链滴

django 实战商城项目注册业务实现

作者：zyjImmortal
原文链接：https://ld246.com/article/1583926378334
来源网站：链滴

设计到的前端知识

项目的前端页面使用vue来实现局部刷新，通过数据的双向绑定实现与用户的交互，下面来看一下需，在用户输入内容后，前端需要做一些简单的规则校验，我们希望在在用户输入后能够实时检测，如有错误能够在输入框的下方显示出来。

```
<li>
    <label>用户名:</label>
    <input type="text" name="username" id="user_name">
    <span class="error_tip">请输入5-20个字符的用户</span>
</li>
<li>
    <label>密码:</label>
    <input type="password" name="password" id="pwd">
    <span class="error_tip">请输入8-20位的密码</span>
</li>
```

上面是一个用户和密码的输入框，当用户输入完用户名以后，光标离开输入框，能够实时的检测输入容的正确性，当输入有问题的时候，在输入框的下方显示错误信息。

v-model实现数据的双向绑定，v-on进行事件绑定，v-show是控制dom显示与否，下面是加入vue的部分代码

```
<li>
    <label>用户名:</label>
    <input type="text" name="username" id="user_name" v-model="username" @blur="chec
_username">
    <span class="error_tip" v-show="error_name">[[error_name_message]]</span>
</li>
<li>
    <label>密码:</label>
```

```
    <input type="password" name="password" id="pwd" v-model="password" @blur="check
password">
    <span class="error_tip" v-show="error_password">请输入8-20位的密码</span>
</li>
```

用户输入的用户名和username变量绑定，光标消失触发绑定时间check_username，通过v-show绑到布尔值变量error_name，来控制是否显示字符串变量error_name_message，其他的输入框都类这种操作。

注册业务实现

前端注册业务逻辑

注册表单代码：

```
<form method="post" class="register_form" >
    {{ csrf_input }}
    <ul>
        <li>
            <label>用户名:</label>
            <input type="text" name="username" id="user_name" v-model="username" @blur="
heck_username">
            <span class="error_tip" v-show="error_name">[[error_name_message]]</span>
        </li>
        <li>
            <label>密码:</label>
            <input type="password" name="password" id="pwd" v-model="password" @blur="c
eck_password">
            <span class="error_tip" v-show="error_password">请输入8-20位的密码</span>
        </li>
        <li>
            <label>确认密码:</label>
            <input type="password" v-model="password2" @blur="check_password2" name="pa
sword2"
                   id="cpwd">
            <span class="error_tip" v-show="error_password2">两次输入的密码不一致</span>
        </li>
        <li>
            <label>手机号:</label>
            <input type="text" v-model="mobile" @blur="check_mobile" name="mobile" id="ph
ne">
            <span class="error_tip" v-show="error_mobile">[[ error_mobile_message ]]</span>
        </li>

        <li>
            <label>图形验证码:</label>
            <input type="text" name="image_code" id="pic_code" class="msg_input">
            <img src="{{ static('images/pic_code.jpg') }}" alt="图形验证码" class="pic_code">
            <span class="error_tip">请填写图形验证码</span>
        </li>
        <li>
            <label>短信验证码:</label>
            <input type="text" name="sms_code" id="msg_code" class="msg_input">
```

```
            <a href="javascript:;" class="get_msg_code">获取短信验证码</a>
            <span class="error_tip">请填写短信验证码</span>
        </li>
        <li class="agreement">
            <input type="checkbox" name="allow" id="allow" v-model="allow" @change="check
allow">
            <label>同意” 商城用户使用协议 “</label>
            <span class="error_tip" v-show="error_allow">请勾选用户协议</span>
        </li>
        <li class="reg_sub">
            <input type="submit" value="注 册" @change="on_submit">
            {% if register_errmsg %}
                <span class="error_tip2">{{ register_errmsg }}</span>
            {% endif %}
        </li>
    </ul>
</form>
```

导入vue.js和ajax请求的js库

```
<script type="text/javascript" src="{{ static('js/vue-2.5.16.js') }}"></script>
<script type="text/javascript" src="{{ static('js/axios-0.18.0.min.js') }}"></script>
```

准备register.js文件

register.js文件主要处理注册页面的交互事件，并且向服务端提交注册表单请求

```
<script type="text/javascript" src="{{ static('js/register.js') }}"></script>
```

