IMG_256

**数智汇-用户端**

**开发者指南**

— 北京希嘉创智教育科技有限公司©版权所有 —

目录

[1获取API地址 3](#_Toc17904864)

[1.1查看API的URL地址 3](#_Toc17904865)

[1.2查看App Key 3](#_Toc17904866)

[2.查询操作 4](#_Toc17904867)

[2.1全量数据查询 4](#_Toc17904868)

[2.2分页 6](#_Toc17904869)

[2.3排序 7](#_Toc17904870)

[2.4条件查询 8](#_Toc17904871)

[2.4.1 简单条件查询 8](#_Toc17904872)

[2.4.2 模糊查询 9](#_Toc17904873)

[2.5.数据类型 12](#_Toc17904874)

[2.6加密与解密 14](#_Toc17904875)

[3.新增操作 15](#_Toc17904876)

[4.更新操作 15](#_Toc17904877)

[5关于key值大小写 16](#_Toc17904878)

[2. 授权方式说明 16](#_Toc17904879)

[3. 调用频率设置说明 17](#_Toc17904880)

[4. 错误码说明 18](#_Toc17904881)

[5.调用示例 19](#_Toc17904882)

[5.1puby调用示例 19](#_Toc17904883)

[5.2JAVA调用示例 20](#_Toc17904884)

[5.3C#调用示例 21](#_Toc17904885)

[5.4php调用示例 23](#_Toc17904886)

[5.5python调用示例 24](#_Toc17904887)

