美多商城项目

Django认证系统

提供了用户模型类User和User的相关操作方法。

自定义User模型类之后，需要设置配置项`AUTH_USER_MODEL`。

1. 前后端域名设置

域名和IP是对应的关系。

DNS解析：获取域名对应的IP

通过域名访问网站时，先到本地的hosts中查找IP和域名的对应关系，如果查到直接根据IP访问网站，如果查不到再进行DNS域名解析。

前端服务器域名: www.meiduo.site

后端API服务器域名：api.meiduo.site

2. 短信验证码

```
URL: GET /sms_codes/(?P<mobile>1[3-9]\d{9})/
参数: url地址中传递mobile
响应:
    {
        "message": "OK"
    }
```

3. 跨域请求

浏览器的同源策略: **协议、主机IP和端口PORT相同的地址是同源，否则是非同源。**

当发起请求的页面地址和被请求的地址不是同源，那么这个请求就是**跨域请求**。

在发起请求时，如果浏览器发现请求是跨域请求，那么在请求的报文头中，会添加如下信息:

> Origin: 源请求IP地址
>
> 例如：Origin: http://www.meiduo.site:8080

在被请求的服务器返回的响应中，如果响应头中包含如下信息：

> Access-Control-Allow-Origin: 源请求IP地址
>
> 例如：Access-Control-Allow-Origin: http://www.meiduo.site:8080

那么浏览器认为被请求服务器支持来源地址对其进行跨域请求，否则认为不支持，浏览器会将请求直接驳回。

Django跨域请求扩展使用。

4. celery异步任务队列

本质:

使用进程或协程调用函数实现异步。

基本概念:

发出者：发出所有执行的任务(任务就是函数)。

(中间人)任务队列：存放所要执行的任务信息。

处理者：也就是工作的进程或协程，负责监听任务队列，发现任务便执行对应的任务函数。

特点:

1）任务发送者和处理者可以分布在不同的电脑上，通过中间人进行信息交换。

2）任务队列中的任务会进行排序，先添加的任务会被先执行。

使用:

1）安装 pip install celery

2）创建Celery对象并配置中间人地址

from celery import Celery

celery_app = Celery('demo')

配置文件：broker_url='中间人地址'

celery_app.config_from_object('配置文件路径')

3）定义任务函数

@celery_app.task(name='my_first_task')

def my_task(a, b):

print('任务函数被执行')

...

4）启动worker

celery -A 'celery_app文件路径' worker -l info

5）发出任务

my_task.delay(2, 3)

5. 用户注册

```
URL: POST /users/
参数:
    {
        "username": "用户名",
        "password": "密码",
        "password2": "重复密码",
        "mobile": "手机号",
```

```
        "sms_code": "短信验证码",
        "allow": "是否同意协议"
    }
响应:
    {
        "id": "用户id",
        "username": "用户名",
        "mobile": "手机号"
    }
```

补充(axios请求发送)

```
axios.get('url地址', [config])
    .then(response => {
        // 请求成功，可通过response.data获取响应数据
    })
    .catch(error => {
        // 请求失败，可通过error.response获取响应对象
        // error.response.data获取响应数据
    })

axios.post('url地址', [data], [config])
    .then(response => {
        // 请求成功，可通过response.data获取响应数据
    })
    .catch(error => {
        // 请求失败，可通过error.response获取响应对象
        // error.response.data获取响应数据
    })
```