Web 项目1-day03-课堂笔记

用户模块-用户注册功能

1.1 【理解掌握】注册用户信息保存接口实现

登录状态的保存：

注册成功即登录。

```
from django.contrib.auth import login

# 只要调用 login 方法，传入 request 和 user 对象
# login 方法就会将 user 用户的信息存储到 session
login(request, user)
```

用户模块-普通登录功能

2.0【理解记忆】用户认证相关方法说明

1）新增用户的方法

User.objects.create(.....)

```
# 保存用户时会把密码先进行加密
User.objects.create_user(username=用户名, password=密码, ...)
```

2）账户名和密码校验

```
from django.contrib.auth import authenticate
# 根据账户名和密码进行校验，正确返回 User 对象，否则返回 None
# User.USERNAME_FIELD = 'username'
# User.USERNAME_FIELD = 'mobile'
user = authenticate(username=账户名, password=密码)
```

3）保存登录用户的状态(存储session)

```
from django.contrib.auth import login
# 在 session 存储用户的数据
login(request, user)
```

4）清除登录用户的状态(删除session)

```
from django.contrib.auth import logout
# 清除 session 存储的用户数据
logout(request)
```

2.1【理解掌握】用户名+密码登录接口实现

思考问题：

1）用户名+密码登录API接口设计时，需要有哪些参数?

2）用户名+密码登录API接口的业务逻辑是什么?

业务逻辑：

API接口设计：

```
============================================================
API: POST /login/
```

```
=========================================================================
请求参数：
    json
    {
      "username": "用户名",
      "password": "密码",
      "remember": "是否记住登录"
    }
参数举例：
    {
      "username": "smart",
      "password": "123456abc",
      "remember": false
    }
=========================================================================
响应数据：
    {
      "code": "响应码",
      "message": "提示信息"
    }
响应举例：
    {
      "code": 0,
      "message": "登录成功"
    }
=========================================================================
```

2.2【理解掌握】手机号+密码登录功能实现

用户登录 API 接口实现多账号登录功能：可以使用 username+password 或 mobile+password 登录。

2.3【理解掌握】用户登录之后首页显示登录用户名

在用户登录或注册之后，将登录的用户名写入到 cookie 中，前端直接从 cookie 中获取用户名进行展示。

2.4【理解掌握】登录用户退出接口实现

调用 logout 清除登录用户的 session 信息：

```
from django.contrib.auth import logout
# 内部封装了清除登录用户 session 数据的过程
logout(request)
```