# 1调用类型说明

## 1.1获取API地址

### 1.1.1查看API的URL地址

开发者为应用申请API成功后，可在数据中心/数据管理/接口数据管理中查看API的URL地址。如图1。

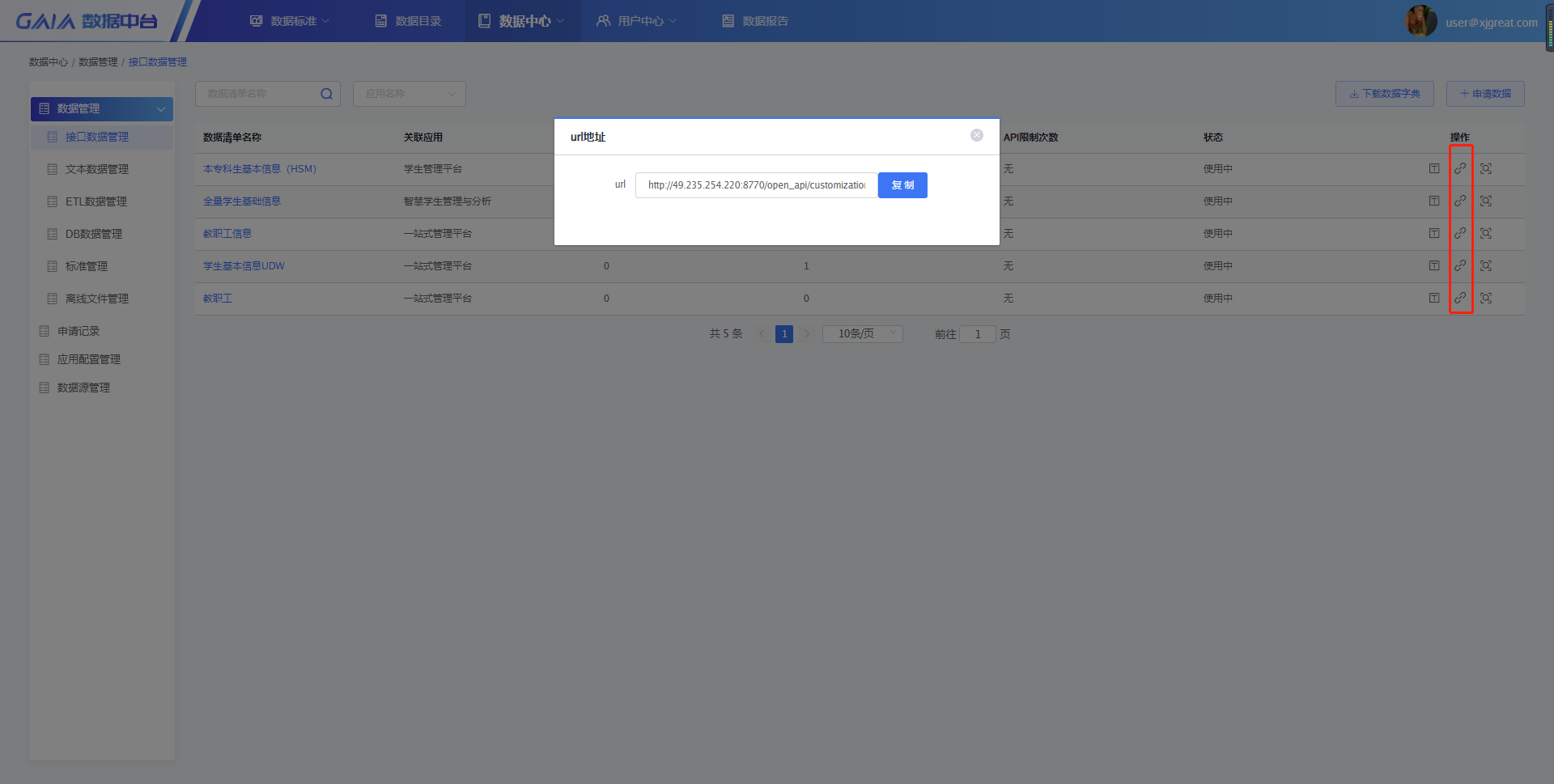
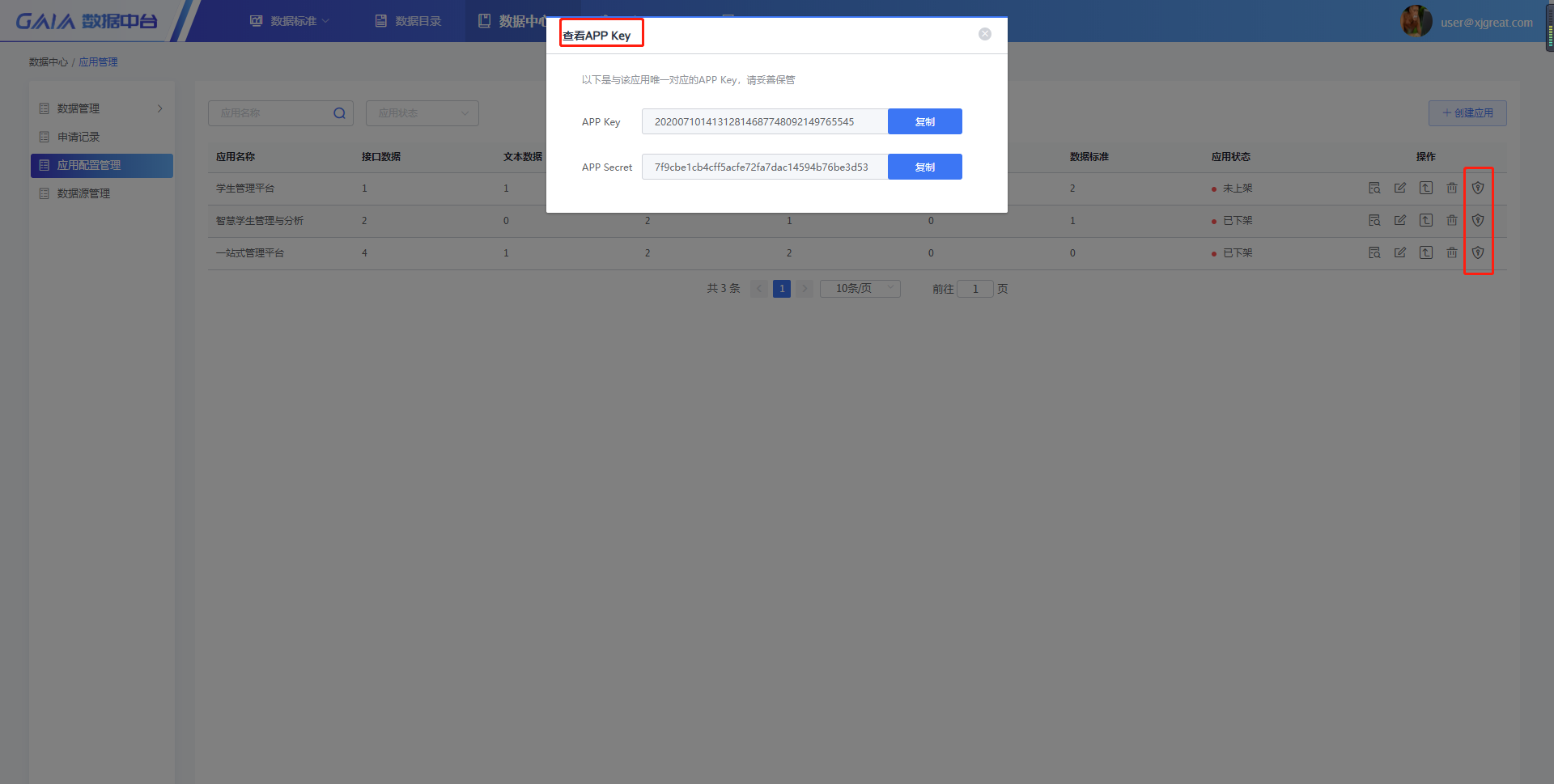


图1

### 1.1.2查看App Key

开发者登录后，在数据中心/应用配置管理页面点击“查看授权码”按钮，可查看应用的App key、App secret。如下图查询到App key、App secret。



## 1.2查询操作

### 1.2.1全量数据查询

开发者通过API接口获取全量数据

1. 获取请求token

开发者可根据应用的App key、App secret生成Token，如：

http://127.0.0.1:80/open\_api/authentication/get\_access\_token?key=\*\*\*&secret=\*\*\*

key、secret可在应用管理->App Key页面进行查看，如上图所示。

2.传参示例（当参数仅仅是键值对形式时，可以直接构建URL访问，非键值对形式的参数建议构建json请求体之后通过POST访问。每个API都支持GET,POST两种传参模式）：http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list?access\_token=xxxxxxxxxxxxxxxxxxxx

1. 返回结果格式如下图说明：



注意：API默认返回格式为JSON,除此之外，还支持XML,TEXT两种，如果要修改返回格式，只需修改请求地址URL信息,如下所示：

JSON格式：

http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list?access\_token=xxxxxxxxxxxxxxxxxxxx

XML格式：

http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list.xml?access\_token=xxxxxxxxxxxxxxxxxxxx

TEXT格式：

http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list.text?access\_token=xxxxxxxxxxxxxxxxxxxx

### 1.2.2分页

开发者通过API接口获取数据时，数据会以分页的形式传输。分页通过page与per\_page参数生效，page表示第几页，per\_page表示当前的页面的数据条数。page默认值为1，per\_page默认值为10。

