更新HelloWorld.vue中的处理逻辑：

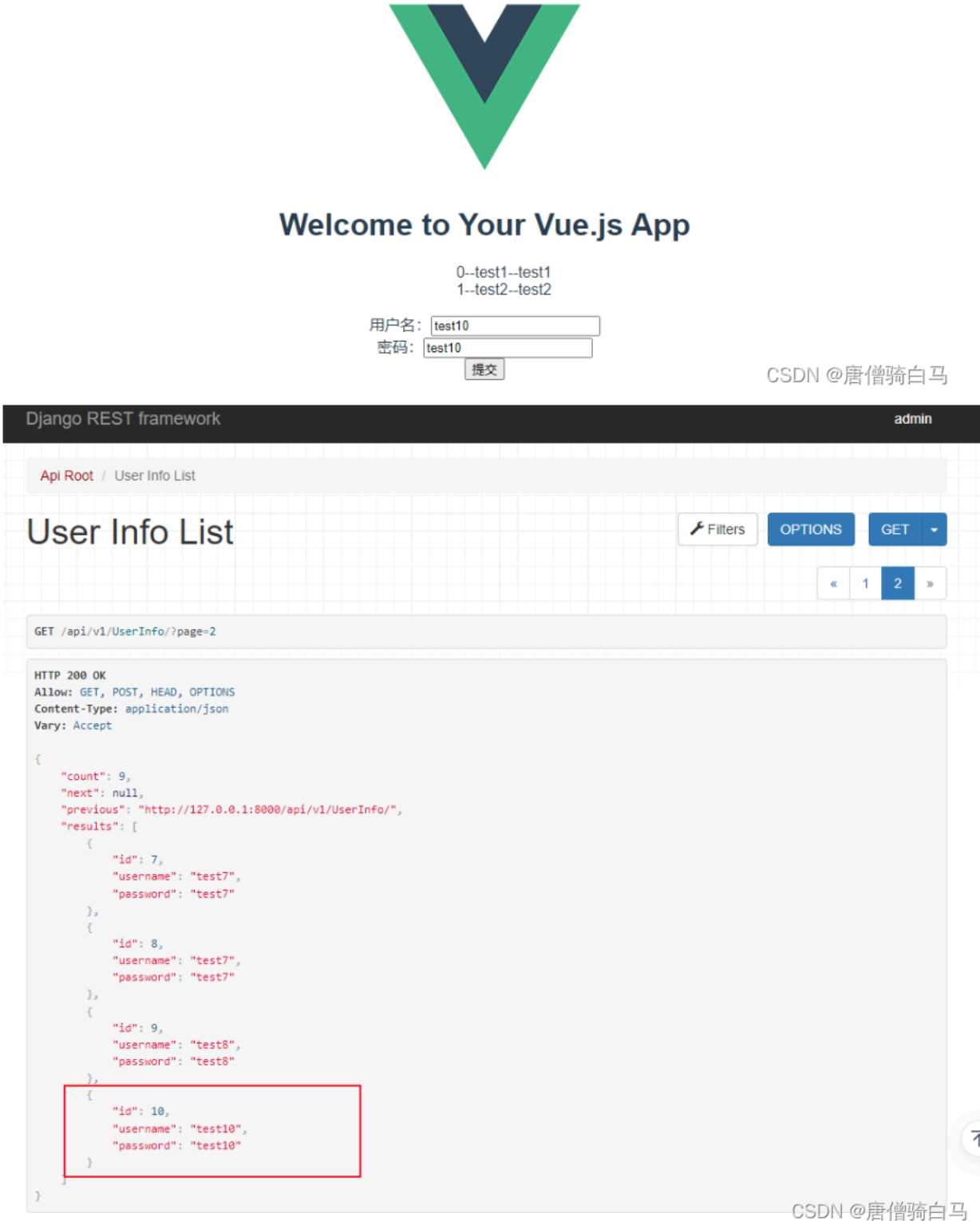
```

1  <template>
2    <div class="hello">
3      <h1>{{ msg }}</h1>
4      <ul>
5        <li v-for="(user,index) in users" :key="index" style="display: block;">
6          {{ index }}--{{ user.username }}--{{ user.password }}
7        </li>
8      </ul>
9      <form action="">
10       用户名: <input type="text" placeholder="user name" v-model="inputUser.username"><br>
11
12       密码: <input type="text" placeholder="user password" v-model="inputUser.password"><br>
13       <button type="submit" @click="userSubmit()">提交</button>
14     </form>
15   </div>
16 </template>
17
18 <script>
19 import { getUsers,postUser } from '../api/api.js';
20 export default {
21   name: 'hellouser',
22   data () {
23     return {
24       msg: 'Welcome to Your Vue.js App',
25       users:[
26         {username: 'test1',password: 'test1'},
27         {username: 'test2',password: 'test2'}
28       ],
29       inputUser:{
30         "username": "",
31         "password": "",
32       }
33     }
34   },
35   methods:{
36     loadUsers(){
37       getUsers().then(response=>{
38         this.users=response.data
39       })
40     }