用户模块-QQ 登录功能

3.1【理解掌握】QQ 登录流程分析和关键点

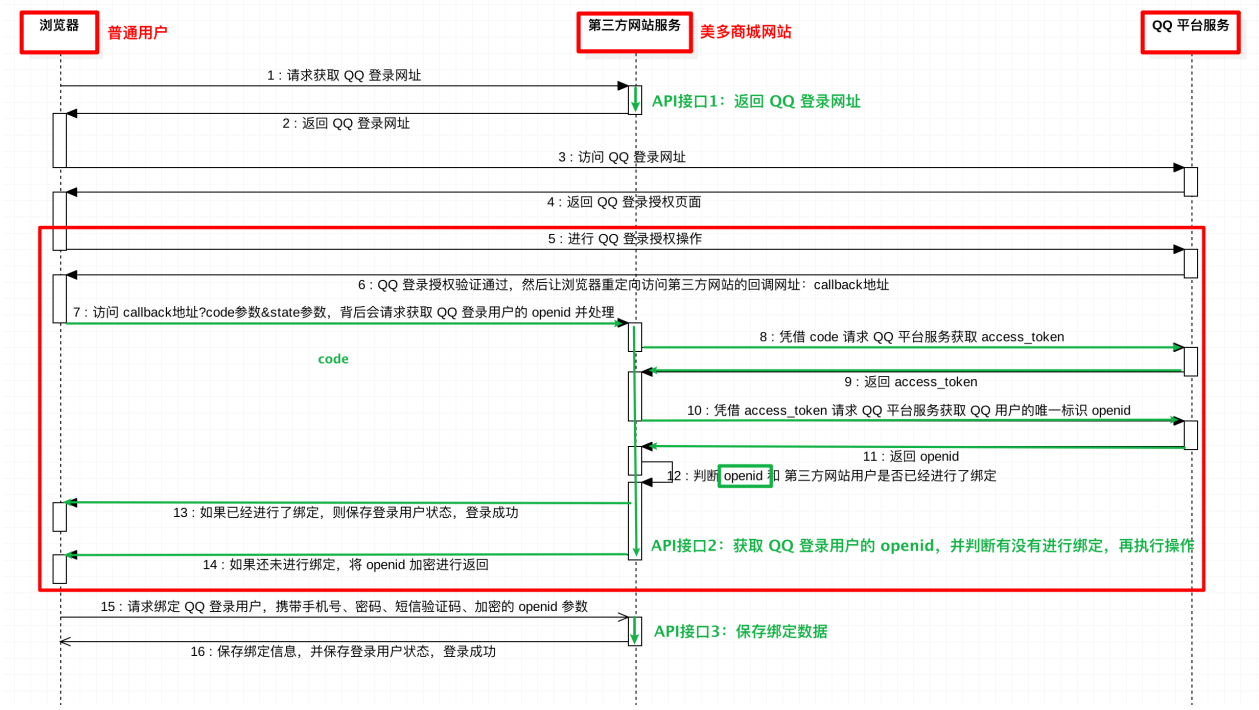
QQ 登录开发流程：

1）申请成为 QQ 互联开发者

2）创建 QQ 开发者应用

3）参考 QQ 开发文档

QQ 登录流程分析：



QQ 登录关键点：

综合上图所示，QQ 登录流程的关键点在于：获取 QQ 登录用户的 openid(`QQ用户身份的唯一标识`)，并判断该 openid 和本网站用户是否已经进行了绑定，若进行了绑定则直接让对应用户登录成功；否则需要先进行 QQ 用户的绑定操作，保存绑定数据，然后登录成功。

QQ 登录过程中，一共需要实现如下 3 个接口：

1）获取 QQ 登录网址

2）获取 QQ 登录用户的 openid 并进行操作

3）绑定 QQ 登录用户

3.2【理解掌握】QQ 登录模型类定义

绑定 QQ 用户数据保存：

进行 QQ 用户绑定操作，需要将绑定数据保存到一张数据表中，保存本网站用户的 id 和 绑定的 QQ 登录用户的 openid，数据表结构大体如下：

id	user_id	openid
1	用户id	绑定QQ登录用户openid
2	用户id	绑定QQ登录用户openid
3	...	...

```
mysql> show tables;
+---------------------------+
| Tables_in_meiduo          |
+---------------------------+
| auth_group                |
| auth_group_permissions    |
| auth_permission           |
| django_admin_log          |
| django_content_type       |
| django_migrations         |
| django_session            |
| tb_oauth_qq               |
| tb_users                  |
| tb_users_groups           |
| tb_users_user_permissions |
+---------------------------+
11 rows in set (0.00 sec)

mysql> desc tb_oauth_qq;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| id          | int(11)     | NO   | PRI | NULL    | auto_increment |
| create_time | datetime(6) | NO   |     | NULL    |                |
| update_time | datetime(6) | NO   |     | NULL    |                |
| openid      | varchar(64) | NO   | MUL | NULL    |                |
| user_id     | int(11)     | NO   | MUL | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

3.2【理解掌握】QQLoginTool 工具包的使用

