### 钉钉企业内部应用服务端集成--获取用户信息

1、****获取access\_token****

文档链接：<https://open-doc.dingtalk.com/microapp/serverapi2/eev437>

a.登录开发者后台，参考文档创建企业内部开发应用，获取appKey和秘钥appSecret，可使用范围以及权限管理详情请查看文档。

b.通过appKey & appSecret获取应用的access\_token，access\_token有效期为7200秒，有效期内重复获取返回结果相同，并自动续期，建议对access\_token进行缓存，过期重新获取。

请求地址（GET）：<https://oapi.dingtalk.com/gettoken?appkey=key&appsecret=secret>

返回结果：

{

"errcode": 0,

"errmsg": "ok",

"access\_token": "fw8ef8we8f76e6f7s8df8s"

}

1. ****获取免授权码—前端传入免登授权码code****

前端参考文档—前端API->H5微应用

文档地址：<https://open-doc.dingtalk.com/microapp/dev>

a.下载安装钉钉的jsapi组件，dingtalk-jsapi安装后可直接使用

b.调用JSAPI组件获取免登授权码code

调用方式：

dd.ready(function() {

// dd.ready参数为回调函数，在环境准备就绪时触发，jsapi的调用需要保证在该回调函数触发后调用，否则无效。

dd.runtime.permission.requestAuthCode({

corpId: "corpid",

onSuccess: function(result) {

/\*{ code: 'hYLK98jkf0m' //string authCode }\*/

},

onFail : function(err) {}

});

});

1. ****获取用户userid****

文档地址：<https://open-doc.dingtalk.com/microapp/serverapi2/clotub>

请求地址（GET）：<https://oapi.dingtalk.com/user/getuserinfo?access_token=access_token&code=code>

access\_token: 第一步获取的调用接口凭证

code：第二步获取的免登授权码

返回结果：

{

"userid": "\*\*\*\*",

"sys\_level": 1,

"errmsg": "ok",

"is\_sys": true,

"deviceId": "\*\*\*",

"errcode": 0

}

4、****获取用户详情****

文档地址：[https://open-doc.dingtalk.com/microapp/serverapi2/ege851#a-names1a获取用户详情](https://open-doc.dingtalk.com/microapp/serverapi2/ege851" \l "a-names1a%E8%8E%B7%E5%8F%96%E7%94%A8%E6%88%B7%E8%AF%A6%E6%83%85)

请求地址（GET）：<https://oapi.dingtalk.com/user/get?access_token=ACCESS_TOKEN&userid=zhangsan>

access\_token: 第一步获取的调用接口凭证

userid: 员工id

返回参数：

{ "errcode": 0, // 返回码

"unionid": "PiiiPyQqBNBii0HnCJ3zljcuAiEiE", // 员工在当前开发者企业账号范围内的唯一标识，系统生成，固定值，不会改变

"remark": "remark", // 备注

"userid": "zhangsan", // 员工在当前企业内的唯一标识

"isLeaderInDepts": "{1:false}", // 在对应的部门中是否为主管：Map结构的json字符串，key是部门的Id，value是人员在这个部门中是否为主管，true表示是，false表示不是

"isBoss": false, // 是否为企业的老板，true表示是，false表示不是

"hiredDate": 1520265600000, // 入职时间。Unix时间戳 （在OA后台通讯录中的员工基础信息中维护过入职时间才会返回)

"isSenior": false, // 是否是高管

"tel": "010-88996533", // 分机号（仅限企业内部开发调用）

"department": [1,2], // 成员所属部门id列表

"workPlace": "beijing", // 办公地点

"email": "ceshi@aliyun.com", // 邮箱

"orderInDepts": "{1:71738366882504}", // 在对应的部门中的排序，Map结构的json字符串，key是部门的Id，value是人员在这个部门的排序值

"mobile": "15901516821", // 手机号

"errmsg": "ok", // 对返回码的文本描述内容

"active": false, // 是否已经激活，true表示已激活，false表示未激活

"avatar": "dingtalk.com/abc.jpg", // 头像url

"isAdmin": false, // 是否为企业的管理员，true表示是，false表示不是

"isHide": false, // 是否号码隐藏，true表示隐藏，false表示不隐藏

"jobnumber": "001", // 员工工号

"name": "test1", // 员工名字

"extattr": {}, // 扩展属性，可以设置多种属性（但手机上最多只能显示10个扩展属性，具体显示哪些属性，请到OA管理后台->设置->通讯录信息设置和OA管理后台->设置->手机端显示信息设置）

"stateCode": "86", // 国家地区码

"position": "manager" // 职位信息

}

后端代码

配置文件

*#钉钉***dd**:  
 **config**:  
 *#应用的唯一标识key* **appkey** : dingfrcbtdz00abgqehq  
 **default-expire** : 7000  
 *#应用的密钥* **appsecret** : D7GL19IDFJ9eITi2wIbj-pXK45pIHcNcOvbuZZn8oxEEh4R3OrhXmDO-5YrREG0T  
 *#获取token请求地址* **access-token-url** : https://oapi.dingtalk.com/gettoken?appkey={0}&appsecret={1}  
 *#获取用户userid* **user-info-url** : https://oapi.dingtalk.com/user/getuserinfo?access\_token={0}&code={1}  
 *#获取用户详情信息地址* **user-details-url** : https://oapi.dingtalk.com/user/get?access\_token={0}&userid={1}