开发者如想要查询第二页的数据，每页100条，即第201条至300条数据。

示例（URL传参）：http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list?access\_token=xxxxxxxxxxxxxxxxxxxx&page=2&per\_page=100

示例（请求体传参）：URL:http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list

请求体body(json):{

        "access\_token":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",

        "page": 2,

        "per\_page":10

}

注意：

1.键值对形式的参数既可写在URL的后面， 也可写在请求体中。注意不要在URL和请求体中出现重复的参数名的情况。

2.带有请求体的请求无法直接在浏览器地址栏中访问，如需测试，请使用postman等工具，或浏览器安装可以发送post请求的插件。如图2以postman为例，在拥有网络环境的情况下，可访问postman官网下载。<https://www.getpostman.com/>

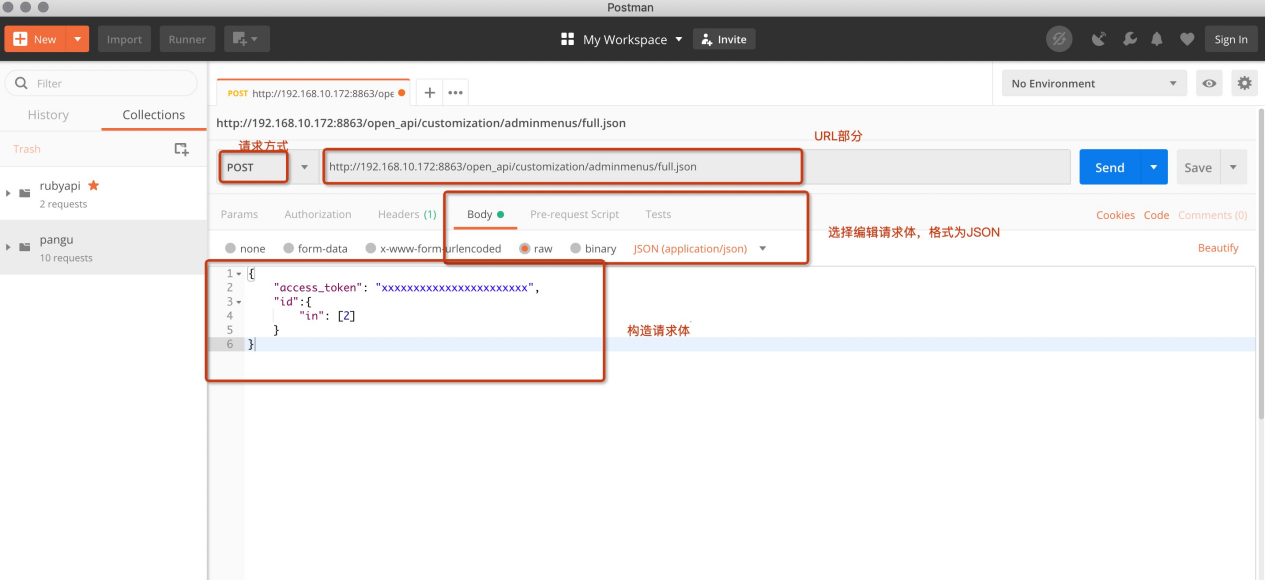


图2

注意：oracle数据库采用的是堆排序，所以每次取出的数据是无序的。在使用oracle数据库的分页时，请务必带上一个或多个排序条件（排序条件尽量唯一或重复少），具体使用方式见2.3（排序）

### 1.2.3排序

实现排序需要使用参数order，而且由于排序参数较为复杂只能使用请求体构建

示例（请求体传参）：

http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list

请求体body(json):

{

        "access\_token":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",

        “order”:{

               "id": "asc"

       }

}

order是声明排序的关键字，id是想要用于排序的字段，asc为排序方式，asc为升序，desc为降序。

相当于SQL语句中的"order by id asc"。

另外，支持多条件同时排序。

{

        "access\_token":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",

        “order”:{

               "group\_id": "asc",

                "created\_at":"desc"

       }

}

先根据group\_id排序，再根据created\_at字段排序。

相当于SQL语句中的“ORDER BY group\_id ASC, create\_at DESC”。

### 1.2.4条件查询

API接口中作为查询条件传递的字段，必须是“可请求字段”（即管理员对开发者提交的API进行审核时，需要勾选“请求字段”），如果该字段不是可请求字段，即便作为查询条件进行传参，也会被过滤掉无法生效。

注：1.如果某个字段是必填字段，即必须传递此字段，否则无法获取API数据（在请求api后会返回相应的错误信息提示） ；

2.如条件查询的值中有空格，需要将空格转义为%20后才可进行查询。

#### 1.2.4.1 简单条件查询

简单条件查询，即相等查询，以键值对的方式传值，在URL或请求体中携带参数皆可。

示例（URL传参）：http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list?access\_token=xxxxxxxxxxxxxxxxxxxx&name=张三

