文档：<https://cn.vuejs.org/v2/guide/>

v-model=同步属性 @click=点击事件 v-bind:=绑定属性(简写为，”:”)

v-for=”(i,index) in item” :key=”index” 循环

<body>

    <div id="app">

        <input type="text" v-model='username'>

        {{ uname }}

      </div>

</body>

</html>

<script>

var app = new Vue({

  el: '#app',//挂在点

  data: {//数据

    username: 'Hello Vue!'

  },

  components:{//注册组件，要先import

  },

  computed: {//计算属性

      uname(){

          return this.message

      },

  },

  methods: {//普通函数

  },

})

</script>

# 生命周期

beforeCreate：创建前

created：创建后

beforeMount：组件挂载前

mounted：组件挂载后

beforeUpdate：数据更新前

updated：数据更新后

beforeDestroy：vue对象销毁前

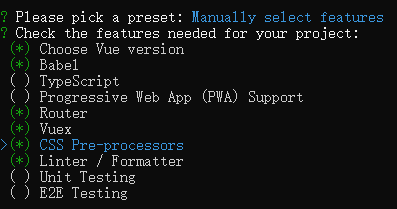
destroyed：vue对象销毁后

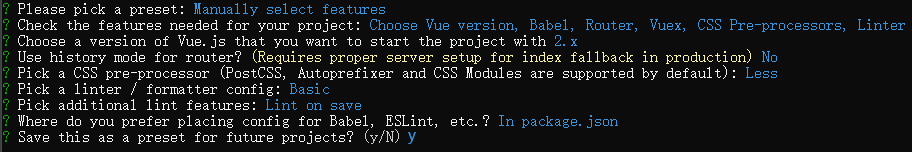
# Vue CLI脚手架

安装：一，要有Node.js

1. cnpm install -g @vue/cli (-g：全局安装)

创建：vue create 名称





# 引入公共样式

app.vue -> style 里：@import './assets/style/common.less';

# 组件

## 局部组件

import Vue from 'vue'

Vue.component('my-component-name',{

  template:'<p>哈哈</p>'

})

然后直接，<my-component-name/>就可

没有显示就新建vue.config.js，添加实时编译

module.exports={

    runtimeCompiler: true,

}

## 全局组件

1. 新建vue文件 2.导入组件

import xxx from '@/xxx/xxx'

1. 注册组件

components: {

    xxx,

  }

1. 使用组件

<xxx/>

## 组件复用

data必须是函数

data: {//数据

  //   username: 'Hello Vue!'

  // },

data() {

    return {

       active: 'home',

    };

# 模板语法

普通文本

<span>内容：{{ msg }}</span>

JavaScript表达式

{{ msg.split('').reverse().join('') }}//字符串反转

{{ result > 0 ? 'yes':'no' }}//三元运算

富文本

<p v-html="rawHtml"></p>

data() {

    return {

       rawHtml:'<p>哈哈</p>'

    };

},

## 指令

v-bind:=绑定属性(简写为，”:”)

<h1 :class="{'cla1':result>0}">设置单css的class</h1>

<h1 :class="[{'cla1':result>0},'cla2']">设置多css的class</h1>

<h1 :style="[{color:'#fff'},{border:'1px solid #fff'}]">设置单css的style</h1>

v-for=循环

v-for="(item,index) in item" :key="index" //列表

v-for="(item,key,index) in item" :key="index" //字典

v-text=内容 v-show=显示隐藏

v-model=双向绑定 v-on:click=点击事件(简写为，”@click”)

鼠标事件修饰符

阻止事件冒泡 @click.stop 阻止默认行为 @click.prevent

只触发一次 @click.once

键盘事件

监听回车键 @keyup.enter 监听删除/退格 @keyup.delete

修饰符 v-model.trim 过滤空白 v-model.number 转为数值型

# 过滤器

定义

function name(vaule){

    if(vaule){

        return vaule.substr(0,1)+'\*\*'

    }

    return vaule

}

export {

    name,

}

注册

Vue.filter('过滤器名',过滤器)

使用

      {{msg | 过滤器}}

      <div :id="raw | 过滤器"></div>

# 组件通信

## 父到子

父：<SightItem v-for="item in dataList" :key="item.id" :item="item"/>

子：export default {

    props:['item']

}

验证数据类型

props:{

    type:String,//类型

required:true,//必须传的

default:100,//默认值}

## 子到父

子组件触发自定义事件

this.$emit(‘myEvent’,可加参数)

父组件监听事件

<p @myEvent=’doSth’>

# 路由

vue-router官方路由管理器，vue一般已装好

安装： cnpm install vue-router -S

<http://xx/?name=ss> this.$route.query.name

{ path: '/home/:id',component: Home },

//{ path: '/home/:id/:id2/',component: Home },

[http://xx/1](http://xx/?name=ss) this.$route.params.id

监听路由

watch:{

        $route(to,from){

            console.log(to)    //去哪里

            console.log(from)  //从哪里来

            this.id=this.$route.params.id

        }

//也可监听data里的数据

    $id(to,from){

            console.log(to)    //更新后数据

            console.log(from)  //原来数据

        }

    },

## 动态路由及跳转

{

    path: '/home/:id/:id2', //路径

    name: 'home', //路由名称

component: Home //显示的组件

  },

通过给路由添加名称，可方便以后修改

<router-link to='{name:"home",params:{id:"11",id2:"12"},query:{name:"aa",name2:"bb"}}'>About</router-link>

路由名称name:"home"

路由动态参数params:{id:"11",id2:"12"}

查询参数query:{name:"aa",name2:"bb"}

JS中跳转:this.$route.push({name:’xxx’,params:{},query:{}})

后退:this.$route.go(-1)

替换(不能后退):this.$route.replace({name:’xxx’})

## 使用

安装后新建router.js

import Vue from 'vue'

import VueRouter from 'vue-router'

Vue.use(VueRouter)

const routes = [

  {

    path: '/about',

    //name: 'About',

    component: () => import(/\* webpackChunkName: "about" \*/ '../views/About.vue')

  }

]

const router = new VueRouter({

  routes

//routes:routes

})

export default router

在main.js引入

import router from './router'

new Vue({

  router, <-加在这

  store,

  render: h => h(App)

}).$mount('#app')

vue中添加显示位置，就可

<router-view></router-view>

详细使用

1. 设置路由规则

const routes = [

  {

    path: '/', //路径

    name: 'Home', //路由名称

component: Home //显示的组件

//component: () => import('../views/About.vue')

  },

]

1. 配置路由规则

import Vue from 'vue'

import VueRouter from 'vue-router'

Vue.use(VueRouter)

const routes = [

  {

    path: '/about',

    name: 'About',

    component: () => import(/\* webpackChunkName: "about" \*/ '../views/About.vue')

  }

]

const router = new VueRouter({

  routes

})

export default router

1. 设置路由切换后页面显示位置

<router-view></router-view>

1. 在模板中实现路由跳转

<router-link to="/about">About</router-link>

# axios异步请求库

和跨域并用

安装：进入项目目录 -> cnpm install axios -S(生产环境) 或 -D(开发环境)

请求方法的别名

axios.request(config) axios.get(url[, config]) axios.delete(url[, config])

axios.head(url[, config]) axios.options(url[, config]) axios.post(url[, data[, config]])

axios.put(url[, data[, config]]) axios.patch(url[, data[, config]])

get

getDataList(){

    axios.get('url/',{

        params:{

            is\_hot:1

        }

    }).then(({data}) => {

        console.log(' 返回数据处理')

    }).catch(function(err){

        console.log('错误处理')

    })

}

post

getDataList(){

    axios.post('url/',{is\_hot:1})

.then(({data}) => {

        console.log(' 返回数据处理')

    }).catch(function(err){

        console.log('错误处理')

    })

}

## axios请求响应拦截

场景：自定义请求头、携带上次cookie、加载时动画、统一错误处理

ajax.js

import axios from 'axios'

export const ajax=axios.create({ //创建实例

    headers:{//自定义请求头

        source:'h5',

        'Content-Type':'application/x-www-form-urlencoded'

    },

    // 携带上次cookie

withCredentials:true

//timeout:超时的毫秒

//data:post/put/patch请求的数据

//params:url中的参数

//responseType:默认为jso

})

ajax.interceptors.request.use(function (config){

    // 发送请求前做什么

    return config;

},function(error){

    // 错误处理

    return Promise.reject(error);

})

ajax.interceptors.response.use(function (response){

    // 对响应数据做什么

    return response;

},function(error){

    // 错误处理

    if(error.response.status === 401){

        window.alert('未登录')

    }

    return Promise.reject(error);

})

xxx.vue

要用的组件导入ajax即可

import {ajax} from '@/utils/ajax'

methods:{

    getDataList(){

        ajax.get(SightApis.sightListUrl,{

            params:{

                is\_top:1

            }

        }).then(({data}) => {

           this.dataList=data.objects

        })

    }

},

## PS

post请求后台拿不到数据 或 格式不正确 ，用transformRequest处理

export const ajax=axios.create({

    headers:{

        'source':'h5',

        'Content-Type':'application/x-www-form-urlencoded'

    },

    transformRequest:function(data,headers){//处理发送参数

        return qs.stringify(data)

    },

    withCredentials:true

})

# 跨域

## vue解决跨域

只需新建vue.config.js，修改后需重启

module.exports={ //Node.js里的模块

    devServer: { //devServer表示开发服务器

        proxy:{

            '/api': {

                target: 'http://localhost:8000',//目标接口

                changOrigin: true,  //允许跨域

                pathRewrite: {//url重写规则

                    '^/api': ''

                },

            '/test': {

                target: 'http://localhost:8000',//目标接口

                changOrigin: true,  //允许跨域

                pathRewrite: {//url重写规则

                    '^/test': ''

                },

            }

        }

    }

}

## Django解决跨域

安装：pip install django-cors-headers

配置setting.py：

INSTALLED\_APPS 加上 ‘corsheaders’

添加中间件MIDDLEWARE = [

'django.contrib.sessions.middleware.SessionMiddleware',

'corsheaders.middleware.CorsMiddleware', 加在这中间

'django.middleware.common.CommonMiddleware',]

添加白名单CORS\_ORIGIN\_WHITELIST = (

'http://127.0.0..1’//那些可访问

)

可选配置

CORS\_ALLOW\_HEADERS = (‘source’,’icode’)//允许那些请求头

CORS\_ALLOW\_CREDENTIALS = True//允许携带cookie

# vant

安装：cnpm install vant -S

文档：[https://vant-contrib.gitee.io/vant/#/zh-CN/home](https://vant-contrib.gitee.io/vant/" \l "/zh-CN/home)

**全局引入**：main.js

import Vant from 'vant'//vantUI组件库

import 'vant/lib/index.css'//vantUI 样式

Vue.use(Vant) //把VantUI当做插件，在Vue中使用

就可在其他文件中直接使用，但会增大体积

**按需引入**：在要用的组件下

import { Button } from 'vant'; //组件

import 'vant/lib/button/style'; //样式

export default {

  components: {

Button

//’v-button’:Buton

  }

## vue路由

import Vue from 'vue'

import VueRouter from 'vue-router'

import Home from '../views/Home.vue'

Vue.use(VueRouter)

const routes = [

  {

    path: '/',

    name: 'Home',

    component: Home

  },

  {

    path: '/about',

    name: 'About',

    // route level code-splitting

    // this generates a separate chunk (about.[hash].js) for this route

    // which is lazy-loaded when the route is visited.

    component: () => import(/\* webpackChunkName: "about" \*/ '../views/About.vue')

  },

  {

    path: '/search',

    name: 'Search',

    component: () => import('../views/Search.vue')

  },

  {

    path: '/sight/list',

    name: 'SightList',

    component: () => import('../views/sight/SightList.vue')

  },

  {

    path: '/sight/detail/:id',

    name: 'SightDetail',

    component: () => import('../views/sight/SightDetail.vue')

  },

  {

    path: '/sight/info/:id',

    name: 'SightInfo',

    component: () => import('../views/sight/SightInfo.vue')

  },

  {

    path: '/sight/comment/:id',

    name: 'SightComment',

    component: () => import('../views/sight/SightComment.vue')

  },

  {

    path: '/sight/image/:id',

    name: 'SightImage',

    component: () => import('../views/sight/SightImage.vue')

  },

]

const router = new VueRouter({

  routes

})

export default router

## Banner图



<template>

<!-- 首页轮播图 -->

    <div class="hone-banner-box">

        <van-swipe class="my-swipe" :autoplay="2000" indicator-color="white">

            <van-swipe-item v-for="item in bannerList" :key="item.id">

                <img :src="item.img" alt=""><!-- 要加绑定 -->

<!-- <img src="static\home\banner\banner1.jpg" alt=""> -->

            </van-swipe-item>

        </van-swipe>

    </div>

</template>

<script>

export default {

    data(){

        return{

            bannerList:[]

        }

    },

    methods:{

        getDataList(){

            ajax.get(SystemApis.sliderListUrl).then(res => {

                this.bannerList=res.data.objects

            })

        }

    },

    created(){

        this.getDataList()

    }

}

</script>

<style lang="less">

.home-banner-box{

    img{

        width: 100%;

        height: auto;

    }

}

</style>

## 热门推荐



<template>

<!-- 热门景点 -->

    <div class="home-hot-box">

       <!-- 导航 -->

       <van-cell title="热门推荐" value="全部榜单" is-link title-style="text-align:left" icon="static/home/hot/fire.png"/>

       <!-- //导航 -->

       <!-- 景点列表 -->

       <div class="box-main">

           <a href="#" class="hot-item" v-for="item in dataList" :key="item.id">

               <div class="img">

                   <span></span>

                   <img :src="item.img" alt="">

               </div>

               <h5 class="van-ellipsis">{{item.name}}</h5>

               <div class="line-price">

                   <span class="price">￥{{item.price}}</span>起

               </div>

           </a>

       </div>

       <!-- //景点列表 -->

    </div>

</template>

<script>

export default {

    data(){

        return{

            dataList:[]

        }

},

// methods:{

    //     getDataList(){

    //         ajax.get(SightApis.sightListUrl,{params:{is\_hot:1}}).then(({data})=>{

    //             console.log(data.objects)

    //             this.dataList=data.objects

    //         })

    //     }

    // },

created() {

// this.getDataList()

        this.dataList=[

            {id:1,img:'/static/home/hot/h1.jpg',name:'名称',price:64},

            {id:2,img:'/static/home/hot/h2.jpg',name:'名称名称名称名称名称名',price:64},

            {id:3,img:'/static/home/hot/h3.jpg',name:'名称',price:64},

            {id:4,img:'/static/home/hot/h4.jpg',name:'名称',price:64},

            {id:5,img:'/static/home/hot/h5.jpg',name:'名称',price:64},

        ]

    },

}

</script>

<style lang="less">

.home-hot-box{

    padding: 0 10px;

    .van-cell{

        padding: 10px 0;

    }

    .box-main{

        width: 100%;

        display: flex;

        overflow-x: scroll; //横向滚动条

        .hot-item{

            display: flex;

            flex-direction: column; //flex竖向排列

            margin-right: 10px;

            width: 100px;

            &:nth-child(1) .img span{ //

                background: url(/static/home/hot/top1.png) no-repeat;

                background-size: 100% auto;

            }

            &:nth-child(2) .img span{

                background: url(/static/home/hot/top2.png) no-repeat;

                background-size: 100% auto;

            }

            &:nth-child(3) .img span{

                background: url(/static/home/hot/top3.png) no-repeat;

                background-size: 100% auto;

            }

            .img{

                position: relative;

                span{

                    position:absolute;

                    left: 0;

                    top: 0;

                    display:inline-block;

                    width: 42px;

                    height: 20px;

                    z-index: 10;

                }

                img{

                    width: 100px;

                    height: 100px;

                }

            }

            h5{

                color: #212121;

                font-size: 12px;

            }

            .line-price{

                color: #212121;

                font-size: 12px;

                .price{

                    color: #f50;

                    font-size: 13px;

                }

            }

        }

    }

}

</style>

## 精选景点



<template>

<!-- 精选景点 -->

    <div class="home-fine-box">

        <!-- 导航 -->

        <van-cell title="精选景点" value="更多" is-link title-style="text-align:left" icon="location-o"/>

        <!-- //导航 -->

        <!-- 景点列表 -->

        <div class="box-main">

            //<SightItem v-for="item in dataList" :key="item.id" :item="item"/> 父到子

            <a href="#" class="sight-item" v-for="item in dataList" :key="item.id">

                <img :src="item.img" alt="">

                <div class="right">

                    <h5>{{item.name}}</h5>

                    <van-rate v-model="item.score" allow-half readonly/>

                    <div class="tips">4人点评 | 100%满意</div>

                    <div class="tips light">广东-广州</div>

                    <div class="line-price">￥ {{item.price}} 起</div>

                </div>

            </a>

        </div>

        <!-- //景点列表 -->

    </div>

</template>

<script>

export default {

    data(){

        return{

            dataList:[]

        }

},

// methods:{

    //     getDataList(){

    //         ajax.get(SightApis.sightListUrl,{params:{is\_top:1}}).then(({data})=>{

    //             console.log(data.objects)

    //             this.dataList=data.objects

    //         })

    //     }

    // },

created(){

// this.getDataList()

        this.dataList=[

            {id:1,img:'/static/home/hot/h1.jpg',name:'名称',score:3.5,price:64},

            {id:2,img:'/static/home/hot/h2.jpg',name:'名称',score:3,price:64},

            {id:3,img:'/static/home/hot/h3.jpg',name:'名称',score:1.5,price:64},

            {id:4,img:'/static/home/hot/h4.jpg',name:'名称',score:5,price:64},

            {id:5,img:'/static/home/hot/h5.jpg',name:'名称',score:3.5,price:64},

        ]

    }

}

</script>

<style lang="less">

.home-fine-box{

    padding: 10px;

    .van-cell{

        padding: 10px 0;

    }

    .box-main{

        .sight-item{

            display: flex;

            // border-bottom: 1px solid #999;

            padding-top: 10px;

            img{

                width: 100px;

                height: 100px;

                margin-right: 10px;

            }

            .right{

                text-align: left;

                flex: 1;

                position: relative;

                border-bottom: 1px solid #f6f6f6;

                h5{

                    color: #212121;

                    font-size: 14px;

                    padding: 5px 0;

                }

                .line-price{

                    position:absolute;

                    display:inline-block;

                    right: 10px;

                    top: 10px;

                    color: #f50;

                    font-size: 16px;

                    font-weight: bold;

                }

                .tips{

                    font-size: 12px;

                    color: #666;

                }

                .light{

                    color: #999;

                }

            }

        }

    }

}

</style>

## 页面底部组件

IMG_256

<template>

<!-- 底部导航 -->

    <div>

        <van-tabbar v-model="active">

            <van-tabbar-item name="home" icon="home-o">首页</van-tabbar-item>

            <van-tabbar-item name="search" icon="search">搜索</van-tabbar-item>

            <van-tabbar-item name="mine" icon="user-o">我的</van-tabbar-item>

        </van-tabbar>

    </div>

</template>

<script>

export default {

  data() {

    return {

      active: 'home',

    };

  },

};

</script>

## 搜索页面



IMG_256

<template>

<!-- 搜索 -->

    <div class="page-search">

        <!-- 标题 -->

        <van-nav-bar title="搜索景点"/>

        <!-- //标题 -->

        <!-- 搜索框 -->

        <van-search v-model="sightName" show-action label="景点" placeholder="请输入搜索关键词" @search="onSearch" @clear="onClear">

            <template #action>

                <div @click="onSearch">搜索</div>

            </template>

        </van-search>

        <!-- //搜索框 -->

        <!-- 景点列表 -->

        <div class="sight-list">

            <SightItem v-for="item in dataList" :key="item.id" :item="item"/>

        </div>

        <!-- //景点列表 -->

        <!-- 分页 -->

        <van-pagination v-model="currentPage" :total-items="totalItem" :items-per-page="perPage" :show-page-size="3" @change="getDataList"/>