实现代码：

private String accessTokenKey = "access\_token";  
private String MobileIsHide = "isHide";  
private String userIdKey = "userid";  
private String mobile = "mobile";  
@Autowired  
private RedisUtils redisUtils;  
  
@Override  
public String getAccessToken(DDConfig ddConfig) throws IOException {  
 // redis 获取 redis token  
 String accessToken = redisUtils.get(RedisKeys.*ACCESS\_TOKEN*);  
 if (StringUtils.*isEmpty*(accessToken)) {  
 //获取token地址  
 String accessTokenUrl = ddConfig.getAccessTokenUrl();  
 //拼装参数  
 String url = StrUtil.*indexedFormat*(accessTokenUrl, ddConfig.getAppkey(), ddConfig.getAppsecret());  
 *log*.info("访问" + accessTokenKey + "请求地址:" + url);  
 //发送请求  
 JSONObject jsonObject = HttpUtil.*doGetStr*(url);  
 //判断json结果  
 if (Objects.*nonNull*(jsonObject)) {  
 if (Objects.*nonNull*(jsonObject.get(accessTokenKey))) {  
 accessToken = jsonObject.getString(accessTokenKey);  
 if (StringUtils.*isEmpty*(accessToken)) {  
 *log*.info("钉钉accessToken获取失败,fuck:" + ddConfig.getAppkey() + "," + "shit:" + ddConfig.getAppsecret());  
 AssertUtil.*err*(CodeMsg.*SERVER\_AURH\_ERR*);  
 }  
 //重新设置缓存信息  
 redisUtils.set(RedisKeys.*ACCESS\_TOKEN*, accessToken, ddConfig.getDdtokenDefaultExpire());  
 return accessToken;  
 }  
 *log*.info("钉钉accessToken请求结果:" + jsonObject.toString());  
 AssertUtil.*err*(CodeMsg.*SERVER\_AURH\_ERR*);  
 }  
 }  
 return accessToken;  
}  
  
@Override  
public String getDdUserId(DDConfig ddConfig, String authCode) throws IOException {  
 AssertUtil.*isNull*(ddConfig, CodeMsg.*DD\_CONFIG\_ERR*, "无法获取到ddConfig对象");  
 AssertUtil.*isNull*(authCode, CodeMsg.*DD\_CONFIG\_ERR*, "访问 getDdUserId 不能把设置空 authCode");  
 String accessToken = getAccessToken(ddConfig);  
 String userInfoUrl = ddConfig.getUserInfoUrl();  
 String url = StrUtil.*indexedFormat*(userInfoUrl, accessToken, authCode);  
 *log*.info("访问用户id请求地址:" + url);  
 //发送http链接  
 JSONObject jsonObject = HttpUtil.*doGetStr*(url);  
 AssertUtil.*isNull*(jsonObject, CodeMsg.*DD\_CONFIG\_ERR*, "获取userInfoUrl返回空信息,url:" + url);  
 String s = jsonObject.toString();  
 AssertUtil.*isNull*(Objects.*isNull*(jsonObject.get(userIdKey)), CodeMsg.*DD\_CONFIG\_ERR*, "获取userInfoUrl返回jsonObject获取不到用户信息,data:" + s);  
 //获取用户id  
 String userId = jsonObject.getString(userIdKey);  
 AssertUtil.*isBlank*(userId, CodeMsg.*DD\_CONFIG\_ERR*, "获取userInfoUrl返回jsonObject获取不到用户userid,data:" + jsonObject.toString());  
 return userId;  
}  
  
@Override  
public JSONObject getUserInfo(DDConfig ddConfig, String userId) throws IOException {  
 AssertUtil.*isNull*(ddConfig, CodeMsg.*DD\_CONFIG\_ERR*, "无法获取到ddConfig对象");  
 AssertUtil.*isNull*(userId, CodeMsg.*DD\_CONFIG\_ERR*, "访问 getUserInfo 不能把设置空 userId");  
 String accessToken = getAccessToken(ddConfig);  
 String userDetailsUrl = ddConfig.getUserDetailsUrl();  
 String url = StrUtil.*indexedFormat*(userDetailsUrl, accessToken, userId);  
 *log*.info("访问用户id详情请求地址:" + url);  
 JSONObject jsonObject = HttpUtil.*doGetStr*(url);  
 AssertUtil.*isNull*(jsonObject, CodeMsg.*DD\_CONFIG\_ERR*, "获取userInfoUrl返回空信息,url:" + url);  
 return jsonObject;  
}  
  
@Override  
public String getUserMobile(DDConfig ddConfig, String userId) throws IOException {  
 AssertUtil.*isNull*(ddConfig, CodeMsg.*DD\_CONFIG\_ERR*, "无法获取到ddConfig对象");  
 AssertUtil.*isNull*(userId, CodeMsg.*DD\_CONFIG\_ERR*, "访问 getUserInfo 不能把设置空 userId");  
 JSONObject userInfo = getUserInfo(ddConfig, userId);  
 //号码是否隐藏  
 boolean isHide = userInfo.getBoolean(MobileIsHide);  
 AssertUtil.*isErr*(isHide, (CodeMsg.*PHONE\_HIDE*));  
 String phone = userInfo.getString(mobile);  
 AssertUtil.*isBlank*(phone, "手机号码错误,请检查您的钉钉设置", " getUserMobile 获取 用户 mobile 失败,找不到mobile userInfo:" + userInfo.toString());  
 return phone;  
}