```

```
41     },
42     userSubmit(){
43         postUser(this.inputUser.username,this.inputUser.password).then(response=>{
44             console.log(response)
45             this.loadUsers()
46         })
47     }
48 },
49 created: function(){
50     this.loadUsers()
51 }
52 }
</script>
```

至此，一个简单的Django+vue前后端分离项目就已搭建完成，测试添加数据成功

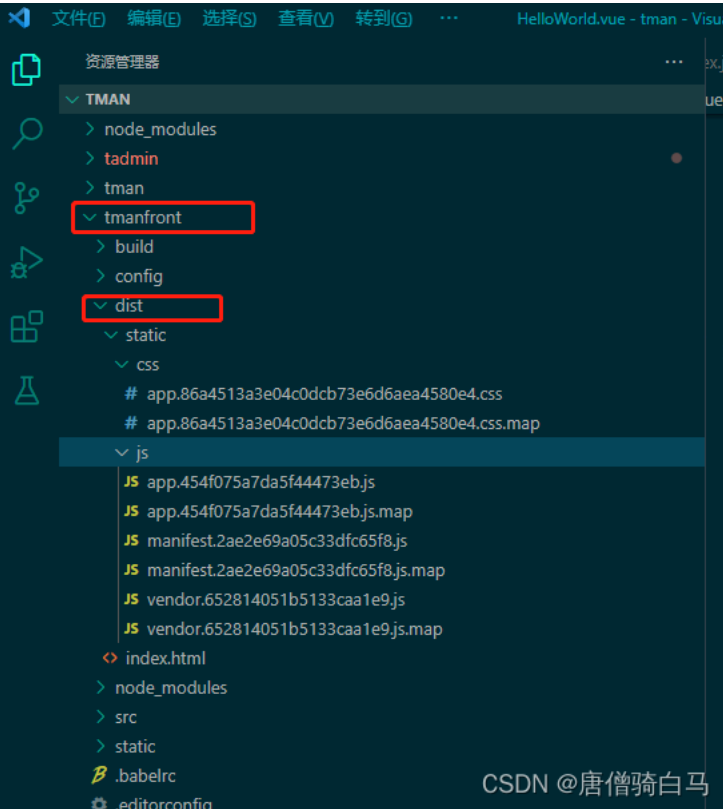


可以看到，列表里面的数据是从后端读取到的，同时前端的提交数据库也能有对应的操作，所以前后端至此是打通了。

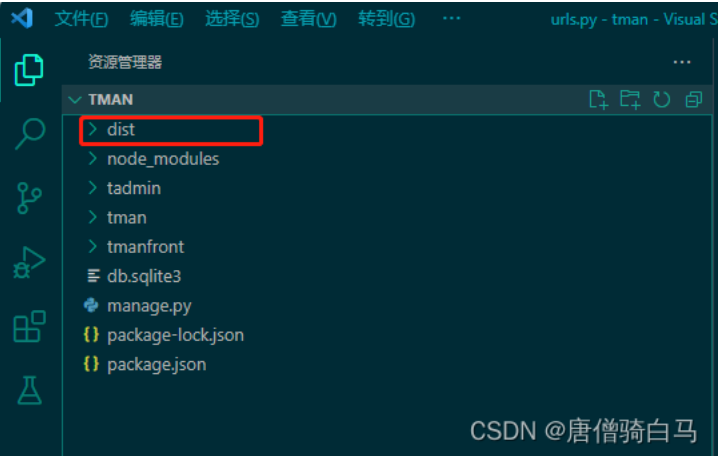
5、前端打包

现阶段是前后端分开开发，但是当最后要用的时候，还需要把代码合在一起。  
首先对前端项目进行打包，这里用Vue的自动打包，进入前端的根目录下：

```
1 | npm run build
```