        <!-- //分页 -->

        <!-- 页脚 -->

        <TripFooter/>

    </div>

</template>

<script>

import SightItem from '@/components/common/ListSight'

import {ajax} from '@/utils/ajax.js'

import {SightApis} from '@/utils/apis'

import TripFooter from '@/components/common/Footer'

export default {

    components:{

        SightItem,

        TripFooter,

    },

    data(){

        return{

            sightName:'',

            dataList:[],

            totalItem:0,//总记录数

            currentPage: 1,//当前页码

            perPage:5,//每页数据大小

        }

    },

    methods:{

        onSearch(){//点击搜索

            if(!this.sightName){

                this.$toast({position: 'top',message:'请输入搜索词'})

            }else{

                //重置数据

                this.currentPage=1

                this.getDataList()

            }

        },

        onClear(){//点击清除

            this.currentPage=1

            this.getDataList()

        },

        getDataList(){

            ajax.get(SightApis.sightListUrl,{params:{

                page:this.currentPage,

                name:this.sightName,

                limit:this.perPage,

                }}).then(({data})=>{

                // console.log(data)

                this.totalItem=data.meta.total\_count//总记录数

                this.dataList=data.objects

            })

        }

    },

    created() {

        this.getDataList()

    },

}

</script>

<style lang="less">

.page-search{

    background-color: #fff;

    margin-bottom: 60px;

    .sight-list{

        padding: 0 10px 0 10px;

    }

}

</style>

## 景点详情



<template>

<!-- 景点详情 -->

    <div class="page-sight-detail">

        <!-- 页面头部 -->

        <van-nav-bar fixed left-text="返回" left-arrow @click-left="goBack"/>

        <!-- //页面头部 -->

        <!-- 大图 -->

        <div class="sight-banner">

            <van-image width="100%" height="100%" :src="sightDetail.img"/>

            <div class="tips">

                <router-link class="pic-sts" :to="{name:'SightImage',params:{id:1}}">

                    <van-icon name="photo-o" />

                    <span>{{sightDetail.image\_count}} 图片</span>

                </router-link>

                <div class="title">{{sightDetail.name}}</div>

            </div>

        </div>

        <!-- //大图 -->

        <!-- 评分，景点介绍 -->

        <div class="sight-info">

            <div class="left" @click="goPage()">

                <div class="info-title">

                    <span>{{sightDetail.score}}分</span>

                    <small>很棒</small>

                </div>

                <div class="info-tips">{{sightDetail.comment\_count}} 条评论</div>

                <van-icon name="arrow" />

            </div>

            <div class="right">

                <div class="info-title">

                    <span>景点介绍</span>

                </div>

                <div class="info-tips">开放时间、贴士</div>

                <van-icon name="arrow" />

            </div>

        </div>

        <!-- //评分，景点介绍 -->

        <!-- 地址 -->

        <van-cell :title="fullArea" icon="location-o" title-style="text-align:left" is-link/>

        <!-- //地址 -->

        <!-- 门票列表 -->

        <div class="sight-ticket">

            <van-cell title="门票" title-style="text-align:left" icon="coupon-o" />

            <div class="ticket-item" v-for="item in TicketList" :key="item.id">

                <div class="left">

                    <div class="title">{{item.name}}</div>

                    <div class="tips">

                        <van-icon name="clock-o" />

                        <span>{{item.desc|desc}}</span>

                        <div class="tags">

                            <van-tag mark type="primary" color="#999">{{item.return\_policy}}</van-tag>

                        </div>

                    </div>

                </div>

                <div class="right">

                    <div class="price">

                        <span>￥</span>

                        <strong>{{item.price}}</strong>

                    </div>

                    <router-link :to="{name:'OrderSubmit',params:{id:item.id}}">

                        <van-button type="warning" size="small">预定</van-button>

                    </router-link>

                </div>

            </div>

        </div>

        <!-- //门票列表 -->

        <!-- 用户评价 -->

        <div class="sight-comment">

            <div class="cmt">

                <van-cell title="热门评论" title-style="text-align:left" icon="comment-o" />

            </div>

            <CommentItem v-for="item in commentList" :key="item.id" :item="item"/>

            <router-link class="link-more" :to="{name:'SightComment',params:{id}}">查看更多 >>></router-link>

        </div>

    </div>

</template>

<script>

import CommentItem from '@/components/sight/CommentItem'

import {ajax} from '@/utils/ajax.js'

import {SightApis} from '@/utils/apis'

export default {

    components:{

        CommentItem

    },

    data(){

        return{

            id:"",

            sightDetail:{},

            TicketList:[],

            commentList:[],

        }

    },

    computed:{

        fullArea(){

            let area=''

            if(this.sightDetail.province){

                area+=this.sightDetail.province

            }

            if(this.sightDetail.city){

                area+=this.sightDetail.city

            }

            if(this.sightDetail.area){

                area+=this.sightDetail.area

            }

            if(this.sightDetail.town){

                area+=this.sightDetail.town

            }

            if(!area){

                area='暂未提供地址信息'

            }

            return area

        }

    },

    methods:{

        goBack(){

            this.$router.go(-1)

        },

        goPage(){

            this.$router.push({name:'SightComment',params:{id:this.id}})

        },

        getSightDetail(){

            const url=SightApis.sightDetailUrl.replace('#{id}',this.id)

            ajax.get(url).then(({data})=>{

                // console.log(data)

                this.sightDetail=data

            })

        },

        getTicketList(){

            const url=SightApis.sightTicketUrl.replace('#{id}',this.id)

            ajax.get(url).then(({data})=>{

                // console.log(data)

                this.TicketList=data.objects

            })

        },

        getCommentList(){

            const url=SightApis.sightCommentUrl.replace('#{id}',this.id)

            ajax.get(url).then(({data})=>{

                // console.log(data)

                this.commentList=data.objects

            })

        },

        loadData(){

            this.id=this.$route.params.id

            this.getSightDetail()

            this.getTicketList()

            this.getCommentList()

        },

    },

    watch:{

        $route(){

            this.loadData()

        }

    },

    created() {

        this.loadData()

    },

}

</script>

<style lang="less">

.page-sight-detail{

    .van-cell{

        padding: 10px;

    }

    .van-nav-bar{

        background-color: transparent;

    }

    .van-hairline--bottom:after{//去除底部边框

        border:0

    }

    .sight-banner{

        position: relative;

        .tips{

            position:absolute;

            left: 10px;

            bottom: 10px;

            font-size: 16px;

            color: #fff;

            .pic-sts{

                    color: #fff;

                    border-radius: 5px;

                    font-size: 14px;

                    background-color: rgba(0, 0, 0, 0.6);

            }

        }

    }

    .sight-info{

        display:flex;

        background-color: #fff;

        border-bottom: 1px solid #f6f6f6;

        &>div{

            flex: 1;

            position: relative;

            text-align: left;

            padding: 5px 10px;

            line-height: 2.0;

        }

        .right{

            border-left: 1px solid #f6f6f6;

        }

        span{

           color: #ff8300;

        }

        .info-tips{

            color: #999;

            font-size: 12px;

        }

        .van-icon{

            position:absolute;

            right: 5px;

            top: 10px;

        }

    }

    .sight-ticket{

        margin-top: 5px;

        background-color: #fff;

        .ticket-item{

            display:flex;

            border-bottom: 1px solid #f6f6f6;

            padding-bottom: 10px;

            .left{

                flex: 1;

                text-align: left;

                padding: 5px 10px;

                .title{

                    padding: 5px 0;

                }

                .tips{

                    font-size: 12px;

                }

            }

            .right{

                width: 100px;

                .price{

                    color: #ff9800;

                    strong{

                        font-size: 20px;

                    }

                }

            }

        }

    }

    .sight-comment{

        margin-top: 5px;

        background-color: #fff;

        .cmt{

            border-bottom: 1px solid #f6f6f6;

        }

    }

    .link-more{

        color: #666;

        padding: 10px;

        display: block;

    }

}

</style>

## 景点评论



过滤器{{item.user.nickname||'匿名用户'|unameFormat}}

将dict类型图片地址提取出

IMG_256

computed:{

        imageUrls(){

            return this.item.images.map(i => i.img)

        }

    },

<template>

<!-- 评论列表 -->

    <div class="comment-item-box">

        <div class="cmt-header">

            <van-rate v-model="value" allow-half readonly/>

            <div class="user">{{item.user.nickname||'匿名用户'|unameFormat}} {{item.created\_at}}</div>

        </div>

        <!-- 评论 -->

        <div class="cmt-content">

            <span>{{item.content}}</span>

        </div>

        <!-- //评论 -->

        <!-- 图片 -->

        <div class="cmt-imgs">

            <van-image width="100" height="100" :src="image.img" @click="show=true" v-for="(image,index) in item.images" :key="index"/>

        </div>

        <van-image-preview v-model="show" :images="imageUrls" @change="onChange" :start-position="index"/>

        <!-- //图片 -->

    </div>

</template>

<script>

export default {

    props:['item'],

    data() {

        return {

            value:this.item.score,

            show: false,

            index: 0,

        }

    },

    computed:{

        imageUrls(){

            return this.item.images.map(i => i.img)

        }

    },

    methods: {

        onChange(index) {

            this.index = index;

        },

    },

}

</script>

<style lang="less">

.comment-item-box{

    padding: 5px 10px;

    border-bottom: 1px solid #f6f6f6;

    .cmt-header{

        display: flex;

        justify-content:space-between;

        .user{

            font-size: 12px;

        }

    }

    .cmt-content{

        color: #616161;

        padding: 5px 0;

        text-align: left;

        font-size: 14px;

        line-height: 2.0;

    }

    .cmt-imgs{

        padding: 5px 0 5px 0;

        text-align: left;

        margin-right: 5px;

    }

}

</style>

## 评论列表(下拉加载)

<template>

<!-- 景点评论 -->

    <div class="page-sight-comment">

        <!-- 页面头部 -->

        <van-nav-bar fixed left-text="返回" title="景点评论" left-arrow @click-left="goBack"/>

        <!-- //页面头部 -->

        <!-- 评论列表 -->

        <van-pull-refresh v-model="refreshing" @refresh="onRefresh">

            <van-list class="sight-comment" v-model="loading" :finished="finished" finished-text="没有更多了" @load="getCommentList" :error.sync="error" error-text="请求失败，点击重新加载">

                <CommentItem v-for="item in commentList" :key="item.id" :item="item"/>

            </van-list>

        </van-pull-refresh>

        <!-- //评论列表 -->

    </div>

</template>

<script>

import CommentItem from '@/components/sight/CommentItem'

import {ajax} from '@/utils/ajax.js'

import {SightApis} from '@/utils/apis'

export default {

    components:{

        CommentItem

    },

    data(){

        return{

            id:"",

            commentList:[],

            currentPage:1,

            loading: false,

            finished: false,

            error: false,

            refreshing: false,

        }

    },

    methods:{

        goBack(){

            this.$router.go(-1)

        },

        getCommentList(){

            const url=SightApis.sightCommentUrl.replace('#{id}',this.id)

            ajax.get(url,{params:{page:this.currentPage}}).then(({data})=>{

                // console.log(data)

                this.commentList=this.commentList.concat(data.objects) //拼接评论数据

                this.loading=false

                this.refreshing=false

                this.currentPage=data.meta.current\_page+1 //分页加1

                if(data.meta.current\_page===data.meta.page\_count){//判断是否最后一页

                    this.finished=true

                }

            }).catch(()=>{//错误处理

                this.loading=false

                this.error = true

                this.refreshing=false

            })

        },

        onRefresh() {//下拉刷新

            this.commentList=[] //清空数据

            this.currentPage=1

            this.finished = false//重置状态

            this.error=false

            this.loading = true

            this.getCommentList()

        },

    },

    created() {

        this.id=this.$route.params.id

        // this.getCommentList()

    },

}

</script>

<style lang="less">

.sight-comment{

    margin-top: 46px;

    background-color: #fff;

}

</style>

## ajax(及错误返回)

外部ajax.js要用vue的组件(router,notify等)，要先 [跳转](#_外部js拿vue实例)

const keys=Object.keys(data.err\_list)

const errObj=data.err\_list[keys[0]][0]

import axios from 'axios'

import qs from 'qs'

export const ajax=axios.create({

    headers:{//自定义请求头

        source:'h5',

        'Content-Type':'application/x-www-form-urlencoded',

    },

    withCredentials:true, // 携带上次cookie

    transformRequest:function(data){  //对请求的参数格式化

        return qs.stringify(data)

    }

    //timeout:超时的毫秒

    //data:post/put/patch请求的数据

    //params:url中的参数

    //responseType:默认为json

})

// 发送请求前做什么

ajax.interceptors.request.use(function (config){

    window.app.$toast.loading({

        message: '加载中...',

        forbidClick: true,

      });

    return config;

},function(error){// 错误处理

    window.app.$toast.clear()

    return Promise.reject(error);

})

// 对响应数据做什么

ajax.interceptors.response.use(function (response){

    window.app.$toast.clear()

    return response;

},function(error){// 错误处理

window.app.$toast.clear()

if(error.response.status === 400){

        const data=error.response.data

        let msg=data.error\_msg ? data.error\_msg : '参数错误'

        if(data.err\_list){

            const keys=Object.keys(data.err\_list)

            const errObj=data.err\_list[keys[0]][0]

            msg=`${errObj.message}`

        }

        window.app.$notify({message:msg})

    }

    if(error.response.status === 401){

        window.app.$notify({ type: 'danger', message: '请登录' })

        window.app.$router.replace({name:'AccountLogin'})

    }

    if(error.response.status === 500){

        window.app.$notify({ type: 'danger', message: '服务器正忙，请稍后重试' })

    }

    return Promise.reject(error);

})

## 用户登录



表单的数据要格式化，django才能拿到 [跳转](#_PS)

<template>

<!-- 登录 -->

    <div class="page-account-login">

        <!-- 导航条 -->

        <van-nav-bar left-arrow left-text="返回" title="用户登录" @click-left="goBack"/>

        <!-- //导航条 -->

        <!-- 表单输入 -->

        <van-form @submit="onSubmit">

            <van-field v-model="username" type="tel" label="手机号" placeholder="手机号" :rules="ruleName" maxlength="11" clearable/>

            <van-field v-model="password" type="password" label="密码" placeholder="密码" :rules="rulepassword" clearable/>

            <div style="margin: 16px;">

                <van-button round block type="info" native-type="submit">提交</van-button>

            </div>

        </van-form>

        <!-- //表单输入 -->

        <!-- 文字提示 -->

        <div class="tips">

            登录表示同意 <router-link to="#">用户使用协议</router-link> 及 <router-link to="#">隐私条款</router-link>

        </div>

        <div class="tips">

            没有账号？ <router-link :to="{name:'AccountRegister'}">点击注册>></router-link>

        </div>

        <!-- //文字提示 -->

        <!-- 版权 -->

        <copyright/>

        <!-- //版权 -->

    </div>

</template>

<script>

import copyright from '@/components/common/Copyright'

import {ajax} from '@/utils/ajax.js'

import {AccountsApis} from '@/utils/apis'

export default {

    components:{

        copyright

    },

    data(){

        return{

            username:'',

            password:'',

            ruleName:[{required: true, message: '请填写手机号'},

            {pattern: /1\d{10}/, message: '手机号不合法'}],

            rulepassword:[{required: true, message: '请填写密码'}],

        }

    },

    methods:{

        goBack(){

            this.$router.go(-1)

        },

        onSubmit(){

            ajax.post(AccountsApis.loginUrl,{

                username:this.username,

                password:this.password

            }).then(({data})=>{

                //存入vuex

                this.$store.commit('updatedUserInfo',data)

                this.$toast('登录成功')

                this.$router.replace({name:'Mine'})

            }).catch(({response:{data}})=>{

                // console.log(data)

                this.$toast(`${data.error\_code},${data.error\_msg}`)

            })

        },

    },

}

</script>

## 用户注册



<template>

<!-- 登录 -->

    <div class="page-account-login">

        <!-- 导航条 -->

        <van-nav-bar left-arrow left-text="返回" title="用户注册" @click-left="goBack"/>

        <!-- //导航条 -->

        <!-- 表单输入 -->

        <van-form @submit="onSubmit">

            <van-field v-model="username" type="tel" label="手机号码" placeholder="手机号码" :rules="rule\_name" maxlength="11" required clearable colon/>

            <van-field v-model="smscode" type="digit" label="验证码" placeholder="短信验证码" :rules="rule\_smscode" maxlength="6" required clearable colon>

                <template #button>

                    <SendSmsCode :phone="username"/>

                </template>

            </van-field>

            <van-field v-model="nickname" label="昵称" placeholder="用户昵称" :rules="rule\_nickName" maxlength="10" required clearable colon/>

            <van-field v-model="password" type="password" label="密码" placeholder="密码" :rules="rule\_password" required clearable colon/>

            <van-field v-model="repassword" type="password" label="确认密码" placeholder="确认密码" :rules="rule\_repassword" required clearable colon/>

            <div style="margin: 16px;">

                <van-button round block type="info" native-type="submit">提交</van-button>

            </div>

        </van-form>

        <!-- //表单输入 -->

        <!-- 文字提示 -->

        <div class="tips">

            注册表示同意 <router-link to="#">用户使用协议</router-link> 及 <router-link to="#">隐私条款</router-link>

        </div>

        <div class="tips">

            已有账号？ <router-link :to="{name:'AccountLogin'}">点击登录>></router-link>

        </div>

        <!-- //文字提示 -->

        <!-- 版权 -->

        <copyright/>

        <!-- //版权 -->

    </div>

</template>

<script>

import copyright from '@/components/common/Copyright'

import SendSmsCode from '@/components/common/SendSmsCode'

import {ajax} from '@/utils/ajax.js'

import {AccountsApis} from '@/utils/apis'

export default {

    components:{

        copyright,

        SendSmsCode

    },

    data(){

        return{

            username:'',

            smscode:'',

            nickname:'',

            password:'',

            repassword:'',

            rule\_name:[{required: true, message: '请填写手机号'},

                    {pattern: /1\d{10}/, message: '手机号不合法'}],

            rule\_smscode:[{required: true, message: '请填写验证码'}],

            rule\_nickName:[{required: true, message: '请填写昵称'}],

            rule\_password:[{required: true, message: '请填写密码'},],

            rule\_repassword:[{required: true, message: '请在次输入密码'},

            {validator:()=>this.password===this.repassword,message: '两次密码不一致'}],

        }

    },

    methods:{

        goBack(){

            this.$router.go(-1)

        },

        onSubmit(){

            // 调用接口

            ajax.post(AccountsApis.registerUrl,{

                username:this.username,

                password:this.password,

                nickname:this.nickname,

                email:this.email,

                sms\_code:this.smscode,

            }).then(({data})=>{

                // console.log(data)

                // 成功返回，用户信息存入vuex

                this.$store.commit('updatedUserInfo',data)

                // 提示用户

                this.$notify({

                    message:'注册成功',

                    type:'success'

                })

                // 跳转

                this.$router.replace({name:'Mine'})

            })

        },

    },

}

</script>

## 发送验证码

IMG_256

@click.prevent 阻止事件冒泡

<template>

<!-- 发送短信验证码 -->

        <van-button size="small" type="primary" @click.prevent="sendSmsCode" :disabled="isSend">{{sendBtnText}}</van-button>

</template>

<script>

import {ajax} from '@/utils/ajax.js'

import {SystemApis} from '@/utils/apis'

export default {

    props:['phone'],

    data() {

        return {

            sendBtnText:'发送验证码',

            counter:5,

            isSend:false,

            smslist:[],

        }

    },

    methods:{

        countDown(){

            let timer=setInterval(()=>{

                this.sendBtnText=`(${this.counter}s)后重新发送`

                this.counter--

                if(this.counter<0){

                    clearInterval(timer)

                    this.counter=5

                    this.sendBtnText='发送验证码'

                    this.isSend=false

                }

            },1000)