示例（请求体传参）：http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list

{

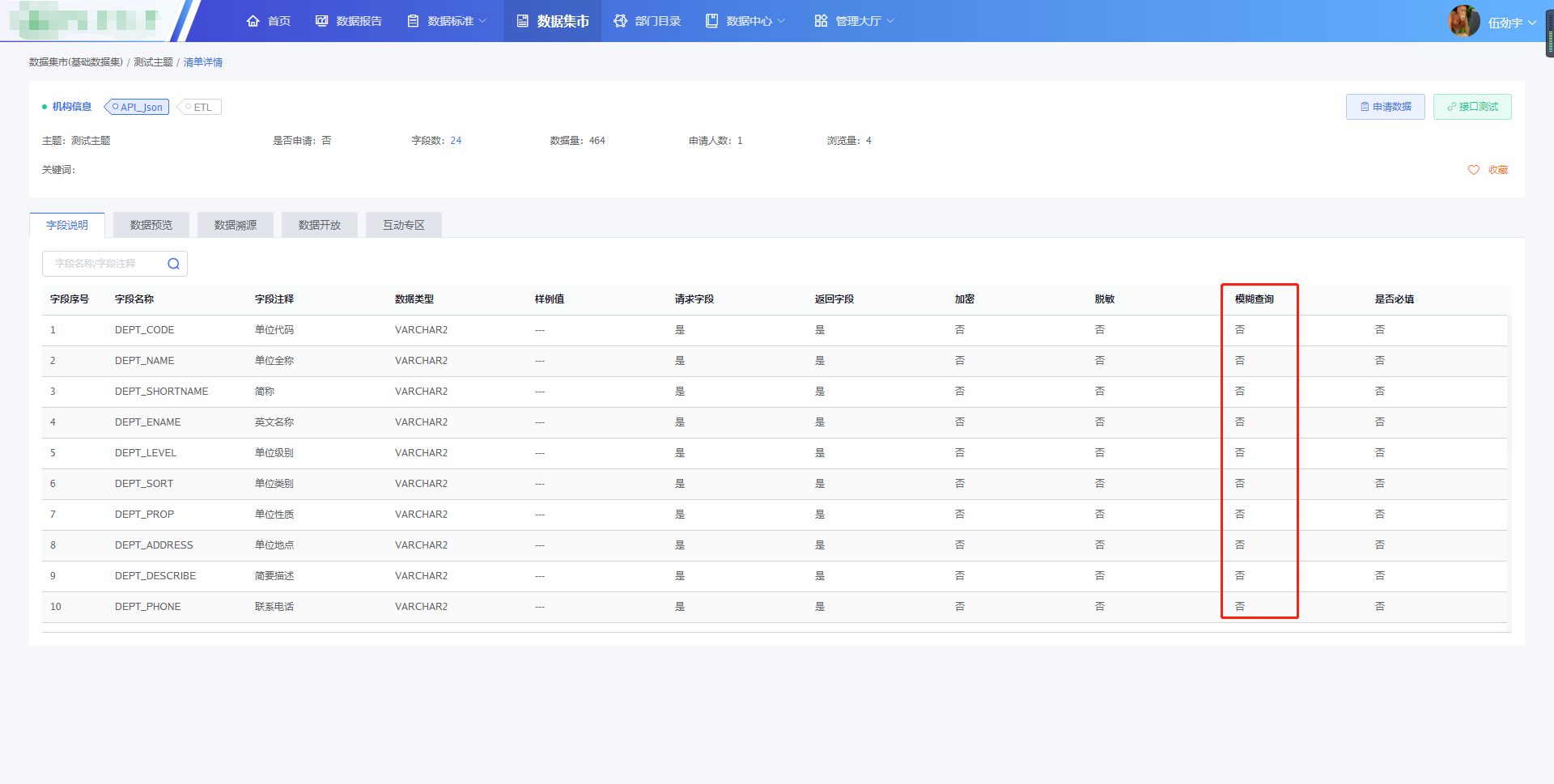
        "access\_token":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",

        “name”:"张三"