下面是实现的前端校验逻辑以及表单提交逻辑

```
methods: {
    // 校验用户名
    check_username() {
        let re = /^[a-zA-Z0-9_-]{5,20}$/;
        if (re.test(this.username)) {
            this.error_name = false;
        } else {
            this.error_name_message = '请输入5-20个字符的用户名';
            this.error_name = true;
        }
    },
    // 校验密码
    check_password() {
        let re = /^[0-9A-Za-z]{8,20}$/;
        this.error_password = !re.test(this.password);
    },
    // 校验确认密码
    check_password2() {
        if (this.password !== this.password2) {
            this.error_password2 = true;
        } else {
            this.error_password2 = false;
```

```
        }
    },
    // 校验手机号
    check_mobile() {
        let re = /^1[3-9]\d{9}$/;
        if (re.test(this.mobile)) {
            this.error_mobile = false;
        } else {
            this.error_mobile_message = '您输入的手机号格式不正确';
            this.error_mobile = true;
        }
    },
    // 校验是否勾选协议
    check_allow() {
        this.error_allow = !this.allow;
    },
    // 监听表单提交事件
    on_submit() {
        this.check_username();
        this.check_password();
        this.check_password2();
        this.check_mobile();
        this.check_allow();
                # 输入字段中有一个不符合规则就禁止
        if (this.error_name === true || this.error_password === true || this.error_password2 ===
rue
            || this.error_mobile === true || this.error_allow === true) {
            // 禁用表单的提交
            window.event.returnValue = false;
        }
    },
}
```

##后端业务注册逻辑

在用户输完用户名之后，我们往往希望能够跟快的给出这个用户名是否符合注册需求，前面只是对用名的规则进行了校验，还想知道他是否已经在系统注册过了，不然当用户都输完提交注册再给出用户或者手机号已经注册过，体验不是特别好。所以需要在光标离开用户名输入框的时候就请求服务端来断是否注册过。

定义路由

```
path('register/', views.RegisterView.as_view(), name='register'),  # name添加命名空间
path('usernames/<str:username>', views.UsernameCountView.as_view(), name="username"),
re_path(r'mobiles/(?P<mobile>1[3-9]\d{9})', views.MobileCountView.as_view(), name='mobile'
```

编写视图类

```
class UsernameCountView(View):

    def get(self, request, username):
        """
        查询该用户名是否在系统中存在
        :param request: 请求对像
```

```
        :param username: 前端传递的用户名
        :return:
        """
        count = User.objects.filter(username=username).count()
        return http.JsonResponse({'code':1001, 'msg':'用户已存在'}) if count == 1 \
            else http.JsonResponse({'code': 1000, 'msg': ''})
```

这里没有对响应做统一处理封装，后面专门介绍一下。

然后就是注册视图类的编写了：

```
class RegisterView(View):
    """用户注册视图类"""

    def get(self, request):
        '''获取注册页面'''
        return render(request, 'register.html')

    def post(self, request):
        """"""
        username = request.POST.get('username')
        password = request.POST.get('password')
        password2 = request.POST.get('password2')
        mobile = request.POST.get('mobile')
        allow = request.POST.get('allow')
        # 判断参数是否齐全
        if not all([username, password, password2, mobile, allow]):
            return http.HttpResponseForbidden('缺少必传参数')
        # 判断用户名是否是5-20个字符
        if not re.match(r'^[a-zA-Z0-9_-]{5,20}$', username):
            return http.HttpResponseForbidden('请输入5-20个字符的用户名')
        # 判断密码是否是8-20个数字
        if not re.match(r'^[0-9A-Za-z]{8,20}$', password):
            return http.HttpResponseForbidden('请输入8-20位的密码')
        # 判断两次密码是否一致
        if password != password2:
            return http.HttpResponseForbidden('两次输入的密码不一致')
        # 判断手机号是否合法
        if not re.match(r'^1[3-9]\d{9}$', mobile):
            return http.HttpResponseForbidden('请输入正确的手机号码')
        # 判断是否勾选用户协议
        if allow != 'on':
            return http.HttpResponseForbidden('请勾选用户协议')

        try:
            user = User.objects.create_user(username=username, password=password, mobile=
obile)
        except DatabaseError as e:
            return render(request, 'register.html', {'register_errmsg': e.args})

        # 注册成功保存会话
        login(request, user)

        return redirect(reverse('contents:index'))
```

django提供的login方法，封装了写入session的操作，帮助我们快速登入一个用户，并实现状态保持将通过认证的用户的唯一标识信息（比如：用户ID）写入到当前浏览器的 cookie 和服务端的 session 中。

```
request.session[SESSION_KEY] = user._meta.pk.value_to_string(user)
request.session[BACKEND_SESSION_KEY] = backend
request.session[HASH_SESSION_KEY] = session_auth_hash
```

session会存入redis，之前在工程创建时进行session存储的配置

```
SESSION_ENGINE = "django.contrib.sessions.backends.cache"
SESSION_CACHE_ALIAS = "session"
```