        },

        sendSmsCode(){

            //判断是否已输入验证码

            if(!this.phone){

                this.$notify('请输入手机号')

                return false

            }

            //调用接口，发送验证码

            ajax.post(SystemApis.sendSmsCodetUrl,{

                phone\_num:this.phone

            }).then(({data})=>{

                // 提示已发送验证码

                this.$notify({

                    message:`验证码为：${data.sms\_code}，${data.time\_out/60}分钟有效`,

                    duration:1000\*10,

                    type:'success'

                })

                this.smslist=data

            }).catch(({response:{data}})=>{

                //错误提示

                // console.log(data)

                this.$toast(`${data.err\_list.phone\_num[0].message}`) //[可用更好的方法实现](#_ajax(及错误返回))(这是超链)

            })

            //开启倒计时

            this.isSend=true

            this.countDown()

        }

    }

}

</script>

## 个人中心



vuex里保存的数据，刷新之后就会消失，通过再次访问用户详情接口获取，再保存至vuex

访问退出登录接口，要同步清空vuex保存的用户数据

<template>

<!-- 个人中心 -->

    <div class="page-mine">

        <!-- 标题 -->

        <van-nav-bar title="个人中心" fixed />

        <!-- //标题 -->

        <!-- 头像、基本信息 -->

        <div class="user-header">

            <div class="avatar">

                <van-image v-if="user.avatar" round width="80px" height="80px" :src="user.avatar"/>

                <van-image v-else round width="80px" height="80px" src="https://img01.yzcdn.cn/vant/cat.jpeg"/>

            </div>

            <p>欢饮您，{{user.nickname}}</p>

            <router-link class="btn-link" to="#" @click.native="logout">退出登录</router-link>

        </div>

        <!-- //头像、基本信息 -->

        <!-- 菜单列表 -->

        <van-row class="user-links">

            <van-col span="6">

                <router-link to="#">

                    <van-icon name="orders-o" />

                    <span>全部订单</span>

                </router-link>

            </van-col>

            <van-col span="6">

                <router-link to="#">

                    <van-icon name="pending-payment" />

                    <span>待支付</span>

                </router-link>

            </van-col>

            <van-col span="6">

                <router-link to="#">

                    <van-icon name="completed" />

                    <span>已完成</span>

                </router-link>

            </van-col>

            <van-col span="6">

                <router-link to="#">

                    <van-icon name="failure" />

                    <span>已取消</span>

                </router-link>

            </van-col>

        </van-row>

        <!-- //菜单列表 -->

        <!-- 底部导航 -->

        <TripFooter/>

        <!-- //底部导航 -->

    </div>

</template>

<script>

import TripFooter from '@/components/common/Footer'

import {mapState} from 'vuex'

import {ajax} from '@/utils/ajax.js'

import {AccountsApis} from '@/utils/apis'

export default {

    components:{

        TripFooter

    },

    computed:{

        ...mapState(['user','profile'])

    },

    methods:{

        getUserIngo(){

            ajax.get(AccountsApis.UserInfo).then(({data})=>{

                //console.log(data)

                this.$store.commit('updatedUserInfo',data)

            })

        },

        logout(){

            //调用接口

            ajax.get(AccountsApis.logoutUrl).then(()=>{

                //提示用户

                this.$notify({

                    type:'success',

                    message:'欢迎下次再来'

                })

                //删除用户登录信息

                this.$store.commit('deleteUserInfo')

                this.$router.replace({name:'Home'})//跳到首页

            })

        }

    },

    mounted(){

        this.getUserIngo()

    }

}

</script>

<style lang="less">

.page-mine{

    .van-nav-bar{

        background-color: transparent;

    }

    .van-nav-bar\_\_title{

        color: #fff;

    }

    .van-hairline--bottom:after{//去除vue导航栏底部边框

        border: 0;

    }

    .user-header{

        background: url(/static/mine/bg.jpg) no-repeat center;

        background-size: 100% auto;

        color: #fff;

        height: 200px;

        display:flex;

        flex-direction: column;

        justify-content: center;

        .btn-link{

            color: #fff;

        }

    }

    .user-links{

        padding: 15px 0;

        font-size: 12px;

        text-align: center;

        background-color: #fff;

        .van-icon{

            display:block;

            font-size: 24px;

        }

    }

}

</style>

## 填写订单



<template>

<!-- 填写订单 -->

    <div class="page-order-submit">

        <!-- 页面头部 -->

        <van-nav-bar  left-text="返回" title="填写订单" left-arrow @click-left="goBack"/>

        <!-- //页面头部 -->

        <!-- 描述信息 -->

        <div class="order-info">

            <div class="left">

                <h3>{{ticketDetail.name}}</h3>

                <div class="tips">22:00前可定</div>

                <span class="tags">

                    <van-icon name="clock-o" />

                    明日可订

                </span>

            </div>

            <div class="right">

                <div class="text-warning">￥{{price}}/张</div>

                <van-button plain type="primary" size="mini" @click="showPopup=true">预定须知</van-button>

                <van-popup v-model="showPopup" position="bottom" :style="{ height: '80%' }" closeable>

                    <ticketTips :ticketDetail="ticketDetail"/>

                </van-popup>

            </div>

        </div>

        <!-- //描述信息 -->

        <!-- 提交表单 -->

        <van-form class="form-box" validate-first>

           <van-field readonly required :value="play\_date" label="出行日期" placeholder="点击选择出行日期" @click="showCalendar = true" :rules="[{ required: true, message: '请选择出行日期' }]"/>

            <van-calendar v-model="showCalendar" @confirm="onConfirm" />

            <van-field label="购买数量" input-align="right">

                <template #input>

                    <van-stepper disable-input v-model="buy\_count" min="1" max="99"/>

                </template>

            </van-field>

            <van-cell-group>

                <van-field required v-model="to\_user" label="收件人" placeholder="收件人" :rules="[{ required: true, message: '请填写用户名' }]" />

                <van-field required v-model="to\_phone" type="tel" label="手机号" placeholder="手机号" maxlength="11" :rules="[{ required: true, message: '请填写手机号' }]"/>

            </van-cell-group>

            <van-submit-bar :price="price\*100\*buy\_count" button-text="提交订单" @submit="onSubmit" />

        </van-form>

        <!-- //提交表单 -->

    </div>

</template>

<script>

import {ajax} from '@/utils/ajax.js'

import {OrderApis} from '@/utils/apis'

import {mapState} from 'vuex'

export default {

    data(){

        return{

            id:'',

            ticketDetail:[],

            showPopup:false,

            showCalendar:false,

            play\_date:'',

            buy\_count:'',

            to\_user:'',

            to\_phone:'',

            price:1,

        }

    },

    computed:{

        ...mapState({ //获取vuex中保存的用户信息

            phoneNum:state => state.user.username,

            nickname:state => state.user.nickname

        }),

    },

    methods:{

        goBack(){

            this.$router.go(-1)

        },

        onConfirm(date) {

            this.play\_date=`${date.getFullYear()}-${date.getMonth() + 1}-${date.getDate()}`;

            this.showCalendar = false;

        },

        onSubmit(){

            //ajax接口调用

            ajax.post(OrderApis.ticketSubmitUrl,{

                'ticket\_id':this.id,

                'play\_date':this.play\_date,

                'to\_user':this.to\_user,

                'to\_phone':this.to\_phone,

                'buy\_count':this.buy\_count

            }).then(({data})=>{

                // console.log(data)

                //提示用户

                this.$notify({

                    message:'提交成功，请支付！',

                    type:'success'

                })

                //跳转待支付页面

                this.$router.replace({name:'OrderPay',params:{sn:data.sn}})

            })

        },

        getTicketDetail(){

            let url=OrderApis.ticketDetailUrl.replace('#{id}',this.id)

            ajax.get(url).then(({data})=>{

                // console.log(data)

                this.ticketDetail=data

                this.price=data.sell\_price

            })

        }

    },

    created(){

        this.id=this.$route.params.id

        this.getTicketDetail()

    },

    mounted(){

        this.to\_user=this.nickname

        this.to\_phone=this.phoneNum

    },

}

</script>

<style lang="less">

.page-order-submit{

    .order-info{

        display: flex;

        padding: 10px;

        background-color: #fff;

        .left{

            flex: 1;

            text-align: left;

            h3{

                padding: 5px 0;

            }

            .tips{

                padding: 5px 0;

                color: #999;

                font-size: 12px;

            }

            .tags{

                font-size: 12px;

            }

        }

        .right{

            width: 90px;

            text-align: right;

        }

    }

    .form-box{

        .van-cell{

            margin-top: 10px;

        }

    }

}

</style>

## 订单支付





<template>

<!-- 订单支付 -->

    <div class="page-order-pay">

        <!-- 页面头部 -->

        <van-nav-bar  left-text="返回" title="订单支付" left-arrow @click-left="goBack"  right-text="取消订单" @click-right="onCancelOreder"/>

        <!-- //页面头部 -->

        <!-- 订单号 -->

         <van-cell title="订单号" :value="sn" title-style="text-align:left;flex:0.3" />

        <!-- //订单号 -->

        <!-- 描述信息 -->

        <div class="order-info" v-for="item in orderdetail.items" :key="item.pk">

            <div class="left">

                <h3>{{item.flash\_name}}</h3>

                <div class="tags">

                    <van-icon name="clock-o" />{{item.remark}}

                </div>

                <div>短信接收</div>

            </div>

            <div class="right">

                <van-icon name="arrow" />

            </div>

        </div>

        <!-- //描述信息 -->

        <!-- 提交表单 -->

        <van-submit-bar :price="totalPrice" button-text="立即支付" @submit="onSubmit" />

        <!-- //提交表单 -->

    </div>

</template>

<script>

import {ajax} from '@/utils/ajax.js'

import {OrderApis} from '@/utils/apis'

import \* as constants from '@/utils/constants'

export default {

    data(){

        return{

            sn:'',

            orderdetail:{},

            totalPrice:10

        }

    },

    methods:{

        goBack(){

            this.$dialog.confirm({

                title: '温馨提示',

                message: '您还没有支付，确定返回吗？',

            }) .then(() => {

                this.$router.go(-1)

            }) .catch(() => {

                // on cancel

            });

        },

        getOrderDetail(){

            let url=OrderApis.ordetrDetailUrl.replace('#{id}',this.sn)

            ajax.get(url).then(({data})=>{

                //console.log(data)

                this.orderdetail=data

                this.totalPrice=data.buy\_amount\*100

            })

        },

        onCancelOreder(){

            this.$dialog.confirm({

                title: '温馨提示',

                message: '确定取消该订单吗？',

            }) .then(() => {

                this.$router.go(-1)

                // TODO

            }) .catch(() => {

                // on cancel

            });

        },

        onSubmit(){

            this.$dialog.confirm({

                title: '支付确认',

                message: `支付金额￥${this.totalPrice/100}元`,

            }) .then(() => {

                // 调用接口

                let url=OrderApis.ordetrDetailUrl.replace('#{id}',this.sn)

                ajax.post(url).then(()=>{

                    // 提示用户

                    this.$notify({

                        message:'支付成功，跳至订单列表',

                        type:'success'

                    })

                    //跳转

                    this.$router.replace({name:'OrderList',params:{'status':constants.ORDER\_STATUS\_ALL}})

                })

            }) .catch(() => {

                // on cancel

            });

        }

    },

    created(){

        this.sn=this.$route.params.id

        this.getOrderDetail()

    }

}

</script>

<style lang="less">

.page-order-pay{

    .order-info{

        display: flex;

        background-color: #fff;

        .left{

            flex: 1;

            text-align: left;

            padding: 10px;

            h3{

                padding: 5px 0;

            }

        }

        .right{

            width: 80px;

            text-align: right;

        }

    }

}

</style>

## 我的订单列表

循环字典：v-for="(value,key,index) in constants"



<template>

<!-- 订单列表 -->

    <div class="page-order-list">

        <!-- 页面头部 -->

        <van-nav-bar  left-text="返回" title="我的订单" left-arrow @click-left="goBack"/>

        <!-- //页面头部 -->

        <!-- 订单状态 -->

        <van-tabs v-model="status" @click="tabChange">

            <van-tab :title="value" :name="key" v-for="(value,key,index) in constants.ORDER\_STATUS" :key="index" replace/>

        </van-tabs>

        <!-- //订单状态 -->

        <!-- 订单记录 -->

        <div class="order-list">

            <div class="order-item" v-for="item in dataList" :key="item.sn" v-show="item.sn">

                <div class="order-head">

                    <div class="order-num">订单号：{{item.sn}}</div>

                    <div class="order-status text-warning">{{constants.ORDER\_STATUS[item.status]}}</div>

                </div>

                <div class="order-body">

                    <div class="left">

                        <van-image width="100" height="100" :src="item.item\_first.flash\_img" />

                    </div>

                    <div class="right">

                        <div class="title">{{item.item\_first.flash\_name}}</div>

                        <div class="remark">{{item.remark}}</div>

                    </div>

                </div>

                <div class="order-footer">

                    <div>总共{{item.buy\_count}}件商品 合计￥{{item.buy\_amount}}元</div>

                    <van-button round plain type="info" size="small">查看</van-button>

                    <van-button round plain type="warning" size="small" v-if="item.status===constants.ORDER\_STATUS\_PAY" :to="{name:'OrderPay',params:{id:item.sn}}">去支付</van-button>

                    <van-button round plain type="danger" size="small" v-else @click="deleteOrder(item)">删除</van-button>

                </div>

            </div>

        </div>

        <!-- //订单记录 -->

    </div>

</template>

<script>

import \* as constants from '@/utils/constants'

import {ajax} from '@/utils/ajax.js'

import {OrderApis} from '@/utils/apis'

export default {

    data(){

        return{

            constants,

            status:0,

            dataList:[]

        }

    },

    methods:{

        goBack(){

            this.$router.go(-1)

        },

        getDataList(){

            ajax.get(OrderApis.orderListUrl,{

                params:{'status':this.status}

            }).then(({data})=>{

                // console.log(data)

                this.dataList=data.objects

            })

        },

        loadData(){

            this.status=this.$route.params.status

            this.getDataList()

        },

        tabChange(name){

            this.$router.replace({name:'OrderList',params:{status:name}})

        },

        deleteOrder(item){

            // 弹框确认

            this.$dialog.confirm({

                title: '温馨提示',

                message: '删除订单将无法恢复，确认删除订单？',

            }) .then(() => {

                // 调用接口

                let url=OrderApis.ordetrDetailUrl.replace('#{id}',item.sn)

                ajax.delete(url).then(({status})=>{

                    // 提示用户

                    if(status===201){

                        this.$notify({

                            message:'删除成功',

                            type:'success'

                        })

                        item.sn=''

                    }else if(status===200){

                        this.$notify({

                            message:'删除失败',

                            type:'danger'

                        })

                    }

                })

            }) .catch(() => {

                // on cancel

            });

        }

    },

    created(){

        this.loadData()

    },

    watch:{

        $route(){

            this.loadData()

        },

    },

}

</script>

<style lang="less">

.page-order-list{

    .order-list{

        .order-item{

            background-color: #fff;

            padding: 10px;

            border-radius: 20px;

            margin-bottom: 10px;

            .order-head{

                display: flex;

                justify-content: space-between; //两端对齐

                font-size: 12px;

                .order-status{

                    font-size: 14px;

                }

            }

            .order-body{

                display: flex;

                padding: 10px 0;

                .right{

                    flex: 1;

                    text-align: left;

                    padding-left: 10px;

                    .title{

                        font-size: 16px;

                        padding: 5px 0;

                    }

                    .remark{

                        font-size: 12px;

                        color: #999;

                    }

                }

            }

            .order-footer{

                text-align: right;

                font-size: 12px;

                .van-button{

                    margin-left: 10px;

                    margin-top: 5px;

                    width: 54px;

                }

            }

        }

    }

}

</style>

## vuex

import Vue from 'vue'

import Vuex from 'vuex'

Vue.use(Vuex)

export default new Vuex.Store({

  state: {

    user:{},

    profile:{}

  },

  mutations: {

    updatedUserInfo(state,{user,profile}){

      this.state.user={

        ...state.user,

        ...user

      }

      this.state.profile={

        ...state.profile,

        ...profile

      }

    },

    deleteUserInfo(state){

      state.user={}

      state.profile={}

    }

  },

  actions: {

  },

  modules: {

  }

})

## 优化

### 外部js拿vue实例

main.js中把实例导出到window.app中，就可，例子 [加载动画及错误提示](#_加载动画及错误提示)

window.app = new Vue({

  router,

  store,

  render: h => h(App)

}).$mount('#app')

### 加载动画及错误提示

要先使用 [外部js拿vue实例](#_外部js拿vue实例)

ajax.js

import axios from 'axios'

export const ajax=axios.create({

    headers:{//自定义请求头

        source:'h5',

        'Content-Type':'application/x-www-form-urlencoded',

    },

    withCredentials:true// 携带上次cookie

    //timeout:超时的毫秒

    //data:post/put/patch请求的数据

    //params:url中的参数

    //responseType:默认为jso

})

// 发送请求前做什么

ajax.interceptors.request.use(function (config){

    window.app.$toast.loading({ //加载动画

        message: '加载中...',

        forbidClick: true, /是否禁止背景点击

      });

    return config;

},function(error){// 错误处理

    window.app.$toast.clear() //清除加载动画

    return Promise.reject(error);

})

// 对响应数据做什么

ajax.interceptors.response.use(function (response){

    window.app.$toast.clear()

    return response;

},function(error){// 错误处理

    window.app.$toast.clear()

    if(error.response.status === 401){

        window.alert('请先登录');

    }

    if(error.response.status === 500){

        window.app.$notify({ type: 'danger', message: '服务器正忙，请稍后重试' }) //弹出错误提示

    }

    return Promise.reject(error);

})

### 底部加载、下拉刷新

<template>

<!-- 景点评论 -->

    <div class="page-sight-comment">

        <!-- 页面头部 -->

        <van-nav-bar fixed left-text="返回" title="景点评论" left-arrow @click-left="goBack"/>

        <!-- //页面头部 -->

        <!-- 评论列表 -->

        <van-pull-refresh v-model="refreshing" @refresh="onRefresh">

            <van-list class="sight-comment" v-model="loading" :finished="finished" finished-text="没有更多了" @load="getCommentList" :error.sync="error" error-text="请求失败，点击重新加载">

                <CommentItem v-for="item in commentList" :key="item.id" :item="item"/>

            </van-list>

        </van-pull-refresh>

        <!-- //评论列表 -->

    </div>

</template>

<script>

import CommentItem from '@/components/sight/CommentItem'

import {ajax} from '@/utils/ajax.js'

import {SightApis} from '@/utils/apis'

export default {

    components:{

        CommentItem

    },

    data(){

        return{

            id:"",

            commentList:[],

            currentPage:1,

            loading: false,//加载

            finished: false,//刷新

            error: false,//错误

            refreshing: false,//下拉

        }

    },

    methods:{

        goBack(){

            this.$router.go(-1)

        },

        getCommentList(){

            const url=SightApis.sightCommentUrl.replace('#{id}',this.id)

            ajax.get(url,{params:{page:this.currentPage}}).then(({data})=>{

                // console.log(data)

                this.commentList=this.commentList.concat(data.objects) //拼接评论数据

                this.loading=false

                this.refreshing=false

                this.currentPage=data.meta.current\_page+1 //分页加1

                if(data.meta.current\_page===data.meta.page\_count){//判断是否最后一页

                    this.finished=true

                }

            }).catch(()=>{//错误处理

                this.loading=false

                this.error = true

                this.refreshing=false

            })