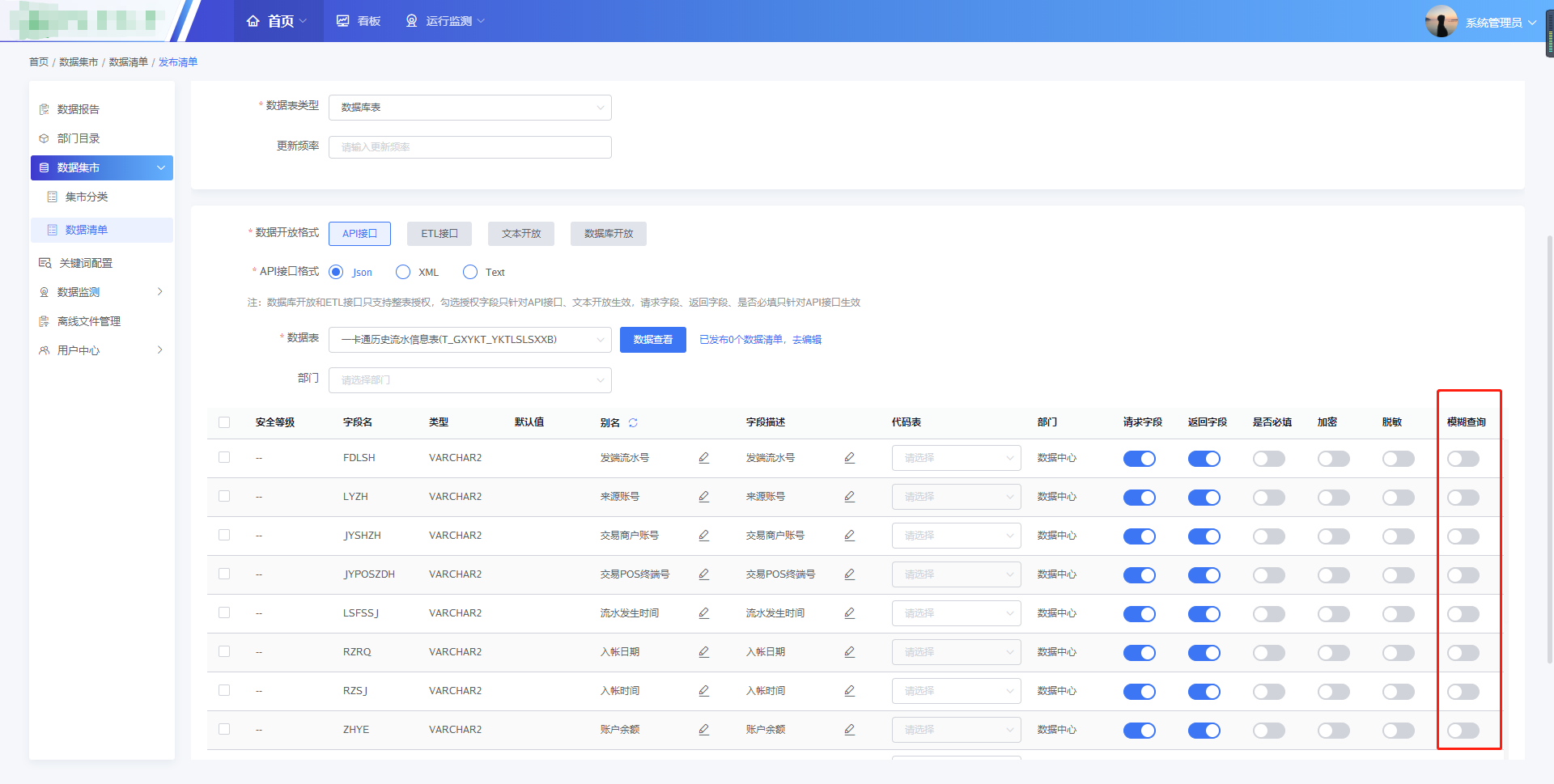
}

相当于SQL语句中的"WHERE name='张三'" 。

#### 1.2.4.2 模糊查询

模糊查询即数据库的使用like关键字进行模糊查询。开发者在数据集市—API详情—请求参数说明中，可以查看到哪些字段是支持模糊查询的字段，如图： 

注意：要对某些字段做模糊查询，管理员必须在发布API的时候勾选支持模糊查询的权限，如图所示管理端API发布页面，如开发者需要哪个字段为模糊查询可联系管理员勾选权限。



如：查询姓名的第一个字为张，学号中包含0714，身份证号码以X结尾的数据。

示例（请求体传参）：

http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list

{

"access\_token":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",

"name":"%%张%%%",

"num":"%%%0714%%%" ,

"id\_card":"%%%X%%"

}

值分为两部分，第一部分为前后各两个%号，用以标注此字段使用模糊查询，第二部分为中间，"张%"表示匹配"张"开头的数据，"%0714%"表示匹配任意位置包含"0714"的数据，"%X"表示匹配以"X"结尾的数据。

相当于SQL语句中的"WHERE name LIKE '张%' AND num LIKE '%0714%' AND id\_card LIKE '%X'"。

另，如果将参数通过URL携带，需要将"%"转码为"%25"，如：查询姓名的第一个字为张的数据。

示例（URL传参）：http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list?access\_token=xxxxxxxxxxxxxxxxxxxx&name=%25%25张%25%25%25

#### 1.2.4.3 比较查询

如果想要对某个字段进行大于（>,>=）、小于(<,<=)、包含（in）查询，需要使用比较查询，由于比较查询参数较为复杂因此只能使用请求体构建。

如:查询5<id<20 且 18<=age<=22 且 group\_id位于1，2，3，4中的其中之一的数据。

示例（请求体传参）：

http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list

{

        "access\_token":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",

        "id":{

                  "gt": 5,

                  "lt": 20

        },

        "age":{

                "gte":18,

                "lte": 22

        },

        "group\_id":{

                "in": [1,2,3,4]

        }

}

其中gt表示大于（>）,lt表示小于（<）,gte表示大于等于（>=）,lte表示小于等于(<=)，in表示包含（in）。

相当于"WHERE (id > 5 AND id < 20) AND (age >= 18 AND age <= 22) AND (group\_id IN (1,2,3,4))"。

#### 1.2.4.4 复杂逻辑查询

如果需要实现复杂的查询条件，需要使用到"and"和"or"关键字，以and和or作为键，其值为一个列表，列表中间的条件会被关键字"and"或"or"连接，由于复杂逻辑查询参数较为复杂因此只能使用请求体构建。

如：查询 name="张三" 或 name="李四" 的数据。

示例（请求体传参）：

http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list

{

        "access\_token":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",

        "or":[

                {"name":"张三"},

                {"name":"李四"}

        ]

}

列表中的{}中可填写的内容与最外层的{}一致，中间可以继续使用所有的条件查询，包括"and"和"or"。

相当于"WHERE (name = '张三' or name='李四')"。

### 1.2.5.数据类型

#### 1.2.5.1 字符串类型

如果字段是字符串类型，使用请求体传递时请使用双引号，避免数据库报错。

接口报错时请首先检查传递字段的类型。

#### 1.2.5.2 数字类型

如果是数字类型，在进行相等查询时，引号不影响比较。如果是比较查询时，且值的长度不统一时，请不要使用引号，会被当做字符串处理，而字符串的比较时从第一位开始比较，与数字类型的比较可能出现结果不同。

