各参数示例值

|  |  |
| --- | --- |
| 参数 | 示例值 |
| app\_code | hr78hif9q84t94t9 |
| scope | base\_info |
| user\_id | 100001 |
| authorize\_code | d9a8b72e60d2b1114f65c30dedb843ee |
| access\_token | 9ba6b21d9141b83e4a87c292f8b63500 |
| expire\_time | 300 |
| timestamp | 1561360494 |

## ① 记录授权码（Authorize Code）到缓存中

（使用redis-string格式记录）

格式：

key的格式为：auth\_{authorize code}

如：auth\_d9cadfe8e54b092ad9c2b3bbb1026d18

value值为为 {app\_code}|{scope}

各个数据之间用竖线“|”分隔

若是请求获取用户授权，则加上“|{u}|{user\_id}”

例：

（获取用户基本信息）hr78hif9q84t94t9|base\_info|u|100001

（获取热图列表）hr78hif9q84t94t9|hot\_photo\_pictures

过期时间设置为5分钟（300秒）

## ② 使用Authorize Code获取 Access\_token时

验证Authorize Code是否有效

从redis中获取数据验证准确性

Key格式为：auth\_{authorize\_code}，若not exists，则表示Authorize Code无效

Value值格式为：{app\_code}|{scope}或{app\_code}|{scope}|{u}|{user\_id}

验证第一个值是否与请求header中的app\_code值一致

根据Access Token生成规则生成Access Token，检查缓存中（Redis）是否已存在对应Access Token

Key：token\_{access\_token}

## ③ 将access\_token记录到redis中(记录令牌到缓存)

（使用redis-string格式记录）

格式：

key的格式为：token\_{access\_token}

例：token\_c7672b9a9fb45bf058475ebae16d7a73

value值格式：{app\_code}|{scope} 或是 {app\_code}|{scope}|{u}|{user\_id}

例：

（获取用户基本信息）hr78hif9q84t94t9|base\_info|u|100001

（获取热图列表）hr78hif9q84t94t9|hot\_photo\_pictures

过期时间设置为30天（2592000秒）

## ④ 使用Access Token请求开放接口，验证Access Token

验证Access Token是否有效

从redis中获取数据验证准确性

Key格式为：token\_{access\_token}，若not exists，则表示Access Token无效，需要重新获取授权

Value值格式为：{app\_code}|{scope}或{app\_code}|{scope}|{u}|{user\_id}

验证第一个值是否与请求header中的app\_code值一致

验证请求参数中的scope是否存在缓存中的scope中（缓存中的scope可能为数组，以英文逗号分隔“,”）

## ⑤ 验证近期是否已授权（防止重复请求授权）

当数据验证通过时，判断请求授权的类型是会员授权还是服务端授权。

1. 会员授权——跳转用户登录

用户登录成功，获取用户标识id，生成对应Authorize Code字段生成规则生成Authorize Code，验证缓存中是否已存在同样Authorize Code

若存在，页面重定向到“您近期已授权”提示页面（或者1-不提示，直接返回；2-同样提示授权成功）

若不存在，记录key：auth\_{authorize\_code},value={app\_code}|{scope}|{u}|{user\_id}到缓存中，设定过期时间

1. 服务端授权——跳过用户登录步骤

使用key字段生成规则（key=auth\_{authorize\_code}）生成key，检查是否存在对应key-value

（验证近期是否已授权，跳过重复授权，重复授权会重置授权码的过期时间）

**接口请求接收跟返回的model规范**

Model头部使用 [DataContract] 约束序列化字段

内的字段使用 [DataMember( Name = ”XXX” )] 定义接收的字段名和返回序列化时的字段名

（这样的好处在于，接口在处理结束后，将数据return时，若有DataContract约束在，只序列化带有DataMember特性的字段。另外一点，若后续要修改接收和响应的参数名时，只需要更改Name部分的参数即可）

在接口的Filter处，考虑使用HttpResponseMessage来做错误响应数据格式

HttpResponseMessage对象可以控制响应的header等信息，如：找不到对应的数据，可直接使用 NotFound()，其状态码（Http Status Code）为404

## Authorize Code & Access Token 生成规则

### ⑥ Authorize Code生成规则

生成authorize code值

生成规则

以 {app\_code}|{权限}|{对象类型标识}|{标识id} 格式拼接后使用MD5加密获得32位字符串，即为Authorize Code值

例：MD5(“hr78hif9q84t94t9|base\_info |u|100001”)

得到：d9a8b72e60d2b1114f65c30dedb843ee 即为Authorize Code 值

### ⑦ Access Token生成规则

生成规则

以 {app\_code}|{权限}|{对象类型标识}|{标识id}|{authorize\_code} 格式拼接

使用MD5加密获得32位字符串，即为Access Token值

例：MD5(“hr78hif9q84t94t9|base\_info |u|100001| d9a8b72e60d2b1114f65c30dedb843ee”)

得到：9ba6b21d9141b83e4a87c292f8b63500 即为 Access Token值

## ⑧ 第三方平台每次请求接口时做请求记录

接口每次请求先经过filter做数据校验，通过校验后将请求数据记录到redis中，并设置过期时间，同时添加在数据库的请求记录日志表

Redis记录格式：redis-string

key字段格式：{app\_code}\_{timestamp}，并设置有效期

如：hr78hif9q84t94t9\_1561360494

sign\_data字段数据记录在cache中，key为app\_code值，value值任意，并设置有效期

防止第三方在短时间内进行重复数据的请求