        },

        onRefresh() {//下拉刷新

            this.commentList=[] //清空数据

            this.currentPage=1

            this.finished = false//重置状态

            this.error=false

            this.loading = true

            this.getCommentList()

        },

    },

    created() {

        this.id=this.$route.params.id

        // this.getCommentList()

    },

}

</script>

<style lang="less">

.sight-comment{

    margin-top: 46px;

    background-color: #fff;

}

</style>

## 过滤器

function unameFormat (v){//格式化字符串

    if(v){

        return v.substr(0,1)+'\*\*'

    }

    return v

}

function desc(v){

    if(!v){

        return '全天可定'

    }

    return v

}

export {

    unameFormat,

    desc

}

注册过滤器

main.js

Object.keys(filters).forEach(k => Vue.filter(k,filters[k]))

# Django

## 错误返回基类

from django.http import JsonResponse  
  
class NotFoundJsonResponse(JsonResponse):  
 # 400 json响应  
 status\_code = 400  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 data = {  
 'error\_code': 404000,  
 'error\_msg': '您访问的内容不存在！'  
 }  
 super().\_\_init\_\_(data, \*args, \*\*kwargs)  
  
class BadRequestJsonResponse(JsonResponse):  
 # 表单请求没有通过，错误显示  
 status\_code = 400  
  
 def \_\_init\_\_(self, err\_list=[], \*args, \*\*kwargs):  
 data = {  
 'error\_code': 400000,  
 'error\_msg': '参数格式不正确！',  
 'err\_list': err\_list  
 }  
 super().\_\_init\_\_(data, \*args, \*\*kwargs)  
  
class MethodNotAllowedJsonResponse(JsonResponse):  
 # 请求方式不被允许  
 status\_code = 405  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 data = {  
 'error\_code': 405000,  
 'error\_msg': '请求方式不被允许！',  
 }  
 super().\_\_init\_\_(data, \*args, \*\*kwargs)  
  
class UnauthorizedJsonResponse(JsonResponse):  
 status\_code = 401  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 data = {  
 'error\_code': 401000,  
 'error\_msg': '请登录！',  
 }  
 super().\_\_init\_\_(data, \*args, \*\*kwargs)  
  
class ServerErrorJsonResponse(JsonResponse):  
 status\_code = 500  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 data = {  
 'error\_code': 500000,  
 'error\_msg': '服务器正忙,请稍后重试！',  
 }  
 super().\_\_init\_\_(data, \*args, \*\*kwargs)

## 序列化基类

class BaseSerializer:  
 def \_\_init\_\_(self, obj):  
 self.obj = obj  
  
 def to\_dict(self):  
 return {}  
  
class MetaSerializer:  
 # 分页元数据  
 def \_\_init\_\_(self, page, page\_count, total\_count):  
 *'''* ***:param*** *page: 当前第几页* ***:param*** *page\_count: 总页数* ***:param*** *total\_count: 总记录数  
 '''* self.page = page  
 self.page\_count = page\_count  
 self.total\_count = total\_count  
  
 def to\_dict(self):  
 return {'total\_count': self.total\_count,  
 'page\_count': self.page\_count,  
 'current\_count': self.page}  
  
class BaseListPageSerializer:  
 # 分页类封装  
 def \_\_init\_\_(self, page\_obj, paginator=None, object\_list=[]):  
 *'''* ***:param*** *page\_obj: 当前页对象* ***:param*** *paginator: 分页器对象* ***:param*** *object\_list: 当前页数据  
 '''* self.page\_obj = page\_obj  
 self.paginator = paginator if paginator else page\_obj.paginator  
 self.object\_list = object\_list if object\_list else page\_obj.object\_list  
  
 def get\_obj(self, obj):  
 # 对象内容，子类重写  
 return {}  
  
 def to\_dict(self):  
 page = self.page\_obj.number  
 page\_count = self.page\_obj.paginator.num\_pages  
 total\_count = self.page\_obj.paginator.count  
 meta = MetaSerializer(page,page\_count,total\_count).to\_dict()  
  
 objects = []  
 for obj in self.object\_list:  
 objects.append(self.get\_obj(obj))  
 return {  
 'meta': meta,  
 'objects': objects  
 }

## 搜索页面

class SightListView(ListView):  
 paginate\_by = 5 # 每页5条数据  
  
 def get\_queryset(self): # 重写查询方法  
 query = Q(is\_valid=True)  
 is\_hot = self.request.GET.get('is\_hot', None)  
 is\_top = self.request.GET.get('is\_top', None)  
 name = self.request.GET.get('name', None)  
 if is\_hot:  
 query = query & Q(is\_hot=True)  
 if is\_top:  
 query = query & Q(is\_top=True)  
 if name:  
 query = query & Q(name\_\_icontains=name)  
 queryset = Sight.objects.filter(query)  
 return queryset  
  
 def get\_paginate\_by(self, queryset):  
 page\_size = self.request.GET.get('limit', self.paginate\_by)  
 return page\_size  
  
 # 因返回不是JS数据，重写返回方法  
 def render\_to\_response(self, context, \*\*response\_kwargs):  
 page\_obj = context['page\_obj'] # 获取分页数据  
 if page\_obj is not None:  
 data = serializers.SightListSerializer(page\_obj).to\_dict()  
 return http.JsonResponse(data)  
 return NotFoundJsonResponse()

## 用户登录/登出/注册(auto+cbv)

import json  
from django import http  
from django.contrib.auth import logout  
from django.shortcuts import render  
from django.views import View  
from django.views.generic import FormView  
from accounts.forms import LoginForm, RegisterForm  
from accounts import serializers  
from utils.response import \*  
  
class user\_api\_login(View):  
 http\_method\_names = ['post']  
  
 def post(self, request):  
 form = LoginForm(request.POST)  
 # 如果通过验证，执行登录  
 if form.is\_valid():  
 user = form.do\_logon(request)  
 # 返回用户信息  
 profile = user.profile # 外键反向引用  
 data = {  
 'user': serializers.UserSerializer(user).to\_dict(),  
 'profile': serializers.UserProfileSerializer(profile).to\_dict()  
 }  
 return http.JsonResponse(data)  
 else:  
 # 没通过，返回表单错误信息  
 err = json.loads(form.errors.as\_json())  
 return BadRequestJsonResponse(err)  
  
 def http\_method\_not\_allowed(self, request, \*args, \*\*kwargs):  
 return MethodNotAllowedJsonResponse()  
  
class user\_api\_logout(View):  
 http\_method\_names = ['get']  
  
 def get(self, request):  
 logout(request)  
 return http.HttpResponse(status=201)  
  
 def http\_method\_not\_allowed(self, request, \*args, \*\*kwargs):  
 return MethodNotAllowedJsonResponse()  
  
class user\_api\_info(View):  
 http\_method\_names = ['get']  
  
 def get(self, request):  
 user = request.user  
 if user.is\_authenticated:  
 profile = user.profile # 外键反向引用  
 data = {  
 'user': serializers.UserSerializer(user).to\_dict(),  
 'profile': serializers.UserProfileSerializer(profile).to\_dict()  
 }  
 return http.JsonResponse(data)  
 return UnauthorizedJsonResponse()  
  
 def http\_method\_not\_allowed(self, request, \*args, \*\*kwargs):  
 return MethodNotAllowedJsonResponse()  
  
class user\_api\_register(FormView):  
 http\_method\_names = ['post']  
 form\_class = RegisterForm  
  
 def form\_valid(self, form): # 通过的  
 data = form.do\_register(request=self.request)  
 if data is not None:  
 user, profile = data  
 data = {  
 'user': serializers.UserSerializer(user).to\_dict(),  
 'profile': serializers.UserProfileSerializer(profile).to\_dict()  
 }  
 return http.JsonResponse(data, status=201)  
 return ServerErrorJsonResponse()  
  
 def form\_invalid(self, form): # 没通过的  
 err\_list = json.loads(form.errors.as\_json())  
 return BadRequestJsonResponse(err\_list)

## 用户登录相关form表单

import re  
from django import forms  
from django.contrib.auth import authenticate, login  
from django.db import transaction  
from django.utils.timezone import now  
from django.core.cache import cache  
from accounts.models import User, Profile  
  
class LoginForm(forms.Form):  
 username = forms.CharField(label='用户名', max\_length=11, min\_length=11)  
 password = forms.CharField(label='密码', widget=forms.PasswordInput, min\_length=2, max\_length=16)  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs):  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
 self.user = ''  
  
 def clean\_username(self):  
 username = self.cleaned\_data.get('username', None)  
 if not re.search(r'^1[0**-**9]{10}$', username):  
 raise forms.ValidationError('手机号不合法')  
 return username  
  
 def clean(self):  
 data = super().clean()  
 if self.errors:  
 return  
 username = data.get('username', None)  
 password = data.get('password', None)  
 user = authenticate(username=username, password=password) #验证用户名和密码是否匹配  
 if not user:  
 raise forms.ValidationError('用户名或密码不正确')  
 elif user.is\_active:  
 raise forms.ValidationError('用户被禁用')  
 self.user = user  
 return data  
  
 def do\_logon(self, request):  
 user = self.user  
 login(request, user)  
 user.last\_login = now()  
 user.save()  
 user.add\_login\_record(user, request) #添加用户登录历史记录  
 return user  
  
class RegisterForm(forms.Form):  
 username = forms.CharField(label='用户名', max\_length=11, min\_length=11)  
 password = forms.CharField(label='密码', widget=forms.PasswordInput, min\_length=2, max\_length=16)  
 nickname = forms.CharField(label='昵称', max\_length=10, min\_length=1)  
 email = forms.EmailField(label='邮箱', widget=forms.EmailInput, required=True)  
 sms\_code = forms.IntegerField(label='验证码', min\_value=4, max\_value=4)  
  
 def clean\_username(self):  
 phone\_num = self.changed\_data.get('username', None)  
 if not re.search(r'^1[0**-**9]{10}$', phone\_num):  
 raise forms.ValidationError('手机号不合法')  
 if User.objects.filter(username=phone\_num).exists():  
 raise forms.ValidationError('手机号已存在')  
 return phone\_num  
  
 def clean(self):  
 data = super().clean()  
 if self.errors:  
 return  
 username = data.get('username', None)  
 sms\_code = data.get('sms\_code', None)  
 code = cache.get(f'sms\_code\_{username}')  
 if str(code) != sms\_code or sms\_code is None:  
 raise forms.ValidationError('验证码不正确')  
 return data  
  
 @transaction.atomic # 事务控制  
 def do\_register(self, request):  
 data = self.cleaned\_data  
 try:  
 user = User.objects.create\_user(username=data['username'], password=data['password'],  
 nickname=data['nickname'], last\_login=now())  
 profile = Profile.objects.create(user=user, username=user.username,  
 version=request.headers.get('version', None),  
 source=request.headers.get('source', None))  
 login(request, user)  
 user.add\_login\_record(user, request) #添加用户登录历史记录  
 return user, profile  
 except Exception as e:  
 print(e)  
 return None

## 验证码

view

class send\_sms(FormView):  
 http\_method\_names = ['post']  
 form\_class = SendSmsCodeForm  
  
 def form\_valid(self, form): # 通过的  
 data = form.send\_sms\_code()  
 if data is not None:  
 return http.JsonResponse(data, status=201)  
 return ServerErrorJsonResponse()  
  
 def form\_invalid(self, form): # 没通过的  
 err\_list = json.loads(form.errors.as\_json())  
 return BadRequestJsonResponse(err\_list)

form

import random  
import re  
from django import forms  
from django.core.cache import cache  
  
class SendSmsCodeForm(forms.Form):  
 phone\_num = forms.CharField(label='手机号', max\_length=11, min\_length=11)  
  
 def clean\_phone\_num(self):  
 phone\_num = self.cleaned\_data.get('phone\_num', None)  
 if not re.search(r'^1[0**-**9]{10}$', phone\_num):  
 raise forms.ValidationError('手机号不合法')  
 return phone\_num  
  
 def send\_sms\_code(self):  
 # 生成验证码  
 try:  
 phone\_num = self.cleaned\_data.get('phone\_num', None)  
 key = f'sms\_code\_{phone\_num}' # radis的key  
 sms\_code = random.randint(1000, 9999)  
 time\_out = 60 \* 5  
 # *TODO 调用发送短线接口* cache.set(key, sms\_code, timeout=time\_out) #存入redis  
 return { #返回发送成功  
 'phone\_num': phone\_num,  
 'sms\_code': sms\_code,  
 'time\_out': time\_out  
 }  
 except Exception as e:  
 print(e)  
 return None

# Django返回全路劲

setting.py -> MEDIA\_URL = 'http://localhost:8080/'

# Django通用视图

## ListView

教程：https://www.kancloud.cn/cyyspring/django/678881

class SightListView(ListView):  
 paginate\_by = 5 # 每页5条数据  
  
 def get\_queryset(self): # 重写查询方法  
 query = Q(is\_valid=True)  
 is\_hot = self.request.GET.get('is\_hot', None)  
 is\_top = self.request.GET.get('is\_top', None)  
 if is\_hot:  
 query = query & Q(is\_hot=True)  
 if is\_top:  
 query = query & Q(is\_top=True)  
 queryset = Sight.objects.filter(query)  
 return queryset  
  
 def get\_paginate\_by(self, queryset):  
 # 重写每页条数方法，接收前端数据，动态设置每页条数  
 return self.request.GET.get('limit', None) or self.paginate\_by  
  
 # 因返回不是JS数据，重写返回方法  
 def render\_to\_response(self, context, \*\*response\_kwargs):  
 page\_obj = context['page\_obj'] # 获取分页数据  
 data = {  
 'meta': {  
 'total\_count': page\_obj.paginator.count,  
 'page\_count': page\_obj.paginator.num\_pages,  
 'current\_count': page\_obj.number  
 },  
 'objects': []  
 }

# object\_list里的数据不能直接JS返回,要处理过  
 for item in page\_obj.object\_list:  
 data['objects'].append({  
 'id': item.id,  
 'name': item.name,  
 'min\_price': item.min\_price,  
 'main\_img': item.main\_img.url,  
 'score': item.score,  
 'province': item.province,  
 'city': item.city,  
 'comment\_count': 1  
 })  
 return http.JsonResponse(data)

## DetailView

教程：<https://www.kancloud.cn/cyyspring/django/678882>

slug\_field：要查询的参数名

slug\_url\_kwarg：url的参数名

class sight\_sight\_info(DetailView):  
 # 景点介绍  
 slug\_field = 'sight\_\_id'  
  
 def get\_queryset(self):  
 return Info.objects.all()  
  
 def render\_to\_response(self, context, \*\*response\_kwargs):  
 page\_obj = context['object']  
 if page\_obj is None:  
 return NotFoundJsonResponse()  
 data = serializers.SightInfoSerializer(page\_obj).to\_dict()  
 return JsonResponse(data)

## FormView

教程：[https://www.kancloud.cn/cyyspring/django/678883](https://www.kancloud.cn/cyyspring/django/678882)

class TicketOrderSubmitView(FormView):  
 form\_class = SubmitTicketOrderForm  
 http\_method\_names = ['post']  
  
 def form\_valid(self, form): #通过的  
 obj = form.save(user=self.request.user)  
 return http.JsonResponse({  
 'sn': obj.sn  
 }, status=201)  
  
 def form\_invalid(self, form): #没通过  
 err = json.loads(form.errors.as\_json())  
 return BadRequestJsonResponse(err)

## BaseDetailView

class OrderDetailView(BaseDetailView):  
 slug\_field = 'sn'  
 slug\_url\_kwarg = 'sn'  
  
 def get\_queryset(self): # 获取用户的订单列表  
 user = self.request.user  
 return Order.objects.filter(user=user, is\_valid=True)  
  
 def get(self, request, \*args, \*\*kwargs):  
 order\_obj = self.get\_object() # get\_object会从get\_queryset中找到对应的sn返回  
 data = serializers.OredeDetailSerializer(order\_obj).to\_dict()  
 return http.JsonResponse(data)

# Django数据库相关

获取地址参数：

path('ticket/list/<int:pk>'),

sight\_id = self.kwargs.get('pk', None)

## 格式化日期：

数据库日期字段格式化

'created\_at': obj.created\_at.strftime('%Y-%m-%d')

# Django-auth模块

安装：(pycharm，基本已做好了)

1.settings.py -> INSTALLED\_APPS，添加：

'django.contrib.auth',  
'django.contrib.contenttypes',

2.settings.py -> MIDDLEWARE，添加：

'django.contrib.sessions.middleware.SessionMiddleware',#SecurityMiddleware下面  
'django.contrib.auth.middleware.AuthenticationMiddleware',<- #CsrfViewMiddleware'上面

3.python manage.py makemigrations python manage.py migrate 同步模型到数据库

## ORM模型配置

1. 新建admin.py

继承admin.ModelAdmin 如果是系统用户表，继承UserAdmin

@admin.register(Profile) #装饰器注册模型类  
class ProfileAdmin(admin.ModelAdmin):  
 list\_display = ['username','sex'] #列表中显示字段  
 list\_per\_page = 15 #每页条数  
 search\_fields = ['username','sex'] #可搜索字段  
 list\_filter = ['username','sex'] #过滤器

fields/exclude 需要/不需要 的字段

form 自定义表单

ordering 排序

2.setting.py - > INSTALLED\_APPS 添加app(一般可忽略),就可

from django.contrib import admin  
from django.contrib.auth.admin import UserAdmin  
  
from accounts.forms import ProfileEditForm  
from accounts.models import Profile, User  
  
@admin.register(User)  
class MyUserAdmin(UserAdmin):  
 list\_display = ('username', 'nickname', 'is\_active', 'is\_staff', 'date\_joined') # 显示字段  
 search\_fields = ('username', 'nickname') # 搜索字段  
 # add\_fieldsets = ( # 新增的表单  
 # (None, {  
 # 'classes': ('wide',),  
 # 'fields': ('username', 'password1', 'password2', 'nickname'),  
 # }),  
 # )  
 add\_fieldsets = UserAdmin.add\_fieldsets + ( # 新增的表单  
 (None, {'fields': ('nickname',)}),  
 )  
 # fieldsets = ( # 修改的表单  
 # (None, {'fields': ('username', 'password')}),  
 # (\_('Personal info'), {'fields': ('first\_name', 'last\_name', 'email')}),  
 # (\_('Permissions'), {  
 # 'fields': ('is\_active', 'is\_staff', 'is\_superuser', 'groups', 'user\_permissions'),  
 # }),  
 # (\_('Important dates'), {'fields': ('last\_login', 'date\_joined')}),  
 # )  
 fieldsets = UserAdmin.fieldsets + ( # 修改的表单  
 (None, {'fields': ('nickname', 'avatar')}),  
 )  
  
 actions = ('disable\_user', 'enable\_user')  
  
 def disable\_user(self, request, queryset):  
 queryset.update(is\_active=False)  
 disable\_user.short\_description = '批量禁用用户' # 显示中文名  
  
 def enable\_user(self, request, queryset):  
 queryset.update(is\_active=True)  
 enable\_user.short\_description = '批量启用用户' # 显示中文名

@admin.register(Profile)  
class profileAdmin(admin.ModelAdmin):  
 list\_display = ('format\_username', 'sex', 'age', 'created\_at') # 显示字段  
 list\_per\_page = 15 # 分页大小，默认100  
 list\_select\_related = ('user',) # 外键关联一并查出  
 list\_filter = ('sex',) # 快捷搜索  
 search\_fields = ('username', 'user\_\_nickname') # 关联搜索 双下划线跨表关联搜索  
 exclude = ('user',) # 排除的字段  
 form = ProfileEditForm # 表单，可自定义表单验证  
  
 def format\_username(self, obj): # 格式化字段内容  
 return obj.username[:3] + '\*\*\*' + obj.username[-4:]  
  
 format\_username.short\_description = '用户名' # 修改显示的字段名字

## 用户模型

注册用户（User）  游客(AnonymousUser )