如：查询id大于30的值。

示例（请求体传参）：

http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list

{

        "access\_token":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",

        "id":{

                  "gt": 30

        }

}

相当于sql语句中的"WHERE id > 30"。

注：如果30被打上引号，相当于"WHERE id > '30'",可能造成查询数据的不准确。

#### 1.2.5.3 时间类型

如果是时间类型，除oracle数据库外，其他数据库的时间类型当做普通字符串类型处理即可。

如果数据源是oracle数据库,由于oracle数据库时间类型的特殊性，查询或比较时需要用to\_date函数声明时间格式。

本api默认采用'yyyy-mm-dd hh24:mi:ss’格式。

如：查询create\_time晚于2018-01-01 00：00：00 且早于 2019-01-01 00：00：00的数据。

示例（请求体传参）：

http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list

{

        "access\_token":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",

        "create\_time":{

                  "gt": "2018-01-01 00：00：00",

                  "lt": "2019-01-01 00：00：00"

        },

}

相当于SQL语句 ：WHERE create\_time > to\_date('2018-01-01 00：00：00', 'yyyy-mm-dd hh24:mi:ss') AND create\_time < to\_date('2019-01-01 00：00：00', 'yyyy-mm-dd hh24:mi:ss')。

如果所要查询的时间字段格式并不是'yyyy-mm-dd hh24:mi:ss’格式，可以使用to\_date声明格式。

如：查询create\_time晚于20181231 且早于 20191231的数据。

示例（请求体传参）：

http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list

{

        "access\_token":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",

        "create\_time":{

                  "gt": "to\_date('20181231', 'yyyymmdd')",

                  "lt": "to\_date('20191231', 'yyyymmdd')"

        },

}

相当于SQL语句 ：WHERE create\_time > to\_date('20181231', 'yyyymmdd') AND create\_time < to\_date('20191231', 'yyyymmdd')

### 1.2.6加密与解密

加密与解密分为两种，第一种为管理员设置的数据加密，第二种为二进制流字段加密。

#### 1.2.6.1 返回密文

如果管理员设置的此API返回密文，此密文是为了防止开发者看到明文，但此密文可以在调用查询、新增、更新时作为值直接使用，并不影响开发者调用数据。

如： 数据库中某条值的CARD\_ID明文为1234567，

前一次请求的api返回了密文CARD\_ID: "m76JwhWgzNnz1HmxZknaUg=="

可直接将此值作为条件查询。

示例（URL传参）：

http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list?access\_token=xxxxxxxxxxxxxxxxxxxx&card\_id=m76JwhWgzNnz1HmxZknaUg==

示例（请求体传参）：

http://127.0.0.1:80/open\_api/customization/testzp\_alpha/list

{

        "access\_token":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",

        "CARD\_ID": "m76JwhWgzNnz1HmxZknaUg=="

}

即可查到CARD\_ID为1234567的相关值。

#### 1.2.6.2 二进制流字段的加密

由于二进制流字段（BLOB，LONGBLOB，binary,var binary,image）无法通过Json传输，所以上述字段的值都会采用AES加密后返回给用户。

salt: 16位。生成方式：用本app的key。如果本app的key长度不足16位，用1补齐，如果超过16位，截断只取16位。

密钥：32位。生成方式：用本app的secret。如果本app的secret长度不足32位，用1补齐，如果超过32位，截断只取32位。

key与secret在应用基本信息，可在数据中心/应用配置管理页面点击“查看授权码”按钮查看。可见1.2查看APP key部分。

二进制流字段示例代码可在前台/数据中心/文档中心/调用类型说明，页面最下方下载二进制流解密示例代码。（下载二进制流解密示例代码）

## 1.3.新增操作

新增的路由是insert ，新增的格式与相等查询大致相同。

示例（URL传参）：

http://121.40.193.25:8863/open\_api/customization/testzp\_alpha/insert?access\_token=xxxxxxxxxxxxxxxxxxxx&name=张三&id=10

示例（请求体传参）：

http://127.0.0.1:80/open\_api/customization/testzp\_alpha/insert

{

        "access\_token":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",

       "id": 10,

        "name": "张三"

}

相当于SQL语句“INSERT INTO table\_name（id，name）VALUES(10, '张三')”

注：为了兼容旧版本，insert也可替换为create。即请求http://121.40.193.25:8863/open\_api/customization/testzp\_alpha/create?access\_token=xxxxxxxxxxxxxxxxxxxx&name=张三&id=10  是同样的效果

## 1.4.更新操作

更新的路由是update，更新较为特殊，分为两个部分，一是需要传递所要更新的字段及其值，二是必须要筛选条件。更新只能通过post构建请求体的方式访问，无法通过浏览器以一条url直接访问。

示例（请求体传参）：http://127.0.0.1:80/open\_api/customization/testzp\_alpha/update

{

        "access\_token":"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",

        "name": "张三"，

        "and":[

                {"id": 10}

        ]

}

第一部分指的就是"name": "张三"，表示的是将name字段的值修改为张三。

第二部分指的是"and"中的内容，是更新的筛选条件， 整个请求表达的意思是，将id=10的那条数据的name字段的值改为张三。相当于SQL语句“UPDATE table\_name SET name = '张三' where id =10；”

注：一定要用and或者or加上限制条件，不然会被拦截，无法修改。

## 1.5关于key值大小写

oracle数据库，如果建表时字段名是大写，key值大小写皆可，如果创建表时字段名是小写，需要给字段名加上转义的双引号。"\"name2\"" 。

其他关系型数据库暂大小写均不严格。

# 2. 授权方式说明

数智汇的api授权token生成方式如下：

(1) 将应用管理App Key页面中的appkey, appsecret作为参数传入, 调用认证接口(/open\_api/authentication/get\_access\_token)获取access\_token。