```
# 安装 QQLoginTool 工具包
pip install QQLoginTool
```

基本使用：

```
# 从下载的 QQLoginTool 中导入 OAuthQQ
from QQLoginTool.QQtool import OAuthQQ

# 创建对象的时候，需要传递四个参数：
oauth = OAuthQQ(client_id='<appid>',
                client_secret='<appkey>',
```

```
                redirect_uri='<redirect_uri>',
                state='<state>')

# 调用 OAuthQQ 对象的 get_qq_url 函数，获取 QQ 登录地址
login_url = oauth.get_qq_url()

# 调用 OAuthQQ 对象的 get_access_token 函数，根据 code 获取 access_token
access_token = oauth.get_access_token(code)

# 调用 OAuthQQ 对象的 get_open_id 函数， 根据 access_token 获取 openid
openid = oauth.get_open_id(access_token)
```

3.3【理解掌握】QQ登录-step1-获取 QQ 登录网址

API接口设计：

```
==========================================================================
API: GET /qq/authorization/?next=<登录之后的访问地址>
==========================================================================
请求参数：
    通过查询字符串传递 next 参数，指定登录之后的访问地址
参数举例：
    GET /qq/authorization/?next=/
==========================================================================
响应数据：
    {
      "code": "响应码",
      "message": "提示信息",
      "login_url": "QQ登录地址"
    }
响应举例：
    {
      "code": 0,
      "message": "OK",
      "login_url": "QQ登录地址"
    }
==========================================================================
```

关键步骤：

```
# 调用 OAuthQQ 对象的 get_qq_url 函数，获取 QQ 登录地址
login_url = oauth.get_qq_url()
```

3.4【理解掌握】QQ登录-step2-获取 QQ 用户openid

API接口设计：

```
==========================================================================
```

```
API: GET /qq/oauth_callback/?code=<QQ返回的Authorization Code>
=============================================================================
请求参数:
    通过查询字符串传递 next 参数，指定登录之后的访问地址
参数举例:
    GET /qq/oauth_callback/?code=3D3A080DE796039C0937A7243B1C3410
=============================================================================
响应数据:
    1）如果 QQ 登录用户还未和本网站用户进行绑定
    {
      "code": "响应码",
      "message": "提示信息",
      "secret_openid": "加密之后的openid"
    }
    2）如果 QQ 登录用户已经和本网站用户进行绑定
    {
      "code": "响应码",
      "message": "提示信息"
    }
响应举例:
    1）如果 QQ 登录用户还未和本网站用户进行绑定
    {
      "code": "300",
      "message": "ok",
      "secret_openid": "加密之后的openid"
    }
    2）如果 QQ 登录用户已经和本网站用户进行绑定
    注：前端要求此处响应数据中的 code 必须是 0
    {
      "code": "0",
      "message": "OK"
    }
=============================================================================
```

itsdangerous 扩展包：

itsdangerous 是一个安全加密相关的 python 模块，使用 itsdangerous 可以实现数据的加解密操作，保障数据传输的安全性。

关键步骤：

```
# 调用 OAuthQQ 对象的 get_access_token 函数，根据 code 获取 access_token
access_token = oauth.get_access_token(code)

# 调用 OAuthQQ 对象的 get_open_id 函数， 根据 access_token 获取 openid
openid = oauth.get_open_id(access_token)
```

3.5【理解掌握】QQ登录-step3-绑定 QQ 登录用户数据保存

API 接口设计：

```
============================================================
API: POST /qq/oauth_callback/
============================================================
请求参数:
    json
    {
      "mobile": "手机号",
      "password": "密码",
      "sms_code": "短信验证码",
      "secret_openid": "加密的openid",
    }
参数举例:
    {
      "mobile": "1356677888",
      "password": "123456abc",
      "sms_code": f235341,
      "secret_openid": "xxxxxxxx"
    }
============================================================
响应数据:
    {
      "code": "响应码",
      "message": "提示信息"
    }
响应举例:
    {
      "code": 0,
      "message": "登录成功"
    }
============================================================
```