常用属性

username 用户名 password 登录密码 email 电子邮箱

is\_staff 是否为内部员工 is\_active 是否为激活用户 is\_superuser 是否为超级管理员

last\_login 最后登录的时间 date\_joined 注册时间

is\_authenticated 是否已登录的只读属性 is\_anonymous 登录是否已失效的只读属性

groups 用户组多对多关系 user\_permissions 用户权限多对多关系

## 用户管理

创建超级管理员 >>> python manage.py createsuperuser

创建普通用户 >>> from django.contrib.auth.models import User

User.objects.create\_user('zhanzhan','123')

## 替换现有用户模型

1. 配置用户模型settings.py 添加：AUTH\_USER\_MODEL = 'accounts.User'

2.继承AbstractUser抽象模型

3.添加字段，同步数据库

## 密码管理

新增、设置、修改用户密码

>>> from django.contrib.auth.models import User

>>> [User.objects.create\_user('bb','bb@qq.com','123')](mailto:User.objects.create_user('bb','bb@qq.com','123')) #新增

>>> u = User.objects.get(username='john')

>>> u.**set\_password**('new password') #设置

>>> u.save()

检查用户密码是否正确

>>> u = User.objects.get(username='john')

>>> u.**check\_password**(‘my password')

## 验证

验证用户名和密码是否匹配 user = authenticate(username, password)

登录：login(request,user,backend=None)

在视图中获取当前用户 request.user

需要登录才可访问 @login\_required装饰函数def my\_view(request)

setting.py -> LOGIN\_URL ='/accounts/user/login/'

判断用户是否具备某权限 request.user.has\_perm('foo.add\_bar')

## 扩展用户模型

方式1：使用OneToOneField对用户进行扩展

方式2(推荐)：替换现有的用户模型

1.settings.py 添加：AUTH\_USER\_MODEL = 'app名.模型名'

2.继承AbstractUser抽象模型

from django.contrib.auth.models import AbstractUser

3.添加字段，同步模型到数据库

## 优化配置

### 本地化

**中文**：

1. setting.py -> LANGUAGE\_CODE = 'zh-hans'

时区：setting.py -> TIME\_ZONE = 'Asia/Shanghai'

2.apps.py -> verbose\_name = '用户账户'

class AccountsConfig(AppConfig):  
 name = 'accounts'  
 verbose\_name = '用户账户'

3.模型 -> verbose\_name = '用户表'

class User(models.Model):  
 nickname = models.CharField(verbose\_name='密码', max\_length=32)  
 class Meta:  
 db\_table = 'account\_user'  
 verbose\_name = '用户表'  
 verbose\_name\_plural = verbose\_name

### 查询速度优化

减少每页大小：list\_per\_page默认100

减少查询次数：list\_select\_related 把外键关联一并查出

@admin.register(Profile)  
class ProfileAdmin(admin.ModelAdmin):  
 list\_display = ['username','sex']  
 list\_per\_page = 15

list\_select\_related = ['user']

### 格式化字段内容

直接写个函数后，添加到list\_display就可

@admin.register(Profile)  
class ProfileAdmin(admin.ModelAdmin):  
 list\_display = ['username', 'format\_username', 'sex']  
  
 def format\_username(self, obj): #obj:Profile  
 return obj.username[:3] + '\*\*\*' + obj.username[-4:]  
 format\_username.short\_description = '用户名' #修改表头名字



### 快捷搜索

list\_filter

@admin.register(Profile)  
class ProfileAdmin(admin.ModelAdmin):  
 list\_display = ['username', 'sex']  
 list\_filter = ['sex']



### 关联搜索

search\_fields

跨表关联搜索：user\_\_nickname 外键字段双下划线加字段名

user = models.OneToOneField(to='User', related\_name='profile')

@admin.register(Profile)  
class ProfileAdmin(admin.ModelAdmin):  
 list\_display = ['username', 'format\_username', 'sex']  
 search\_fields = ['username', 'sex', 'user\_\_nickname']

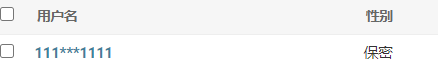
IMG_256

## 自定义模型配置

### 定制列表显示字段

list\_display

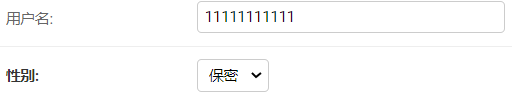
@admin.register(Profile)  
class ProfileAdmin(admin.ModelAdmin):  
 list\_display = ['format\_username', 'sex']



### 限制可编辑字段

fields/exclude 需要/不需要 的字段

@admin.register(Profile)  
class ProfileAdmin(admin.ModelAdmin):  
 list\_display = ['format\_username', 'sex']  
 fields = ['username','sex']



或 模型类字段添加editable=True

username = models.CharField('用户名', max\_length=64, editable=True)

### 自定义表单验证

编辑form，添加验证，就可

form = ProfildEditForm，新建form验证

@admin.register(Profile)  
class ProfileAdmin(admin.ModelAdmin):  
 fields = ['username', 'sex','age']  
 form = ProfildEditForm

form.py

class ProfildEditForm(forms.ModelForm):  
 class Meta:  
 model = Profile  
 fields = ('username', 'sex', 'age')  
 def clean\_age(self):  
 age = self.cleaned\_data.get('age', None)  
 if 0 <= int(age) <= 120:  
 return age  
 return forms.ValidationError('年龄不在0~120之间')

### 重写保存方法，执行其他业务

在form中重写save方法，commit改false

form.py

def save(self, commit=False):  
 obj = super().save(commit)  
 if not obj.source:  
 obj.source = 'web'  
 obj.save()  
 return obj

### 增加批量操作

定义方法xxx(self,request,queryset),后添加到actions

@admin.register(User)  
class MyUserAdmin(UserAdmin):  
 actions=('disable\_user',)

def disable\_user(self,request,queryset):  
 queryset.update(is\_active=False)

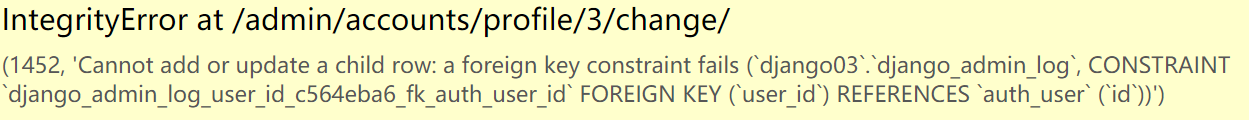
disable\_user.short\_description = '名称' #显示中文名

写成公共模块

from django.contrib import messages  
  
def set\_invalid(modeladmin, request, queryset):  
 queryset.update(is\_valid=False)  
 messages.success(request, '禁用操作成功')  
  
set\_invalid.short\_description = '批量禁用操作' # 显示中文名  
  
def set\_valid(modeladmin, request, queryset):  
 queryset.update(is\_valid=True)  
 messages.success(request, '启用操作成功')  
  
set\_valid.short\_description = '批量启用操作' # 显示中文名

# 错误

新增用户信息出错https://blog.csdn.net/jabony/article/details/78237875



# Redis数据库

windows版：<https://github.com/tporadowski/redis/releases>

启动：redis-server windows-service.conf

连接redis：redis-cli

设置值：set key value  获取值：get key

设多个值：mset k v k v  获取多个值 :mget k k k

append -- 添加字符串  del -- 删除  incr/decr -- 增加/减少 1

## django+redis

1安装：pip install django-redis

2配置settings.py，最后面添加：

CACHES = {  
 "default": {  
 "BACKEND": "django\_redis.cache.RedisCache",  
 "LOCATION": "redis://127.0.0.1:6379/1",  
 "OPTIONS": {  
 "CLIENT\_CLASS": "django\_redis.client.DefaultClient",  
 }  
 }  
}

3存取

from django.core.cache import cache

# 存储 cache.set(key, value, timeout=None) # 查询 cache.get(key)

## 案例

后台缓存，保存时，将首页的精选景点的第一页存到redis

admin.py

def cache\_top\_sight(self):  
 # 获取数据集  
 queryset = Sight.objects.filter(is\_valid=True, is\_top=True)  
 try:  
 # 对数据集进行分页处理  
 p = Paginator(queryset, 20)  
 # 获取第1页数据  
 page\_obj = p.page(1)  
 # 序列化数据  
 data = serializers.SightListSerializer(page\_obj).to\_dict()  
 # 把数据转成字符串，存入redis  
 cache.set(INDEX\_SIGHT\_TOP\_KEY, json.dumps(data), INDEX\_SIGHT\_TIMEOUT)  
 except EmptyPage as e:  
 print('暂无数据', e)  
 except Exception as e:  
 print('缓存失败', e)  
  
def save\_form(self, request, form, change):#重写父类保存方法  
 obj = super().save\_form(request, form, change) # 调用父类获取实例  
 # 将景点缓存到redis  
 self.cache\_top\_sight()  
 return obj

# 富文本拓展

安装：pip install django-ckeditor

配置：

1. setting.py -> INSTALLED\_APPS添加'ckeditor'
2. setting.py添加STATIC\_ROOT = os.path.join(BASE\_DIR, 'static')(先创建'static'文件夹)
3. 执行：python manage.py collectstatic

文件上传配合使用时需要配置：

1. setting.py -> INSTALLED\_APPS添加'ckeditor\_uploader'
2. setting.py添加CKEDITOR\_UPLOAD\_PATH = 'uploads'
3. setting.py添加MEDIA\_ROOT=os.path.join(BASE\_DIR,'medias')(先创建'medias'文件夹)
4. 根目录urls.py添加path('ckeditor/', include('ckeditor\_uploader.urls'))

使用：

将模型中的models.TextField替换成RichTextField，就可

content = RichTextField('详细',null=True,blank=True)

# echarts统计报表

<http://echarts.apache.org/zh/index.html>

下载后，将echarts.min.js进static中就可参考文档使用

如果没有显示，在setting.py添加STATICFILES\_DIRS = [BASE\_DIR / 'static']，不能和STATIC\_ROOT同时使用

{% load static %}  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <script src="/static/echarts.min.js"></script>  
</head>  
<body>  
 <!-- 为 ECharts 准备一个具备大小（宽高）的 DOM -->  
 <div id="main" style="width: 600px;height:400px;"></div>  
</body>  
</html>  
  
<script type="text/javascript">  
 // 基于准备好的dom，初始化echarts实例  
 var ***myChart*** = echarts.init(***document***.getElementById('main'));  
  
 // 指定图表的配置项和数据  
 var ***option*** = {  
 title: {  
 text: 'ECharts 入门示例'  
 },  
 tooltip: {},  
 legend: {  
 data:['销量']  
 },  
 xAxis: {  
 data: ["衬衫","羊毛衫","雪纺衫","裤子","高跟鞋","袜子"]  
 },  
 yAxis: {},  
 series: [{  
 name: '销量',  
 type: 'bar',  
 data: [5, 20, 36, 10, 10, 20]  
 }]  
 };  
  
 // 使用刚指定的配置项和数据显示图表。  
 myChart.setOption(option);  
</script>

# Vuex

专为 Vue.js 应用程序开发的状态管理模式，全局共享数据

Vue CLI脚手架已集成

1.安装：cnpm install vuex -S

2.配置

新建store文件夹 -> index.js

import Vue from 'vue'

import Vuex from 'vuex'

Vue.use(Vuex)

export default new Vuex.Store({

  state: {//全局唯一数据源

  },

  mutations: {//改变state数据

  },

  actions: {//异步

  },

  modules: {//模块

  }

})

main.js注册

import router from './router'

window.app=new Vue({

  ...

**store**,

  ...}).$mount('#app')

查询state中的数据:

computed: {

user () {

return this.$store.state.user

}

}

## mutation

更改store中状态的唯一方法

提交:

vue中的methods写函数通过this.$store.commit提交

    methods:{

       submit(){

            this.$store.commit('updUsername',{

                uname:this.uname

            })

        },}

Vuex.Store中mutations写接收方法，改内容

  state: {

    username:'aa',

  },

  mutations: {

    updUsername (state,payload) {

      this.state.username=payload.uname

    }

  },

## action

同mutation，可异步

提交:

vue中的methods写函数通过this.$store.dispatch提交

    methods:{

        submit(){

            this.$store.**dispatch**('increment2',{

                uname:this.uname

            })

        },

}

vue中传到actions再调用mutations中方法作修改，actions可执行异步

  state: {

    username:'aa',

  },

  mutations: {

    increment (state,payload) {

      this.state.username=payload.uname

    }

  },

  actions: {

    increment2 (context,value) {

      context.commit('increment',value.uname)

    }

  },

异步例子:

  actions: {

    increment2 (context,value) {

      setTimeout(()=>{//设置5秒延迟修改

        context.commit('increment',value.uname)

      },5000)

    }

  },

## 辅助函数

mapState，mapMutations，mapActions

**mapState**:

快捷访问state

先import {mapState} from 'vuex'

对象方式，使用{{ state }}

    computed:mapState({

        username:state => state.user.username,

        nickname:state => state.profile.nickname,

    }),

数组方式，使用{{ user.username }}

computed:mapState(['user','profile']),

结构赋值(computed中已有内容，用...三点结构赋值)

computed:{

xxx

        ...mapState(['user','profile']),

    },

**mapMutations**：

快捷访问mutations

先import {mapMutations} from 'vuex'

methods:{

        ...mapMutations(['updname']), #methods中结构赋值,下面就可直接使用

        submit(){

            this.updname({uname:this.uname})

        },

}

改方法名，把[]换成{}就可

        ...mapMutations({

            updata:'updname'

        }),

**mapActions**：

快捷访问actions

同上mapMutations

先import -> methods中结构赋值 -> 直接使用(把mapMutations改mapActions就可)

# 部署

https://docs.djangoproject.com/zh-hans/2.2/howto/deployment/

Django常见部署方式

nginx + uwsgi

nginx + gunicorn

docker

## 切换下载源

1.备份原文件：

mv /etc/yum.repos.d/CentOS-Base.repo /etc/yum.repos.d/CentOS-Base.repo.backup

1. 下载配置文件(centos 8)并放入/etc/yum.repos.d/，更名为CentOS-Base.repo

wget -O /etc/yum.repos.d/CentOS-Base.repo http://mirrors.aliyun.com/repo/Centos-8.repo

1. 生成缓存运行：yum makecache
2. 查看yum repolist

## 防火墙

1，设置防火墙

开启3306(Mysql远程访问)：[root@cpyy ~]# firewall-cmd --permanent --add-port=3306/tcp

重启防火墙：[root@cpyy ~]# service firewalld restart

开启：service firewalld start

重启：service firewalld restart

关闭：service firewalld stop

查询端口是否开放：firewall-cmd --query-port=3306/tcp

开放端口：firewall-cmd --permanent --add-port=3306/tcp

移除端口：firewall-cmd --permanent --remove-port=3306/tcp

重启防火墙(修改配置后)：firewall-cmd --reload

## 实现步骤

一，准备操作系统，安装必要软件

[Web服务器Nginx](#_nginx)

[关系型数据库MySQL](#_关系型数据库MySQL)

[非关系型数据库Redis](#_非关系型数据库Redis)

[Python3](#_Python3)

二，配置Nginx

三，[部署Vue.js项目](#_部署Vue.js项目)

四，[部署Django项目](#_部署Django项目)

五，[配置慕旅游网(项目)](#_配置慕旅游网(项目))

## nginx

nginx官网：<http://nginx.org/en/download.html>

nginx教程：<http://nginx.org/en/linux_packages.html#RHEL-CentOS>

1. 安装前置软件：sudo yum install yum-utils
2. 设置yum存储库，创建/etc/yum.repos.d/nginx.repo 写入以下内容 ：

# 将nginx官方源加入到yum源中

[nginx-stable]

name=nginx stable repo

baseurl=http://nginx.org/packages/centos/$releasever/$basearch/

gpgcheck=1

enabled=1

gpgkey=https://nginx.org/keys/nginx\_signing.key

module\_hotfixes=true

[nginx-mainline]

name=nginx mainline repo

baseurl=http://nginx.org/packages/mainline/centos/$releasever/$basearch/

gpgcheck=1

enabled=0

gpgkey=https://nginx.org/keys/nginx\_signing.key

module\_hotfixes=true

1. 默认使用稳定的nginx存储库。如要用主线nginx存储库，运行以下命令：

sudo yum-config-manager --enable nginx-mainline

四，安装Nginx：sudo yum install nginx

启动命令：[root@cpyy ~]# nginx

检查启动：[root@cpyy ~]# ps -ef |grep nginx

root 27312 1 0 13:43 ? 00:00:00 nginx: master process nginx

nginx 27313 27312 0 13:43 ? 00:00:00 nginx: worker process

root 27354 2736 0 13:44 pts/0 00:00:00 grep --color=auto nginx

就可通过网址访问：8.135.42.34

nginx配置文件地址：/etc/nginx/

重启nginx：nginx -s reload

## 部署Vue.js项目

注意：修改api地址

1. 构建：npm run build生成dist文件

2，可用WinSCP把dist文件打包.zip上传到centos服务器(推荐/mnt/www/trip-mobile/)

3，解压：unzip dist.zip没有工具先下载yum install unzip

4，配置nginx：

进入/etc/nginx/conf.d/，备份cp default.conf default.conf.bak

编辑vim default.conf,修改成上传的目录地址:

location / {

root /user/share/nginx/html; -> /mnt/www/trip-mobile/dist;

}

修改配置要重启，可先测试：nginx -t ，再重启nginx：nginx -s reload

访问网址8.135.42.34就可打开

## 关系型数据库MySQL

MyNewPass4!