(2) 认证接口请求URL形如http://192.168.10.78:8863/open\_api/authentication/get\_access\_token?key=\*\*\*&secret=\*\*\*，实际调用方法因开发语言不同会有差异，请根据实际情况修改调用逻辑。

(3) 认证接口返回对象包含：code(返回代码)，message(返回说明)，description(返回说明中文)，result(结果对象)。其中result中包含：access\_token(API调用凭证)，expires\_in(token有效期，单位为秒)。示例如下：

{

'code': 10000,  
'message': 'ok',  
'description': 'api请求成功',  
'result': {

'access\_token': 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxx',  
'expires\_in': 7200

}

}

(4) 建议在调用API时将access\_token在本地做缓存，根据首次请求获取的expires\_in进行检查，若token未过期，则可以继续使用，否则重新调用认证接口获取新的token即可，这样可以减少调用API期间的服务器请求压力。

(5) 若不考虑将access\_token做缓存，可以使用refresh\_access\_token(刷新调用凭证)接口(/open\_api/authentication/refresh\_access\_token)，该接口每次都会返回新的token(返回结果格式同3)中结果对象示例)，从而避免了token过期的问题，但相应的需要每次调用API前调用该接口获取token，会增加服务器请求压力。

(6) 请求API接口时，除API文档中支持的标准参数外，需同时传入access\_token参数作为调用唯一凭证。

# 3. 调用频率设置说明

数智汇开放接口限制每段时间只能请求一定的次数。限制的单位时间有每小时、每天、每月；限制的维度为单个授权应用。例如：

• 单个应用每小时内只能请求数智汇开放接口n次；

• 单个应用每天累计只能请求数智汇开放接口m次；

• 单个应用每月累计只能请求数智汇开放接口x次；

其中n、m、x的具体数值，数智汇将采用应用质量评估系统，实现智能评估应用质量，质量高的应用相应的这些数值就高，也就是请求限制小。因此限制值不固定，不同的应用有不同的限制，取决于应用自身的质量。

# 4. 错误码说明

|  |  |  |
| --- | --- | --- |
| **错误代码** | **错误信息** | **详细描述** |
| 10001 | System error | 系统错误 |
| 10002 | Service unavailable | 服务暂停 |
| 10003 | Remote service error | 远程服务错误 |
| 10004 | IP limit | IP限制不能请求该资源 |
| 10005 | Permission denied, need a high level appkey | 该资源需要appkey拥有授权 |
| 10006 | required paramters are missing | 必填参数缺失 |
| 10007 | Unsupport mediatype | 不支持mediatype类型 |
| 10008 | Param error, see doc for more info | 参数错误,请参考API文档 |
| 10009 | Too many pending tasks, system is busy | 任务过多,系统繁忙 |
| 10010 | Job expired | 任务超时 |
| 10011 | Illegal request | 非法请求 |
| 10012 | Insufficient app permissions | 应用的接口访问权限受限 |
| 10013 | Miss required parameter , see doc for more info | 缺失必填参数，请参考API文档 |
| 10014 | Parameter type invalid , see doc for more info | 参数类型无效,请参考API文档 |
| 10015 | Request body length over limit | 请求长度超过限制 |
| 10016 | Api requests out of rate limit | Api请求频率超过上限 |
| 10017 | Api not found | Api 不存在 |
| 10018 | Api invalid or expired | Api授权无效或过期 |
| 10019 | Api params error | Api 参数错误 |
| 10020 | No permission to access the API | 没有权限访问当前API |
| 10021 | Application not found | 应用不存在 |
| 20001 | parameter is null | 参数为空 |
| 20002 | Content is illegal | 包含非法内容 |
| 20003 | Image size too large | 图片太大 |
| 20004 | Parameter is too many | 参数超长 |
| 20005 | Unsupported image type, only suport JPG, GIF, PNG | 不支持的图片类型，仅仅支持JPG、GIF、PNG |
| 20006 | Domain not exist | 域名不存在 |
| 20007 | Expired token | token过期 |
| 20008 | Parameter format error | 参数格式错误 |
| 20009 | Token error | token 错误 |
| 20010 | Token is invalid | 无效的token |
| 20011 | Contains forbid world | 含有敏感词 |
| 20012 | Param decrypt error | 加密字段解析错误 |
| 20013 | Records not found | 查询结果为空 |
| 20014 | Redis is not connected | Redis服务器无法连接 |
| 20015 | Elastic search is not connected | ES服务器无法连接 |
| 20016 | Elastic search index is not existed | ES索引不存在 |

# 5.调用示例

## 5.1puby调用示例

支持在文档中心-调用示例-puby调用示例页面下载puby调用视频。

#首先需要获取token，是根据key和secret生成的,其中127.0.0.1:3033根据实际部署环境api地址IP替换

token\_url = "http://127.0.0.1:3033/open\_api/authentication/get\_access\_token"

#key,secret在开发者中心列表中应用查看页面中获取

token\_params[:key] = '123456'

token\_params[:secret] = 'abcdef'

token\_body = RestClient.post(token\_url,

token\_params.to\_json, :content\_type => :json)

res\_hash = JSON.parse(token\_body)

if res\_hash[:message] != 'ok' #如果请求结果不是ok

  # error 处理

else

  access\_token =

res\_hash['result']['access\_token']

end

#以学生基础信息列表查询API为例

request\_url = "http://127.0.0.1:3033/open\_api/student/infos/list"

request\_params[:access\_token] = access\_token

body = RestClient.post(request\_url, request\_params.to\_json, :content\_type

=> :json)

body = JSON.parse(body)

