# 美多商城项目

**1. 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"
    }
```

创建新用户：User.objects.create_user(*username*, *email=None*, *password=None*, ***extra_fields*)

**2. JWT token**

1）session认证

```
1. 接收用户名和密码
2. 判断用户名和密码是否正确
3. 保存用户的登录状态(session)
    session['user_id'] = 1
    session['username'] = 'smart'
    session['mobile'] = '13155667788'
4. 返回应答，登录成功
```

session认证的一些问题:

> 1. session存储在服务器端，如果登录的用户过多，服务器开销比较大。
> 2. session依赖于cookie，session的标识存在cookie中，可能会有CSRF(跨站请求伪造)。

2）jwt 认证机制(替代session认证)

```
1. 接收用户名和密码
2. 判断用户名和密码是否正确
3. 生成(签发)一个jwt token(token中保存用户的身份信息) 公安局(服务器)=>签发身份证(jwt token)
4. 返回应答，将jwt token信息返回给客户端。
```

如果之后需要进行身份认证，客户端需要将jwt token发送给服务器，由服务器验证jwt token的有效性。

3）jwt 的数据格式

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFO
```

a）headers头部

```
{
    "token类型声明",
    "加密算法"
}
```

base64加密(很容易被解密)

b）payload(载荷)：用来保存有效信息

```
{
    "user_id": 1,
    "username": "smart",
    "mobile": "13155667788",
    "exp": "有效时间"
}
```

base64加密

c）signature(签名)：防止jwt token被伪造

将headers和payload进行拼接，用.隔开，使用一个密钥(secret key)进行加密，加密之后的内容就是签名。

jwt token是由服务器生成，密钥保存在服务器端。

**3. jwt 扩展签发jwt token**

```
from rest_framework_jwt.settings import api_settings

jwt_payload_handler = api_settings.JWT_PAYLOAD_HANDLER
jwt_encode_handler = api_settings.JWT_ENCODE_HANDLER

payload = jwt_payload_handler(user)
token = jwt_encode_handler(payload)
```

**4. 用户登录**

1）jwt 扩展的登录视图obtain_jwt_token：

接收username和password，进行用户名和密码验证，正确情况下签发jwt token并返回给客户端。

2）更改obtain_jwt_token组织响应数据的函数：

```
def jwt_response_payload_handler(token, user=None, request=None):
    """
    自定义jwt认证成功返回数据
    """
    return {
        'token': token,
        'user_id': user.id,
        'username': user.username
    }

# JWT扩展配置
JWT_AUTH = {
    # ...
    'JWT_RESPONSE_PAYLOAD_HANDLER':
'users.utils.jwt_response_payload_handler',
}
```

3）登录支持用户名和手机号

```
obtain_jwt_token
-> from django.contrib.auth import authenticate
-> from django.contrib.auth.backends import ModelBackend(authenticate: 根据
用户名和密码进行校验)
```

自定义Django的认证系统后端，同时设置配置项`AUTHENTICATION_BACKENDS`。

**5. QQ登录**

1）成为QQ开发者

2）创建开发者应用