官网：<https://www.mysql.com/>

如何用yum安装：<https://dev.mysql.com/doc/refman/8.0/en/linux-installation-yum-repo.html>

一，添加MySQL Yum存储库

1. 下载 (选择系统版本：[https://dev.mysql.com/downloads/repo/yum/](https://dev.mysql.com/downloads/repo/yum/" \t "https://dev.mysql.com/doc/refman/8.0/en/_top))

[root@cpyy ~]# wget <https://dev.mysql.com/get/mysql80-community-release-el8-1.noarch.rpm>

1. 安装：

[root@cpyy ~]# yum install mysql80-community-release-el8-1.noarch.rpm

1. 检查是否已成功添加MySQL Yum存储库

[root@cpyy ~]# yum repolist enabled | grep "mysql.\*-community.\*"

二，安装MySQL

[root@cpyy ~]# sudo yum install mysql-community-server

ps:禁用系统mysql，执行安装不行再用

yum module disable mysql

三，启动MySQL服务器

1，启动：[root@cpyy ~]# systemctl start mysqld

检查MySQL服务器的状态：[root@cpyy ~]# systemctl status mysqld

2，查看密码：[root@cpyy ~]# sudo grep 'temporary password' /var/log/mysqld.log

3，使用临时密码登录并更改密码 mysql -uroot -p

mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass4!';

4，设置可远程访问(要[开启防火墙](#_防火墙))：

mysql> update mysql.user set host='%' where user='root';

ps远程登录mysql可能会有不支持验证方式，可升级登录软件或修改mysql验证方式为5.7的

> ALTER USER 'root'@'%' IDENTIFIED WITH mysql\_native\_password BY '你的密码';

## 非关系型数据库Redis

官网：<https://redis.io/>

1，下载(v5.0)：[root@cpyy ~]# wget <https://download.redis.io/releases/redis-5.0.12.tar.gz>

2，解压：[root@cpyy ~]# tar -zxvf redis-5.0.12.tar.gz

3，安装编译器：[root@cpyy ~]# yum install gcc gcc-c++

4，进入解压后文件夹：cd redis-5.0.12 ps:README.md文件是安装教程

5，编译安装：[root@cpyy redis-5.0.12]# make && make install

6，修改配置文件,后台运行：vim redis.conf 下的daemonize改yes

7,启动redis服务器：[root@cpyy redis-5.0.12]# ./src/redis-server redis.conf

查看是否启动成功：ps -ef | grep redis

## Python3

官网：<https://www.python.org/>

安装：

1. 下载：wget <https://www.python.org/ftp/python/3.8.6/Python-3.8.6.tgz>
2. 解压：tar -zxvf Python-3.8.6.tgz
3. 安装编译器：yum install gcc gcc-c++
4. 安装依赖：

yum install zlib zlib-devel openssl openssl-devel

yum install glib2-devel pcre-devel bzip2-devel gzip-devel

yum install libffi-devel libjpeg libjpeg-devel

yum install freetype freetype-devel

5.进入解压后文件夹,检测：sh configure

6.编译安装：先make 再make install

## 部署Django项目

1，提取依赖：pip freeze > requirements.txt

2，用WinSCP上传到centos服务器(推荐/mnt/www/)

把python2替换成python3，进入/usr/bin/,备份里面的python2

mv python python.bak

ln -s python3 python

ln -s pip3 pip

修改yum：vim /usr/bin/yum第一行的python改成python2，就可

3，进入项目目录编辑setting.py，配置DEBUG，数据库，MEDIA\_URL=服务器地址

DEBUG = False

ALLOWED\_HOSTS = ['\*']

配置数据库密码MyNewPass4!，等

MEDIA\_URL = 'http://8.135.42.34/'

python的mysqlclient包可能会安装不成功，要先安装依赖：

yum install mysql-devel python-devel

yum install python3-devel

yum install mariadb-devel

再pip install -r requirements.txt

创建数据库：create database trip\_django charset=utf8;

python manage.py check

python manage.py migrate

## uWSGI

官网：<https://uwsgi-docs.readthedocs.io/en/latest/index.html>

安装文档：https://uwsgi-docs.readthedocs.io/en/latest/Install.html#installing-uwsgi

安装

1，使用pip安装uWSGI：[root@cpyy trip]# pip install uwsgi

2，配置uWSGI：https://uwsgi-docs.readthedocs.io/en/latest/Configuration.html#configuring-uwsgi

创建配置文件：vim /mnt/www/trip.ini (注释要清除)

[uwsgi]

socket = 127.0.0.1:9090 #uwsgi运行在那个地方 本机9090端口

master = 1 #启动主进程

processes = 1 #启动n个工作进程，生成指定数目的worker/进程

pidfile = /var/run/trip.pid #指定进程pid文件

daemonize = /var/log/uwsgi/trip.log #进程后台执行，并保存日志到指定的文件

module = trip.wsgi:application #加载的WSGI模块，项目里的wsgi.py文件

chdir = /mnt/www/trip #在app加载前切换到当前目录， 指定运行目录

listen = 100 #设置socket的监听队列大小，默认：100

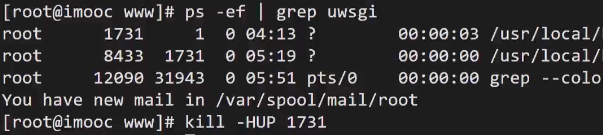
buffer-size = 32768 # 设置用于uwsgi包解析的内部缓存区大小

max-requests = 200 #每个工作进程设置的请求上限

新建uwsgi日志目录mkdir -p /var/log/uwsgi/再启动：[root@cpyy www]# uwsgi --ini trip.ini

3查看是否启动：ps -ef | grep uwsgi

如果修改了代码，要重启uwsgi，kill -HUP 1731,就可



配置nginx

进入/etc/nginx/conf.d/复制cp default.conf django.conf

编辑vim django.conf

1. server {

listen 8000;

}

2.location / {

uwsgi\_pass 127.0.0.1:9090; #同uwsgi配置文件的socket

uwsgi\_param UWSGI\_SCRIPT trip.wsgi; #用哪个脚本执行？项目的wsgi.py

include uwsgi\_params;

}

检查nginx -t 重启nginx -s reload

就可访问8.135.42.34:8000/admin

如果关闭防火墙 service firewalld stop也无法访问，在阿里云添加端口

## 配置慕旅游网(项目)

把/api开头的代理到127.0.0.1:8000(vue中vue.config.js配置的地址)

注意django把静态文件收集到 STATIC\_ROOT中没！（setting.py里要用STATIC\_ROOT，注销掉STATICFILES\_DIRS）

1，配置nginx反向代理：vim /etc/nginx/conf.d/default.conf 添加

location ^~/api {

proxy\_pass http://127.0.0.1:8000/;

}

就可，更多参数(带上请求头，远程IP地址等)

location ^~/api {

proxy\_pass http://127.0.0.1:8000/;

add\_header X-Slave $upstream\_addr;

proxy\_redirect off;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

}

2，配置后台(admin)访问，vim /etc/nginx/conf.d/default.conf 添加

location ^~/admin {

proxy\_pass http://127.0.0.1:8000/admin;

}

3，配置静态文件地址，vim /etc/nginx/conf.d/default.conf 添加

location ^~/static {

alias /mnt/www/trip/static;

}

重启nginx -s reload

# 测试

viewport

# css

调整居中等css

max-width: 640px;

left:50%!important;

transform:translateX(-50%);

# 代码

## 图片列表展示

 getImageList(){

<template>

<!-- 景点图片 -->

    <div class="page-sight-image">

        <!-- 页面头部 -->

        <van-nav-bar fixed left-text="返回" title="景点评论" left-arrow @click-left="goBack"/>

        <!-- //页面头部 -->

        <van-pull-refresh v-model="refreshing" @refresh="onRefresh">

            <van-list class="sight-image" v-model="loading" :finished="finished" finished-text="没有更多了" @load="getImageList" :error.sync="error" error-text="请求失败，点击重新加载">

                <van-row>

                    <van-col span="12" v-for="image in imagetList" :key="image.id">

                        <van-image @click="show=true" width="150" height="150" :src="image.img"/>

                    </van-col>

                </van-row>

            </van-list>

        </van-pull-refresh>

        <van-image-preview v-model="show" :images="imageUrls" @change="onChange" :start-position="0"/>

    </div>

</template>

<script>

import {ajax} from '@/utils/ajax.js'

import {SightApis} from '@/utils/apis'

export default {

    data(){

        return{

            id:"",

            imagetList:[],

            currentPage:1,

            loading: false,

            finished: false,

            error: false,

            refreshing:false,

            show:false,

            index: 0,

        }

    },

    computed:{

        imageUrls(){

            return Object.values(this.imagetList).map(i=>i.img)

        }

    },

    methods:{

        goBack(){

            this.$router.go(-1)

        },

        getImageList(){

            const url=SightApis.sightImageUrl.replace('#{id}',this.id)

            ajax.get(url,{params:{page:this.currentPage}}).then(({data})=>{

                console.log(data)

                this.imagetList=this.imagetList.concat(data.objects)

                this.loading=false

                this.refreshing=false

                this.currentPage=data.meta.current\_page+1

                if(data.meta.current\_page===data.meta.page\_count){

                    this.finished=true

                }

            }).catch(()=>{

                this.loading=false

                this.error = true

                this.refreshing=false

            })

        },

        onRefresh() {//下拉刷新

            this.imagetList=[] //清空数据

            this.currentPage=1

            this.finished = false//重置状态

            this.error=false

            this.loading = true

            this.getImageList()

        },

        onChange(index) {

            this.index = index;

        },

    },

    watch:{

        $route(){

            this.id=this.$route.params.id

            this.imagetList=[],

            this.currentPage=1,

            this.loading=false,

            this.finished=false,

            this.error=false,

            this.refreshing=false,

            this.show=false,

            this.index=0

        }

    },

    created() {

        this.id=this.$route.params.id

        // this.getImageList()

    },

}

</script>

<style lang="less">

.sight-image{

    margin: 56px 10px 0 10px;

    .van-col{

        margin-bottom: 10px;

    }

}

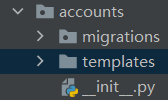
</style>

class sight\_imagelist(ListView):  
 paginate\_by = 5  
  
 def get\_queryset(self):  
 sight\_id = self.kwargs.get('pk', 0)  
 sight = Sight.objects.filter(pk=sight\_id, is\_valid=True).first()  
 if sight:  
 return sight.images.filter(is\_valid=True)  
 return Sight.objects.none()

def get\_paginate\_by(self, queryset):  
 page\_size = self.request.GET.get('limit', self.paginate\_by)  
 return page\_size

def render\_to\_response(self, context, \*\*response\_kwargs):  
 page\_obj = context['page\_obj']  
 if page\_obj is not None:  
 data = serializers.SightImageListSerializer(page\_obj).to\_dict()  
 return JsonResponse(data)  
 return NotFoundJsonResponse()

## django-auth用户登录



app下的templates文件，要在setting.py设置DIRS，'APP\_DIRS': True,就可

TEMPLATES = [  
 {  
 。。。

'DIRS': [BASE\_DIR / 'templates'],  
 'APP\_DIRS': True,

**@login\_required**装饰器，用户必须登录才能访问，和LOGIN\_URL搭配使用

@login\_required(login\_url=’’)或加在setting.py -> LOGIN\_URL = '/accounts/user/login/'

**form.py**

class Login\_Form(forms.Form):  
 username = forms.CharField(label='用户名', widget=forms.TextInput, max\_length=11, min\_length=11, strip=True)  
 password = forms.CharField(label='密码', widget=forms.PasswordInput)  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs): # 重写父类方法，增加user字段，保存用户  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
 self.user = None  
  
 def clean\_username(self):  
 username = self.cleaned\_data.get('username')  
 if not re.search(r'^1[0**-**9]{10}$', username):  
 raise forms.ValidationError('手机号格式不正确')  
 return username  
  
 def clean(self):  
 data = super().clean() # 调用父类获取表单数据  
 username = data.get('username', None)  
 password = data.get('password', None)  
 if self.errors:  
 return None

#用django-auth验证用户名，密码，返回用户对象，否则返回空  
 user = authenticate(username=username, password=password)  
 if not user:  
 raise forms.ValidationError('用户名或密码不正确')  
 elif not user.is\_active:  
 raise forms.ValidationError('该用户被禁用')  
 self.user = user #保存用户  
 return data  
  
 def do\_login(self, request):  
 # 自定方法，用户登录  
 login(request, self.user) # 调用django-auth登录，要给(request，用户对象)  
 self.user.last\_login = now() # 修改最后登录时间  
 self.user.save()  
 #TODO添加日志数据库  
 return self.user

**views.py**

def user\_login(request):  
 # 用户登录  
 if request.method == 'POST':  
 form = Login\_Form(request.POST) #把提交数据给form  
 if form.is\_valid(): #验证数据有效  
 form.do\_login(request) #登录  
 return redirect(reverse('accounts:user\_info'))  
 else:  
 forms = Login\_Form()  
 return render(request, 'user\_login.html', locals())  
  
  
@login\_required #登录验证  
def user\_info(request):  
 # 用户信息  
 return render(request, 'user\_info.html')  
  
  
def user\_logout(request):  
 logout(request) # 调用django-auth退出，要给(request)  
 return redirect(reverse('accounts:user\_info'))

**html**

return redirect(reverse('accounts:user\_info'))

调用django-auth登录完，就可直接在html用 用户对象 无需传参

<body>  
{{ user.username }}  
{% if user.is\_authenticated %} #is\_authenticated 用户是否登录  
<a href="{% url 'accounts:user\_logout' %}">退出登录</a>  
{% endif %}  
</body>

## 用户登录接口

post请求后台拿不到数据，用transformRequest处理

ajax.py

export const ajax=axios.create({

    headers:{

        'source':'h5',

        'Content-Type':'application/x-www-form-urlencoded'

    },

    transformRequest:function(data,headers){//处理发送参数

        return qs.stringify(data)

    },

    withCredentials:true

})

**view.py**

def user\_api\_login(request):  
 # 用户登录接口  
 if request.method == 'POST':  
 form = Login\_Form(request.POST) # 把数据传给form  
 if form.is\_valid():  
 user = form.do\_login(request) #调用登录函数，返回用户对象  
 data = { #制作返回JS信息  
 'user': UserSerializer(user).to\_dict(), #序列化  
 'profile': UserProfileSerializer(user.profile).to\_dict(),  
 }  
 return JsonResponse(data)  
 else: # 验证不通过  
 err = json.loads(form.errors.as\_json()) #form自带的错误JS返回，要再loads  
 return BadRequestJsonResponse(err) #自己写的返回函数  
 else: # 不是POST请求  
 return MethodNotAllowedJsonResponse()  
  
def user\_api\_logout(request):  
 # 用户退出接口  
 logout(request)  
 return HttpResponse(status=201)

**BadRequestJsonResponse**

class BadRequestJsonResponse(JsonResponse):  
 # 表单请求没有通过，错误显示  
 status\_code = 400  
 def \_\_init\_\_(self, err\_list=[], \*args, \*\*kwargs):  
 data = {  
 'error\_code': 400000,  
 'error\_msg': '参数格式不正确！',  
 'err\_list': err\_list}  
 super().\_\_init\_\_(data, \*args, \*\*kwargs)

**serializer.py**

choices字段可用get\_字段名\_display(),返回对应中文

class UserProfileSerializer(BaseSerializer):  
 # 用户详细信息  
 def to\_dict(self):  
 obj = self.obj  
 return {  
 'real\_name': obj.real\_name,  
 'sex': obj.get\_sex\_display()} #

//sex = models.SmallIntegerField('性别', choices=SEX\_CHOICES, default=2)

vue -- 登录

methods:{

        goBack(){

            this.$router.go(-1);

        },

        onSubmit(){

            // 调用接口

            ajax.post(AccountsApis.loginUrl,{

                username:this.username,

                password:this.password

            }).then(({data})=>{

                // 存到Vuex

                this.$store.commit('updateUserInfo',data)

                this.$toast('登录成功')//提示信息

                this.$router.replace({name:'mine'})//路由跳转

            }).catch(({response:{data}})=>{

                // 异常处理

                this.$toast(`${data.error\_code},${data.error\_msh}`);//提示信息

            })

        },

    }

vue -- 个人中心

<script>

import {AccountsApis} from '@/utils/apis'

import {ajax} from '@/utils/ajax'

import TripFooter from '@/components/common/footer'

import {mapState} from 'vuex'

export default {

    components: {

        TripFooter,

    },

    computed:mapState(['user','profile']), //快捷访问vuex -> state

    methods:{

        getUserInfo(){ //刷新，vuex里的数据就会重置

            if(!this.profile.sex){ //如果里面没数据

                ajax.get(AccountsApis.UserInfo).then(({data})=>{ //访问接口

                    this.$store.commit('updateUserInfo',data) //重新添加vuex数据

                })

            }

        },

        logout(){ //登出

            ajax.get(AccountsApis.logoutUrl).then(()=>{

                this.$store.commit('deleteUserInfo') //调用vuex方法，清空vuex数据

                this.$router.push({name:'Home'}) //跳到首页

                this.$notify({ //提示

                    message:'退出成功',

                    type:'success'

                })

            })

        }

    },

    mounted(){ //挂在后

        this.getUserInfo()

    },

}

vuex

export default new Vuex.Store({

  state: {

    user:{},

    profile:{}

  },

  mutations: {

    updateUserInfo(state,{user,profile}){

      this.state.user={

        ...state.user,

        ...user

      }

      this.state.profile={

        ...state.profile,

        ...profile

      }

    },

    deleteUserInfo(){ //登出方法

      this.state.user={}

      this.state.profile={}

    }

  },

  actions: {

  },

  modules: {

  }

})

## 用户详情cbv

用django-login登录过的，可直接request.user获取用户对象

class UserDetailView(View):  
 # 用户详细信息  
 def get(self, request):  
 user = request.user #获取用户对象  
 if user.is\_authenticated: #判断用户是否登录  
 data = {  
 'user': UserSerializer(user).to\_dict(), #序列化  
 'profile': UserProfileSerializer(user.profile).to\_dict(),  
 }  
 return JsonResponse(data)  
 else:  
 return UnauthorizedJsonResponse() #没登录，返回错误JS

## 用户注册接口cbv

from django.db import transaction

@transaction.atomic 同时添加两个表，开启数据库事务机制

遇到问题：注册用户密码为明文

view.py

class UserRegisterView(FormView):  
 form\_class = RegisterForm  
 http\_method\_names = ['post'] # 接受哪些请求  
 def form\_valid(self, form):  
 # 表单通过的方法  
 result = form.do\_register(request=self.request)  
 if result:  
 user, profile=result  
 data = {  
 'user': UserSerializer(user).to\_dict(),  
 'profile': UserProfileSerializer(user.profile).to\_dict(),  
 }  
 return JsonResponse(data)  
 return ServerErrorJsonResponse()  
 def form\_invalid(self, form):  
 # 表单没有通过方法  
 # json.loads(form.errors.as\_json()) 表单错误信息并转成JS  
 err\_list = json.loads(form.errors.as\_json())  
 return BadRequestJsonResponse(err\_list)

forms.py

class RegisterForm(forms.Form):  
 username = forms.CharField(label='手机号', max\_length=11, min\_length=11, required=True,  
 error\_messages={'required': '请输入手机号'})  
 password = forms.CharField(label='密码', max\_length=128, widget=forms.PasswordInput, required=True,  
 error\_messages={'required': '请输入密码'})  
 nickname = forms.CharField(label='昵称', max\_length=16,required=False)  
 sms\_code = forms.CharField(label='验证码', max\_length=6, min\_length=6, required=True,  
 error\_messages={'required': '请输入验证码'})  
  
 def clean\_username(self):  
 username = self.cleaned\_data.get('username')  
 if not re.search(r'^1[0**-**9]{10}$', username):  
 raise forms.ValidationError('手机号格式不正确')  
 if User.objects.filter(username=username).exists():  
 raise forms.ValidationError('手机号已注册')  
 return username  
  
 def clean(self):  
 data = super().clean()  
 if self.errors:  
 return  
 phone\_num = self.cleaned\_data.get('username', None)  
 sms\_code = self.cleaned\_data.get('sms\_code', None)  
 code = cache.get('sms\_code\_{}'.format(phone\_num))  
 if code is None:  
 raise forms.ValidationError('验证码已过期')  
 if str(code) != sms\_code:  
 raise forms.ValidationError('验证码错误')  
 return data  
  
 @transaction.atomic # 事务机制  
 def do\_register(self, request):  
 # 注册  
 data = self.cleaned\_data  
 version = request.headers.get('version', None) # 从请求头获取版本号  
 source = request.headers.get('source', None) # 从请求头获取登陆来源  
 try:  
 # 添加到用户表  
 user = User.objects.create(username=data.get('username', None),  
 password=data.get('password', None),  
 nickname=data.get('nickname', None),  
 last\_login=now()) # 自带方法now()为当前时间  
 # 添加到用户详情表  
 profile = Profile.objects.create\_user(user=user, username=data.get('username', None),  
 version=version, source=source)  
 login(request, user) # 调用自带登录方法  
 user.save()  
 profile.save()  
 # 添加到日志表  
 user.add\_login\_record(username=user.username, ip=request.META.get('REMOTE\_ADDR'),  
 version=version, source=source)  
 return user, profile  
 except Exception as e:  
 print(e)  
 return

vue

template>

<!-- 注册 -->

  <div class="page-account-register">

    <!-- 导航条 -->

    <van-nav-bar title="用户注册" left-arrow @click-left="goBack"/>

    <!-- 表单输入 -->

    <van-form @submit="onSubmit">

        <van-field v-model="username" type="tel" label="手机号" placeholder="手机号" :rules="ruleName" maxlength="11" clearable/>