## 5.2JAVA调用示例

支持在文档中心-调用示例-JAVA调用示例页面下载JAVA调用视频。

public static void postMain(String[] args) {

try {

#首先需要获取token，是根据key和secret生成的,其中127.0.0.1:3033根据实际部署环境api地址IP替换

URL targetUrl = new URL("http://127.0.0.1:3033/open\_api/authentication/get\_access\_token");

#创建一个POST请求，调用上面的API URL信息

HttpURLConnection httpConnection = (HttpURLConnection) targetUrl.openConnection();

httpConnection.setDoOutput(true);

httpConnection.setRequestMethod("POST");

httpConnection.setRequestProperty("Content-Type", "application/json");

#构建一个json对象，key,secret的值在应用查看中获取

String input = "{\"key\":111,\"secret\":\"222\"}";

#将请求返回的结果写入流中

OutputStream outputStream = httpConnection.getOutputStream();

outputStream.write(input.getBytes());

outputStream.flush();

if (httpConnection.getResponseCode() != 200) {

throw new RuntimeException("Failed : HTTP error code : " +

httpConnection.getResponseCode());

}

#读取流信息，答应出返回的结果

BufferedReader responseBuffer = new BufferedReader(new InputStreamReader(

(httpConnection.getInputStream())));

String output;

System.out.println("Output from Server:\n");

while ((output = responseBuffer.readLine()) != null) {

System.out.println(output);

}

#断开HTTP连接

httpConnection.disconnect();

} catch (MalformedURLException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

## 5.3C#调用示例

public void requestApiForPostMethod() {

#首先需要获取token，是根据key和secret生成的,其中127.0.0.1:3033根据实际部署环境api地址IP替换

string apiAddress = "http://127.0.0.1:3033/open\_api/authentication/get\_access\_token";

HttpWebRequest request = (HttpWebRequest)

WebRequest.Create(apiAddress);

request.Method = "POST";

request.ContentType = "application/json";

//创建要发送的数据

string key = "111"

string secret = "222"

StringBuilder data = new StringBuilder();

data.Append("key=" + HttpUtility.UrlEncode(key));

data.Append("secret=" + HttpUtility.UrlEncode(secret));

//创建要发送的数据的字节数组

byte[] byteData = UTF8Encoding.UTF8.GetBytes(data.ToString());

//设置请求头的长度

request.ContentLength = byteData.Length;

//将数据写入流

using(Stream postStream = request.GetRequestStream())

{

postStream.Write(byteData, 0, byteData.Length);

}

//获取响应

using(HttpWebResponse response = request.GetResponse() as

HttpWebResponse) {

// 获取响应的流信息

StreamReader reader = new StreamReader(response.GetResponseStream());

//输出

Response.Write(reader.ReadToEnd());

}

}

## 5.4php调用示例

#首先需要获取token，是根据key和secret生成的,其中127.0.0.1:3033根据实际部署环境api地址IP替换

$token\_url = "http://127.0.0.1:3033/open\_api/authentication/get\_access\_token";

$token\_params = Array("key" => "123", "secret" => "456");

$headers = Array("Content-type: application/json");

$ch = curl\_init();//初始化curl

curl\_setopt($ch, CURLOPT\_URL, $token\_url);//抓取指定网页

curl\_setopt($ch, CURLOPT\_HEADER, 0);//设置header

curl\_setopt($ch, CURLOPT\_HTTPHEADER, $headers);//设置请求头

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);//要求结果为字符串且输出到屏幕上

curl\_setopt($ch, CURLOPT\_POST, 1);//post提交方式

curl\_setopt($ch, CURLOPT\_POSTFIELDS, json\_encode($token\_params));

$data = curl\_exec($ch);//运行curl

curl\_close($ch);

$res\_hash = json\_decode($data, true);

if ($res\_hash['message'] != 'ok') {

//错误处理

} else {

$access\_token = $res\_hash['result']['access\_token'];

#以学生基础信息列表查询API为例

$request\_params = Array();

$request\_url = "http://127.0.0.1:3033/open\_api/student/infos/list";

$headers = Array("Content-type: application/json");

$request\_params['access\_token'] = $access\_token;

$ch = curl\_init();//初始化curl

curl\_setopt($ch, CURLOPT\_URL, $request\_url);//抓取指定网页

curl\_setopt($ch, CURLOPT\_HEADER, 0);//设置header

curl\_setopt($ch, CURLOPT\_HTTPHEADER, $headers);//设置请求头

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);//要求结果为字符串且输出到屏幕上

curl\_setopt($ch, CURLOPT\_POST, 1);//post提交方式

curl\_setopt($ch, CURLOPT\_POSTFIELDS, json\_encode($request\_params));

$return\_data = curl\_exec($ch);//运行curl

curl\_close($ch);

}

## 5.5python调用示例

# 首先需要获取token，是根据key和secret生成的

import requests

import json

token\_params = {}

token\_url = "http://127.0.0.1:3033/open\_api/authentication/get\_access\_token"

token\_params['key'] = "123"

token\_params['secret'] = "456"

token\_body = requests.post(token\_url,

json.dumps(token\_params),headers={'content-type': 'application/json'})

res\_hash = json.loads(token\_body.text)

if res\_hash['message'] != 'ok':

# error 处理

pass

else:

#拿到access\_token

access\_token = res\_hash['result']['access\_token']

#以学生基础信息列表查询API为例

request\_params = {}

request\_url = "http://127.0.0.1:3033/open\_api/student/infos/list

request\_params['access\_token'] = access\_token

body = requests.post(request\_url,

json.dumps(request\_params),headers={'content-type': 'application/json'})

body = json.loads(body.text)