可以看到前端项目中多出了一个 `dist` 文件夹，这个就是前端文件的打包结果。需要把 `dist` 文件夹复制到 `tman` 项目文件夹中



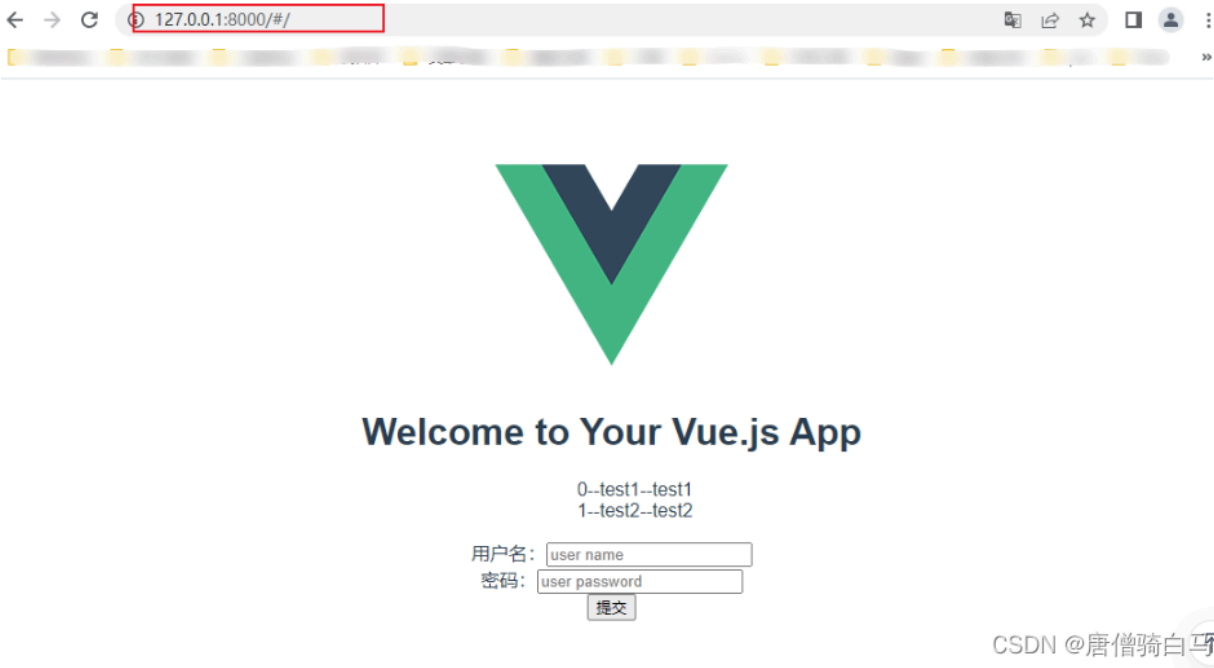
然后对settings.py文件进行相应的修改，其实就是帮django指定模版文件和静态文件的搜索地址：

```
1 | TEMPLATES = [  
2 |     {  
3 |         'BACKEND': 'django.template.backends.django.DjangoTemplates',  
4 |         'DIRS': [os.path.join(BASE_DIR, 'dist')],  
5 |         ...  
6 |     },  
7 | ]  
8 | ...  
9 | STATICFILES_DIRS = [  
10 |     os.path.join(BASE_DIR, 'dist/static'),  
11 | ]
```

最后在项目根urls.py文件中配置一下入口html文件的对应路由：

```
1 from django.views.generic.base import TemplateView
2 urlpatterns = [
3     path('', TemplateView.as_view(template_name='index.html'))
4 ]
```

重新启动项目，这次用浏览器访问 127.0.0.1:8000，即django服务的对应端口即可。  
可以看到，项目的交互是正常的，符合我们的预期。



三、总结

本文以一个非常简单的demo为例，介绍了利用 `django+drf+vue` 的前后端分离开发模式，基本可以算是手把手入门。有了这个小demo之后，不管是前端页面还是后端功能，都可以做相应的扩展，从而开发出更加复杂使用的网站。



高效运维之美  
分享运维知识

 微信公众号 >

显示推荐内容

 唐僧骑白马

29

11 条评论

 weixin\_66009678 热评 原来在下面，谢谢

写评论