        <van-field v-model="sms\_code" center clearable label="短信验证码" placeholder="短信验证码" :rules="[{required: true, message: '请输入验证码'}]">

            <template #button>

                <SendSmsCode :phoneNum='username'/>

            </template>

        </van-field>

        <van-field v-model="nickname" label="昵称" placeholder="昵称" maxlength="12" clearable/>

        <van-field v-model="password" type="password" label="密码" placeholder="密码" :rules="[{ required: true, message: '请填写密码' }]" clearable/>

        <van-field v-model="repassword" type="password" label="确认密码" placeholder="确认密码" :rules="rulepassword" clearable/>

        <div style="margin: 16px;">

            <van-button round block type="info" native-type="submit">

            提交

            </van-button>

        </div>

    </van-form>

    <!-- 文字提示 -->

    <div class="tips">

        注册表示同意 <a href="#">用户使用协议</a>及<a href="#">隐私条款</a>

    </div>

    <div class="tips">

        已有账号？ <router-link :to="{name:'accountLogin'}">点击登录>></router-link>

    </div>

    <!-- 版权 -->

    <Copyright/>

  </div>

</template>

<script>

import {AccountsApis} from '@/utils/apis'

import {ajax} from '@/utils/ajax'

import \* as types from '@/store/mutation-types'

import Copyright from '@/components/common/Copyright'

import SendSmsCode from '@/components/common/SendSmsCode'

export default {

    components:{

        Copyright,

        SendSmsCode

    },

    data(){

        return{

            ruleName:[

                { required: true, message: '请填写用户名' },

                {pattern:/1\d{10}/,message: '手机号不正确'},

            ],

            rulepassword:[

                { required: true, message: '请再次输入密码'},

                {validator:()=>this.password===this.repassword,message:'两次密码不一致'}

            ],

            username:'',

            sms\_code:'',

            nickname:'',

            password:'',

            repassword:'',

        }

    },

    methods:{

        goBack(){

            this.$router.go(-1);

        },

        onSubmit(){

            ajax.post(AccountsApis.registerUrl,{

                username:this.username,

                password:this.password,

                nickname:this.nickname,

                sms\_code:this.sms\_code

            }).then(({data})=>{

                this.$store.commit(types.UPDATE\_USER\_INFO,data),//用户信息写入vuex

                this.$notify({ //提示消息

                    message:'注册成功',

                    type:'success'

                })

                this.$router.replace({name:'mine'}) //跳转

            })

        },

    }

}

</script>

## 验证码接口cbv+Redis

view.py

class SmsCodeView(FormView):  
 form\_class = SendSmsCodeForm  
 def form\_valid(self, form):  
 # 表单通过的方法  
 data = form.send\_sms\_code()  
 if data:  
 return JsonResponse(data, status=201)  
 return ServerErrorJsonResponse()  
 def form\_invalid(self, form):  
 # 表单没有通过方法  
 err\_list = json.loads(form.errors.as\_json())  
 return BadRequestJsonResponse(err\_list)

form.py

class SendSmsCodeForm(forms.Form):  
 phone\_num = forms.CharField(label='手机号', required=True, error\_messages={'required': '请输入手机号'})  
 def clean\_phone\_num(self):  
 phone\_num = self.cleaned\_data.get('phone\_num', None) # 获取手机号  
 if not re.search(r'^1[0**-**9]{10}$', phone\_num): # 正则校验  
 raise forms.ValidationError('手机号不正确')  
 return phone\_num  
 def send\_sms\_code(self):  
 timeout = 60 \* 5  
 sms\_code = random.randint(100000, 999999)  
 phone\_num = self.cleaned\_data.get('phone\_num', None) # 获取手机号  
 try:  
 # TODO调用第三方发短信  
 key = 'sms\_code\_{}'.format(phone\_num) # 设置redis的key  
 cache.set(key, sms\_code, timeout=timeout) # 保存到redis  
 return {  
 'phone\_num': phone\_num,  
 'sms\_code': sms\_code,  
 'timeout': timeout  
 }  
 except Exception as e:  
 print(e)  
 return None

vue

export default {

    props:['phoneNum'],

    data() {

        return {

            sendBtnText: '发送验证码',//文本内容

            counter:5,//等待秒数

            isSmsSend:false,

        };

    },

    methods:{

        countDown(){

            var timer=setInterval(()=>{//计时器 每秒设置文本内容

                this.sendBtnText=`${this.counter}s 后重新发送`;

                this.counter--;

                if(this.counter<0){

                    clearInterval(timer)//清除计时器

                    this.counter=5;//重置属性

                    this.sendBtnText='发送验证码';

                    this.isSmsSend=false;

                }

            },1000)

        },

        sendSmsCode(){

            //   判断手机号不为空，并验证有效

            if(!this.phoneNum || this.phoneNum.search(/1\d{10}/)==-1){

                this.$notify('请输入手机号');

                return false

            }

            //   调用接口，发送验证码

            ajax.post(SystemApis.sendSmsCodetUrl,{

                phone\_num:this.phoneNum

            }).then(({data})=>{

                this.$notify({ //提示用户已发送

                    message:`验证码为:${data.sms\_code}，${data.timeout/60}分钟内有效`,

                    duration:1000\*6,

                    type:'success',

                })

            }).catch(err=>{ //错误处理

               console.log(err) //TODO错误时重置属性

            })

            this.isSmsSend=true;

            //   倒计时

            this.countDown();

        },

    },

};

</script>

## 门票下单接口cbv

#select\_related('sight')，把反向引用一起查，节省查询次数

@transaction.atomic # 事务机制

views.py

@method\_decorator(login\_required, name='dispatch')#登录验证，在dispath时  
class TicketOrderSubmitView(FormView):  
 form\_class = SubmitTicketOrderForm  
 http\_method\_names = ['post'] #限制请求  
  
 def form\_valid(self, form):  
 # 通过的  
 obj = form.save(user=self.request.user)  
 return JsonResponse({'sn': obj.sn}, status=201) #返回订单号  
  
 def form\_invalid(self, form):  
 # 不通过的  
 err = json.loads(form.errors.as\_json())  
 return BadRequestJsonResponse(err)

form.py

class SubmitTicketOrderForm(forms.ModelForm):  
 ticket\_id = forms.IntegerField(label='门票ID', required=True)  
 play\_date = forms.DateField(label='出行日期', required=True)  
  
 class Meta:  
 model = Order  
 fields = ['to\_user', 'to\_phone', 'buy\_count'] #需要的字段  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs): # 重写父类，增加保存ticket对象  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
 self.ticket = ''  
 self.buy\_count = ''  
  
 def clean\_ticket\_id(self):  
 self.buy\_count = self.cleaned\_data.get('buy\_count') # 保存购买数量  
 ticket\_id = self.cleaned\_data.get('ticket\_id', None)

#select\_related('sight')，把反向引用一起查，节省查询次数  
 self.ticket = Ticket.objects.select\_related('sight').filter(is\_valid=True, pk=ticket\_id).first()  
 if not self.ticket:  
 raise forms.ValidationError('门票信息不存在')  
 else:  
 # 如果剩余门票小于购买数  
 if self.ticket.remain\_stock < self.buy\_count:  
 raise forms.ValidationError('剩余门票不足')  
 return ticket\_id  
  
 @transaction.atomic # 事务机制  
 def save(self, user, commit=False): # 重写父类保存方法，增加user对象  
 obj = super().save(commit=commit) # 首先调用父类保存方法  
 obj.user = user  
 obj.sn = gen\_trans\_id() # 生成订单号方法  
 obj.buy\_amount = self.ticket.sell\_price \* self.buy\_count # 计算价格  
 obj.save()  
 self.ticket.remain\_stock = F('remain\_stock') - self.buy\_count # 剩余门票减购买数  
 self.ticket.save()  
 # 关联订单详情，保存快照  
 ctype = ContentType.objects.get\_for\_model(Ticket)  
 OrderItem.objects.create( #保存快照到数据库  
 user=user,  
 order=obj,  
 flash\_name=self.ticket.name,  
 flash\_img=self.ticket.sight.main\_img,  
 flash\_price=self.ticket.sell\_price,  
 flash\_origin\_price=self.ticket.price,  
 flash\_discount=self.ticket.discount,  
 count=self.buy\_count,  
 amount=obj.buy\_amount,  
 content\_type=ctype,  
 object\_id=self.ticket.id,  
 remark=f'出行日期：{self.cleaned\_data.get("play\_date")}'  
 )  
 return obj

## 订单详情接口RESTful，cbv

RESTful：使用POST，DELETE，PUT和GET四种请求对应增删改查

views.py

@method\_decorator(login\_required, name='dispatch') # 登录验证，在dispatch时  
class OrderDetail(BaseDetailView):  
 slug\_field = 'sn' # 按sn订单号查询  
 slug\_url\_kwarg = 'sn'  
  
 def get\_queryset(self): #查找数据  
 user = self.request.user  
 return Order.objects.filter(user=user, is\_valid=True)  
  
 def get(self, request, \*args, \*\*kwargs): # get获取订单详情  
 # self.get\_object(),会根据‘sn’，从get\_queryset中找到对应的数据  
 order\_obj = self.get\_object()  
 data = OrderDetailSerializer(order\_obj).to\_dict()  
 return JsonResponse(data)  
  
 @transaction.atomic #多张表，要事务机制  
 def post(self, required, \*args, \*\*kwargs): # post订单支付  
 # 选择支付方式  
 # 数据验证  
 order\_obj = self.get\_object()  
 if order\_obj.status == OrderStatus.SUBMIT:#如果是未支付  
 # 调用支付  
 # 修改订单状态  
 order\_obj.status = OrderStatus.PAID #改为已支付  
 order\_obj.save()  
 order\_obj.order\_items.update(status=OrderStatus.PAID)  
 return JsonResponse({}, status=201)  
 return JsonResponse({}, status=200)  
  
 @transaction.atomic  
 def put(self, required, \*args, \*\*kwargs): # put取消订单  
 order\_obj = self.get\_object()  
 if order\_obj.status == OrderStatus.SUBMIT:  
 order\_obj.status = OrderStatus.CANCELED  
 order\_obj.save()  
 items = order\_obj.order\_items.filter(status=OrderStatus.SUBMIT)  
 for item in items:  
 obj = item.content\_object # 复合关联  
 obj.remain\_stock = F('remain\_stock') + item.count # 剩余票数+退票  
 obj.save()  
 items.update(status=OrderStatus.CANCELED)  
 return JsonResponse({}, status=201)  
 return JsonResponse({}, status=200)  
  
 @transaction.atomic  
 def delete(self, required, \*args, \*\*kwargs): # delete删除订单  
 order\_obj = self.get\_object()  
 if order\_obj.status != OrderStatus.SUBMIT: # 已支付,已取消可删除订单  
 order\_obj.is\_valid = False  
 order\_obj.save()  
 order\_obj.order\_items.update(is\_valid=False)  
 return JsonResponse({}, status=201)  
 return JsonResponse({}, status=200)

serializer.py

class OrderItemSerializer(BaseSerializer):  
 def to\_dict(self):  
 obj = self.obj  
 return {  
 'pk': obj.pk,  
 ...

'app\_label': obj.content\_type.app\_label,#复合类型对应的app

'model': obj.content\_type.model, #复合类型对应的model  
 }  
  
  
class OrderDetailSerializer(BaseSerializer):  
 def to\_dict(self):  
 obj = self.obj  
 items = []  
 for item in obj.order\_items.all():  
 items.append(OrderItemSerializer(item).to\_dict())#调用上面的序列化类  
 return {  
 'sn': obj.sn,  
 'status': obj.get\_status\_display(), #将choice转成明文  
 ...

'items': items  
 }

## 我的订单接口cbv

view.py

@method\_decorator(login\_required, name='dispatch')  
class OrderListView(ListView):#继承同样视图ListView  
 paginate\_by = 5 #设置每页大小  
  
 def get\_queryset(self):  
 user = self.request.user  
 query = Q(user=user, is\_valid=True)  
 status = self.request.GET.get('status', None)  
 if status and status != '0':  
 query = query & Q(status=status)  
 return Order.objects.filter(query)  
  
 def render\_to\_response(self, context, \*\*response\_kwargs):  
 # 重写返回方法  
 page\_obj = context['page\_obj']  
 if page\_obj is not None:  
 data = OrderListSerializer(page\_obj).to\_dict()  
 return JsonResponse(data)  
 return NotFoundJsonResponse()  
  
 def get\_paginate\_by(self, queryset): # 可根据参数limit控制分页大小  
 page\_size = self.request.GET.get('limit', self.paginate\_by)  
 return page\_size

serializer.py

class OrderListSerializer(BaseListPageSerializer):  
 def get\_obj(self, obj):  
 img = obj.order\_items.first().flash\_img.url  
 return {  
 'sn': obj.sn,  
 'buy\_amount': obj.buy\_amount,  
 'buy\_count': obj.buy\_count,  
 'types': obj.get\_types\_display(),  
 'status': obj.get\_status\_display(),  
 'created\_at': obj.created\_at,  
 'remark': obj.remark,  
 'img': img}

## 门票详情接口cbv

view.py

class ticket\_detail(DetailView):  
  
 def get\_queryset(self):  
 return Ticket.objects.filter(is\_valid=True)  
  
 def render\_to\_response(self, context, \*\*response\_kwargs):  
 page\_obj=context['object']  
 if page\_obj is not None:  
 data = serializers.TicketDetailSerializer(page\_obj).to\_dict()  
 return JsonResponse(data)  
 return NotFoundJsonResponse()

serializer.py

class TicketDetailSerializer(BaseSerializer):  
 def to\_dict(self):  
 obj = self.obj  
 return {'id': obj.id,  
 ...

'remark': obj.remark}

## 统计报表

view.py

要点：获取前几天日期calc\_time = now\_time - timedelta(days=i)及created\_at\_\_date

聚合函数：order\_list.aggregate(amount=Sum('buy\_amount'))['amount']

from datetime import datetime, timedelta  
  
from django.contrib.auth.decorators import login\_required  
from django.db.models import Sum, Q  
from django.shortcuts import render  
  
from django.utils.timezone import now  
  
from accounts.models import Profile, User  
from order.choices import OrderStatus  
from order.models import Order  
from sight.models import Sight, Comment  
  
def get\_data\_count(start=None, end=None): # 今天，昨天数据统计  
 query = Q()  
 if start:  
 query = query & Q(created\_at\_\_gte=start) # get >=  
 if end:  
 query = query & Q(created\_at\_\_lte=end) # lte <=  
 order\_list = Order.objects.filter(is\_valid=True, status=OrderStatus.PAID).filter(query)  
 user\_list = Profile.objects.select\_related('user').filter(user\_\_is\_active=True).filter(query)  
 return {  
 'order\_amount': order\_list.aggregate(amount=Sum('buy\_amount'))['amount'], # 销售额  
 'order\_count': order\_list.count(), # 支付订单数  
 'user\_add\_count': user\_list.count(), # 新增用户数  
 # 下单用户数 取用户字段，去重，统计  
 'order\_user\_count': order\_list.values('user').distinct().count()  
 }  
  
def get\_latest\_order\_stats(days=7): # 最近几天数据统计  
 now\_time = now()  
 date\_array, amount\_array, count\_array = [], [], []  
 for i in range(days, 0, -1):  
 # 日期  
 calc\_time = now\_time - timedelta(days=i)  
 date\_array.append(f'{calc\_time.day}号')  
 queryset = Order.objects.filter(is\_valid=True, status=OrderStatus.PAID, created\_at\_\_date=calc\_time.date())  
 # 订单金额  
 result = queryset.aggregate(amount=Sum('buy\_amount'))  
 # result = queryset.aggregate(amount=Sum('buy\_amount'),count=Sum('buy\_count'))  
 amount\_array.append(result['amount'] or 0)  
 # 订单数量  
 count\_array.append(queryset.count())  
 # count\_array.append(len(queryset))  
 return {  
 'date': date\_array,  
 'amount': amount\_array,  
 'count': count\_array  
 }  
  
@login\_required(login\_url='/admin/login/') # 登录验证 使用自带的登录  
def index(request):  
 # 数据统计  
 total\_stats = {  
 'sight\_count': Sight.objects.filter(is\_valid=True).count(), # 景点总数  
 'comment\_count': Comment.objects.filter(is\_valid=True).count(), # 评价总数  
 'user\_count': User.objects.filter(is\_active=True).count(), # 用户总量  
 'order\_count': Order.objects.filter(is\_valid=True, status=OrderStatus.PAID).count(), # 订单总量  
 }  
 # 今日数据  
 now\_time = now()  
 # print(now\_time.strftime('%Y-%m-%d'))  
 now\_stats = get\_data\_count(start=datetime(now\_time.year, now\_time.month, now\_time.day))  
 # 昨日数据  
 yesterday = now\_time - timedelta(days=1) # 获取昨天=今天-1  
 yesterday\_stats = get\_data\_count(  
 start=datetime(yesterday.year, yesterday.month, yesterday.day),  
 end=datetime(now\_time.year, now\_time.month, now\_time.day)  
 )  
 # 数据走势  
 latest\_stats = get\_latest\_order\_stats()  
 return render(request, 'master/index.html', locals())

return render(request, 'master/index.html', locals())

html

{% load static %}  
<!DOCTYPE html>  
<html lang="zh-CN">  
  
<head>  
 <meta charset="utf-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <!-- 上述3个meta标签\*必须\*放在最前面，任何其他内容都\*必须\*跟随其后！ -->  
 <title>慕旅游网-后台管理</title>  
  
 <!-- Bootstrap -->  
 <link href="{% static 'plugins/bootstrap/css/bootstrap.min.css' %}" rel="stylesheet">  
  
 <!-- HTML5 shim 和 Respond.js 是为了让 IE8 支持 HTML5 元素和媒体查询（media queries）功能 -->  
 <!-- 警告：通过 file:// 协议（就是直接将 html 页面拖拽到浏览器中）访问页面时 Respond.js 不起作用 -->  
 <!--[if lt IE 9]>  
 <script src="{% static 'plugins/html5shiv.min.js' %}"></script>  
 <script src="{% static 'plugins/respond.min.js' %}"></script>  
 <![endif]-->  
  
 <link rel="stylesheet" href="{% static 'style/index.css' %}">  
</head>  
  
<body class="layout page-home" id="top">  
  
<div class="container-fluid">  
 <div class="layout-left"></div>  
 <div class="layout-right">  
 <!-- 数据统计 -->  
 <div class="panel-info">  
  
 <h3>数据统计</h3>  
 <div class="row">  
 <div class="col-md-3">  
 <div class="widget-card card\_\_blue am-cf">  
 <div class="card-header">景点总数</div>  
 <div class="card-body">  
 <div class="card-value">{{ total\_stats.sight\_count }}</div>  
 <div class="card-description">当前景点总数量</div>  
 <span class="card-icon glyphicon glyphicon-flag"></span>  
 </div>  
 </div>  
 </div>  
 <div class="col-md-3">  
 <div class="widget-card card\_\_primary am-cf">  
 <div class="card-header">评价总数</div>  
 <div class="card-body">  
 <div class="card-value">{{ total\_stats.comment\_count }}</div>  
 <div class="card-description">景点评价总数</div>  
 <span class="card-icon glyphicon glyphicon-comment"></span>  
 </div>  
 </div>  
 </div>  
 <div class="col-md-3">  
 <div class="widget-card card\_\_red am-cf">  
 <div class="card-header">用户总量</div>  
 <div class="card-body">  
 <div class="card-value">{{ total\_stats.user\_count }}</div>  
 <div class="card-description">当前用户总数量</div>  
 <span class="card-icon glyphicon glyphicon-user"></span>  
 </div>  
 </div>  
 </div>  
 <div class="col-md-3">  
 <div class="widget-card card\_\_violet am-cf">  
 <div class="card-header">订单总量</div>  
 <div class="card-body">  
 <div class="card-value">{{ total\_stats.order\_count }}</div>  
 <div class="card-description">已付款订单总数量</div>  
 <span class="card-icon glyphicon glyphicon-jpy"></span>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
  
 <div class="panel-info">  
 <h3>实时数据</h3>  
 <div class="row">  
 <div class="col-md-3">  
 <div class="widget-outline dis-flex flex-y-center">  
 <div class="outline-left">  
 <i class="glyphicon glyphicon-jpy"></i>  
 </div>  
 <div class="outline-right dis-flex flex-dir-column flex-x-between">  
 <div style="color: rgb(102, 102, 102); font-size: 1.3rem;">销售额(元)</div>  
 <div style="color: rgb(51, 51, 51); font-size: 2.4rem;">{{ now\_stats.order\_amount|default\_if\_none:0 }}</div>  
 <div style="color: rgb(153, 153, 153); font-size: 1.2rem;">  
 昨日：{{ yesterday\_stats.order\_amount|default\_if\_none:0 }}  
 </div>  
 </div>  
 </div>  
 </div>  
 <div class="col-md-3">  
 <div class="widget-outline dis-flex flex-y-center">  
 <div class="outline-left">  
 <i class="glyphicon glyphicon-ok"></i>  
 </div>  
 <div class="outline-right dis-flex flex-dir-column flex-x-between">  
 <div style="color: rgb(102, 102, 102); font-size: 1.2rem;">支付订单数</div>  
 <div style="color: rgb(51, 51, 51); font-size: 2.4rem;">{{ now\_stats.order\_count }}</div>  
 <div style="color: rgb(153, 153, 153); font-size: 1.2rem;">  
 昨日：{{ yesterday\_stats.order\_count }}  
 </div>  
 </div>  
 </div>  
 </div>  
 <div class="col-md-3">  
 <div class="widget-outline dis-flex flex-y-center">  
 <div class="outline-left">  
 <i class="glyphicon glyphicon-user"></i>  
 </div>  
 <div class="outline-right dis-flex flex-dir-column flex-x-between">  
 <div style="color: rgb(102, 102, 102); font-size: 1.3rem;">新增用户数</div>  
 <div style="color: rgb(51, 51, 51); font-size: 2.4rem;">{{ now\_stats.user\_add\_count }}</div>  
 <div style="color: rgb(153, 153, 153); font-size: 1.2rem;">  
 昨日：{{ yesterday\_stats.user\_add\_count }}  
 </div>  
 </div>  
 </div>  
 </div>  
 <div class="col-md-3">  
 <div class="widget-outline dis-flex flex-y-center">  
 <div class="outline-left">  
 <i class="glyphicon glyphicon-tags"></i>  
 </div>  
 <div class="outline-right dis-flex flex-dir-column flex-x-between">  
 <div style="color: rgb(102, 102, 102); font-size: 1.2rem;">下单用户数</div>  
 <div style="color: rgb(51, 51, 51); font-size: 2.4rem;">{{ now\_stats.order\_user\_count }}</div>  
 <div style="color: rgb(153, 153, 153); font-size: 1.2rem;">  
 昨日：{{ yesterday\_stats.order\_user\_count }}  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
 </div>  
  
 <div class="panel-info">  
 <h3>数据走势</h3>  
 <div class="row">  
 <div class="col-md-6">  
 <div id="dayCount" class="widget-echarts"></div>  
 </div>  
 <div class="col-md-6">  
 <div id="dayAmount" class="widget-echarts"></div>  
 </div>  
 </div>  
 </div>  
  
 </div>  
</div>  
  
<!-- jQuery (Bootstrap 的所有 JavaScript 插件都依赖 jQuery，所以必须放在前边) -->  
<script src="{% static 'plugins/jquery-3.4.1.min.js' %}"></script>  
<!-- 加载 Bootstrap 的所有 JavaScript 插件。你也可以根据需要只加载单个插件。 -->  
<script src="{% static 'plugins/bootstrap/js/bootstrap.min.js' %}"></script>  
<script src="{% static 'plugins/echarts.min.js' %}"></script>  
<script>  
 // 基于准备好的dom，初始化echarts实例  
 var ***dayCount*** = echarts.init(***document***.getElementById('dayCount'));  
  
 // 指定图表的配置项和数据  
 var ***dayCountOpts*** = {  
 title: {  
 text: '近七日订单数量增长趋势'  
 },  
 xAxis: {  
 type: 'category',  
 data: {{ latest\_stats.date|safe }}  
 },  
 yAxis: {  
 type: 'value'  
 },  
 series: [{  
 data: {{ latest\_stats.count }},  
 type: 'line'  
 }]  
 };  
  
 // 使用刚指定的配置项和数据显示图表。  
 ***dayCount***.setOption(***dayCountOpts***);  
  
  
 // 基于准备好的dom，初始化echarts实例  
 var ***dayAmount*** = echarts.init(***document***.getElementById('dayAmount'));  
  
 // 指定图表的配置项和数据  
 var ***dayAmountOpts*** = {  
 title: {  
 text: '近七日订单总额'  
 },  
 tooltip: {},  
 legend: {  
 data: ['金额']  
 },  
 xAxis: {  
 data: {{ latest\_stats.date|safe }}  
 },  
 yAxis: {},  
 series: [{  
 name: '金额',  
 type: 'bar',  
 data: {{ latest\_stats.amount }}  
 }]  
 };  
  
 // 使用刚指定的配置项和数据显示图表。  
 ***dayAmount***.setOption(***dayAmountOpts***);  
</script>  
</body>  
  
</html>

## 工具

### 用户登录校验

装饰器

from functools import wraps  
from utils.response import UnauthorizedJsonResponse  
  
def login\_required(view\_func):  
 *"""登录校验，没有登录返回401"""* @wraps(view\_func) # wraps不改变使用装饰器原有函数的结构(如name, doc)  
 def \_wrapped\_view(request, \*args, \*\*kwargs):  
 user = request.user  
 if not user.is\_authenticated:  
 return UnauthorizedJsonResponse()  
 return view\_func(request, \*args, \*\*kwargs)  
  
 return \_wrapped\_view

### 生成订单号

import random  
from django.utils.timezone import now  
  
def gen\_trans\_id(date=None):  
 *"""生成订单号"""* if date is None:  
 date = now()  
 str\_date = date.strftime('%Y%m%d%H%M%S%f') #年月日时分秒毫秒  
 str\_random = random.randint(1000, 9999)  
 return f'{str\_date}{str\_random}'

## 订单

### view

@method\_decorator(login\_required, name='dispatch') # 类装饰器 登录验证

import json  
from django import http  
from django.db import transaction  
from django.db.models import F, Q  
from django.shortcuts import render  
  
from django.utils.decorators import method\_decorator  
from django.views.generic import FormView, ListView  
from django.views.generic.detail import BaseDetailView  
  
from order import serializers  
from order.choices import OrderStatus  
from order.forms import SubmitTicketOrderForm  
from order.models import Order  
from sight.models import Ticket  
from utils.response import BadRequestJsonResponse, NotFoundJsonResponse  
from utils.views import login\_required  
  
  
@method\_decorator(login\_required, name='dispatch') # 类装饰器 登录验证  
class TicketOrderSubmitView(FormView): #订单提交  
 # 验证用户是否已登录  
 # 数据验证(门票ID，库存)  
 # 关联用户，生成订单号，计算价格，生成订单  
 # 返回内容：订单ID  
 form\_class = SubmitTicketOrderForm  
 http\_method\_names = ['post']  
  
 def form\_valid(self, form):  
 obj = form.save(user=self.request.user)  
 return http.JsonResponse({  
 'sn': obj.sn  
 }, status=201)  
  
 def form\_invalid(self, form):  
 err = json.loads(form.errors.as\_json())  
 return BadRequestJsonResponse(err)  
  
  
@method\_decorator(login\_required, name='dispatch') # 类装饰器 登录验证  
class OrderDetailView(BaseDetailView): #订单详情(获取，支付，取消，删除)  
 slug\_field = 'sn'  
 slug\_url\_kwarg = 'sn'  
  
 def get\_queryset(self): # 获取用户的订单列表  
 user = self.request.user  
 return Order.objects.filter(user=user, is\_valid=True)  
  
 def get(self, request, \*args, \*\*kwargs):  
 *"""GET订单详情"""* order\_obj = self.get\_object() # get\_object会从get\_queryset中找到对应的sn返回  
 data = serializers.OredeDetailSerializer(order\_obj).to\_dict()  
 return http.JsonResponse(data)  
  
 @transaction.atomic # 事务控制  
 def post(self, request, \*args, \*\*kwargs):  
 *"""POST订单支付"""* # TODO选择支付方式  
 order\_obj = self.get\_object()  
 # 数据验证，状态判断  
 if order\_obj.status == OrderStatus.SUBMIT:  
 # TODO调用支付接口  
 # 改变订单状态  
 order\_obj.status = OrderStatus.PAID  
 order\_obj.save()  
 order\_obj.order\_items.update(status=OrderStatus.PAID) # 批量更新  
 return http.JsonResponse({'msg': '支付成功'}, status=201)  
 return http.JsonResponse({'msg': '支付失败'}, status=200)  
  
 @transaction.atomic # 事务控制  
 def put(self, request, \*args, \*\*kwargs):  
 *"""PUT取消订单"""* # 获取订单对象  
 order\_obj = self.get\_object()  
 # 数据验证，状态判断  
 if order\_obj.status == OrderStatus.SUBMIT:  
 # 改变状态  
 order\_obj.status = OrderStatus.CANCELED  
 order\_obj.save()  
 # 回退库存  
 items = order\_obj.order\_items.filter(status=OrderStatus.SUBMIT) # 查询订单中商品状态为未支付的  
 for i in items: # 循环商品，复合关联门票表，回退库存  
 obj = i.content\_object  
 obj.remain\_stock = F('remain\_stock') + i.count  
 obj.save()  
 items.update(status=OrderStatus.CANCELED) # 批量更新商品状态  
 return http.JsonResponse({'msg': '取消成功'}, status=201)  
 return http.JsonResponse({'msg': '取消失败'}, status=200)  
  
 @transaction.atomic # 事务控制  
 def delete(self, request, \*args, \*\*kwargs):  
 *"""DELETE订单删除"""* # 获取订单对象  
 order\_obj = self.get\_object()  
 # 数据验证，状态判断  
 if order\_obj.status != OrderStatus.SUBMIT and order\_obj.is\_valid:  
 order\_obj.is\_valid = False  
 order\_obj.save()  
 order\_obj.order\_items.update(is\_valid=False) # 批量更新  
 return http.JsonResponse({'msg': '删除成功'}, status=201)  
 return http.JsonResponse({'msg': '删除失败'}, status=200)  
  
  
@method\_decorator(login\_required, name='dispatch') # 类装饰器 登录验证  
class OrderListView(ListView): #订单列表  
 paginate\_by = 5  
  
 def get\_queryset(self):  
 user = self.request.user  
 query = Q(is\_valid=True, user=user)  
 status = self.request.GET.get('status', 0)  
 if status:  
 query = query & Q(status=status)  
 return Order.objects.filter(query)  
  
 def get\_paginate\_by(self, queryset):  
 return self.request.GET.get('limit', self.paginate\_by)  
  
 def render\_to\_response(self, context, \*\*response\_kwargs):  
 page\_obj = context['page\_obj']  
 if page\_obj is not None:  
 data = serializers.OrderListSerializer(page\_obj).to\_dict()  
 return http.JsonResponse(data)  
 return NotFoundJsonResponse()

### forms表单

@transaction.atomic # 事务控制

F函数

from django import forms  
from django.contrib.contenttypes.models import ContentType  
from django.db import transaction  
from django.db.models import F  
  
from order.models import Order, OrderItem  
from sight.models import Ticket  
from utils import tools  
  
class SubmitTicketOrderForm(forms.ModelForm): **#订单提交表单**

ticket\_id = forms.IntegerField(label='门票ID')  
 play\_date = forms.DateField(label='出行时间')  
  
 class Meta:  
 model = Order  
 fields = ('to\_user', 'to\_phone', 'buy\_count',)  
  
 def \_\_init\_\_(self, \*args, \*\*kwargs): # 重写构造方法，添加字段保存，用作后面计算  
 super().\_\_init\_\_(\*args, \*\*kwargs)  
 self.ticket = None  
 self.buy\_count=None  
  
 def clean\_ticket\_id(self):  
 ticket\_id = self.cleaned\_data.get('ticket\_id', None)  
 ticket = Ticket.objects.select\_related('sight').filter(is\_valid=True, pk=ticket\_id).first()  
 self.ticket = ticket  
 self.buy\_count = self.cleaned\_data.get('buy\_count', None)  
 if ticket is None:  
 raise forms.ValidationError('门票信息不存在')  
 elif ticket.remain\_stock < self.buy\_count:  
 raise forms.ValidationError('剩余门票不足')  
 return ticket\_id  
  
 @transaction.atomic # 事务控制  
 def save(self, user, commit=False):  
 obj = super().save(commit=commit)  
 # 订单表 生成数据并保存  
 obj.user = user  
 obj.sn = tools.gen\_trans\_id() # 生成订单号  
 buy\_count = self.cleaned\_data.get('buy\_count', None) # 获取购买数量  
 obj.buy\_amount = self.ticket.sell\_price \* buy\_count # 计算总价格  
 obj.save()  
 # 门票表 扣减数量  
 self.ticket.remain\_stock = F('remain\_stock') - buy\_count  
 self.ticket.save()  
 # 门票明细表 关联订单明细，保存快照  
 ctype = ContentType.objects.get\_for\_model(Ticket)  
 OrderItem.objects.create(  
 user=user,  
 order=obj,  
 flash\_name=self.ticket.name, # 商品名称  
 flash\_img=self.ticket.sight.main\_img, # 商品主图  
 flash\_price=self.ticket.sell\_price, # 实付价格(原价\*折扣)  
 flash\_origin\_price=self.ticket.price, # 原价  
 flash\_discount=self.ticket.discount, # 折扣  
 count=self.buy\_count, # 购买数量  
 amount=obj.buy\_amount, # 总额  
 content\_type=ctype,  
 object\_id=self.ticket.id,  
 remark=f'出行日期：{self.cleaned\_data.get("play\_date", None)}' # 备注  
 )  
 return obj

### Serializer

from utils.serializers import BaseSerializer  
  
class OrderItemSerializer(BaseSerializer):  
 # 订单详情中的订单明细  
 def to\_dict(self):  
 obj = self.obj  
 return {  
 'pk': obj.pk,  
 'flash\_name': obj.flash\_name,  
 'flash\_img': obj.flash\_img.url,  
 'flash\_price': obj.flash\_price,  
 'flash\_origin\_price': obj.flash\_origin\_price,  
 'flash\_discount': obj.flash\_discount,  
 'count': obj.count,  
 'amount': obj.amount,  
 'remark': obj.remark,  
 'object\_id': obj.object\_id,  
 'app\_label': obj.content\_type.app\_label,  
 'model': obj.content\_type.model,  
 }  
  
class OredeDetailSerializer(BaseSerializer):  
 # 订单详情  
 def to\_dict(self):  
 obj = self.obj  
 items = []  
 for item in obj.order\_items.all():  
 items.append(OrderItemSerializer(item).to\_dict())  
 return {  
 'sn': obj.sn,  
 'buy\_amount': obj.buy\_amount,  
 'buy\_count': obj.buy\_count,  
 'types': obj.get\_types\_display(),  
 'status': obj.get\_status\_display(),  
 'created\_at': obj.created\_at,  
 'remark': obj.remark,  
 'to\_user': obj.to\_user,  
 'to\_area': obj.to\_area,  
 'to\_address': obj.to\_address,  
 'to\_phone': obj.to\_phone,  
 'express\_type': obj.express\_type,  
 'express\_no': obj.express\_no,  
 'items': items  
 }

### model

from django.contrib.contenttypes.fields import GenericForeignKey  
from django.contrib.contenttypes.models import ContentType  
from django.db import models  
  
from accounts.models import User  
from order import choices  
from utils.models import CommonModel  
  
class Order(CommonModel):  
 *"""订单"""*  
 sn = models.CharField('订单编号', max\_length=32)  
 user = models.ForeignKey(to=User, on\_delete=models.CASCADE, related\_name='order\_user', verbose\_name='关联用户')  
 buy\_count = models.PositiveIntegerField('购买数量', default=1)  
 buy\_amount = models.FloatField('总价', default=0.0)  
  
 to\_user = models.CharField('收货人', max\_length=16)  
 to\_area = models.CharField('省市区', max\_length=32, null=True, blank=True)  
 to\_address = models.CharField('详细地址', max\_length=128, null=True, blank=True)  
 to\_phone = models.CharField('手机号码', max\_length=11)  
 remark = models.CharField('备注', max\_length=128, null=True, blank=True)  
  
 # 快递信息  
 express\_type = models.CharField('快递', max\_length=32, null=True, blank=True)  
 express\_no = models.CharField('单号', max\_length=32, null=True, blank=True)  
  
 status = models.SmallIntegerField('订单状态', choices=choices.OrderStatus.choices, default=choices.OrderStatus.SUBMIT)  
 types = models.SmallIntegerField('订单类型', choices=choices.OrderTypes.choices,  
 default=choices.OrderTypes.SIGHT\_TICKET)  
  
 class Meta:  
 db\_table = 'order'  
 verbose\_name = '订单表'  
 verbose\_name\_plural = verbose\_name  
  
class OrderItem(CommonModel):  
 *""" 订单明细 """* user = models.ForeignKey(to=User, related\_name='order\_items', on\_delete=models.CASCADE, verbose\_name='关联用户')  
 order = models.ForeignKey(to=Order, on\_delete=models.CASCADE, related\_name='order\_items', null=True,  
 verbose\_name='关联订单')  
 # 商品快照  
 flash\_name = models.CharField('商品名称', max\_length=128, null=True, blank=True)  
 flash\_img = models.ImageField('商品的主图', upload\_to='avatar/%Y%m', null=True, blank=True)  
 flash\_price = models.FloatField('价格', default=0.0)  
 flash\_origin\_price = models.FloatField('原价', default=0.0)  
 flash\_discount = models.FloatField('折扣', default=0.0)  
 count = models.PositiveIntegerField('购买数量', default=1)  
 amount = models.FloatField('总额', default=0.0)  
  
 status = models.SmallIntegerField('订单状态', choices=choices.OrderStatus.choices, default=choices.OrderStatus.SUBMIT)  
 remark = models.CharField('备注', max\_length=255, null=True, blank=True)  
  
 # 复合关联  
 content\_type = models.ForeignKey(ContentType, on\_delete=models.CASCADE)  
 object\_id = models.PositiveIntegerField()  
 content\_object = GenericForeignKey('content\_type', 'object\_id')  
  
 class Meta:  
 db\_table = 'order\_item'  
 verbose\_name = '订单明细表'  
 verbose\_name\_plural = verbose\_name  
  
class Payment(CommonModel):  
 *""" 支付凭证 """* user = models.ForeignKey(to=User, on\_delete=models.CASCADE, related\_name='payments')  
 order = models.ForeignKey(to=Order, on\_delete=models.CASCADE, related\_name='payments')  
 amount = models.FloatField('金额', default=0.0, help\_text='实际支付的金额')  
 sn = models.CharField('流水号', max\_length=32, null=True, blank=True)  
 third\_sn = models.CharField('第三方订单号', max\_length=128, null=True, blank=True)  
  
 status = models.SmallIntegerField('支付状态', default=1)  
  
 meta = models.CharField('其他数据', max\_length=128, null=True, blank=True)  
 remark = models.CharField('备注信息', max\_length=128, null=True, blank=True)  
  
 class Meta:  
 db\_table = 'order\_payment'  
 verbose\_name = '支付信息表'  
 verbose\_name\_plural = verbose\_name