**Android 开发文档**

|  |
| --- |
| **OneLogin（一键登录）** |
| 进阶模式和常规模式是一键登录的两种调用逻辑，如无特别需求建议直接使用进阶模式，两模式不可 混合调用。  **常规模式** |

**1、调用逻辑**

预取号和授权页拉起时机均由开发者自主控制。使用此调用逻辑，开发者需自主处理预取号的时机和 超期重试的逻辑等。

1. init() 初始化 SDK 并配置 APPID

2. preGetToken() 预取号（应用启动或进入登录页的前一个页面、用户登出时是调用该方法的时 机）

3. requestToken() 拉起授权页面（调用该方法前**建议先调用 isPreGetTokenResultValidate 判断预 取号是否成功**）

4. dismissAuthActivity() 关闭授权页

 注：同一个流程 preGetToken 只可调用一次，否则可能出现后一次调用无回调结果的情况

**2、初始化**

. 初始化 init

SDK 初始化接口 init ，可放在 调用。

**方法描述**

Application

的

onCreate

方法中进行初始化，或者其他预取号之前

**public** OneLoginHelper init(Context context);

**public** OneLoginHelper init(Context context, String appId);

**参数说明**

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| context | Context | 上下文 |
| appId | String | 后台配置唯一产品 APPID ，请在官网申请 |

**3、预取号**

. 预取号 preGetToken

**方法描述**

预取号接口，可同 init 方法一起放在 Application 的 onCreate 方法中进行注册，或者在登录页前一 个页面初始化时调用；登录成功后预取号失效，如果用户退出登录后，为了方便下次重新登录能快速 拉起授权页，也可以重新调用 preGetToken 进行预取号。

**public** void preGetToken(String appId, AbstractOneLoginListener oneLoginListener);

**public** void preGetToken(String appId, int timeout, AbstractOneLoginListener oneLoginListener);

**参数说明**

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| appId | String | 产品 APPID, 如 init 接口 传了 appId 此处可传空值 |
| timeout | int | 超时时间，单位: ms ，取 值范围: 1000~15000 ，默 认 8000 。传递该参数会统 一设置预取号超时时间和 取号超时时间为设定值，  分别设置请使用不带  timeout 的方法并参考 setRequestTimeout 。 |
| oneLoginListener | AbstractOneLoginListener | 回调监听器，需要开发者 自己实现 |

**代码示例**

OneLoginHelper .with()

.setLogEnable(**true**)

*//第一个参数为当前* *Application 或* *Activity 的* *Context //第二个参数为所需要配置的* *APPID*

.init(context, APPID)

.preGetToken("", **new** AbstractOneLoginListener() {

@Override

**public** void onResult(JSONObject jsonObject) { oneLoginResult.onResult();

Log.i(TAG, "预取号结果为： " + jsonObject.toString()); **try** {

int status = jsonObject.getInt("status");

*// status=200 为预取号成功，* *其他返回码请参考返回码章节*

**if** (status == 200) {

**if** (isRequestToken) { *// 预取号成功后，* *可以调用拉起授权页* oneLoginRequestToken();

}

} **else** {

ToastUtils.toastMessage(context, "预取号失败 :" + jsonObject.toString()); }

} **catch** (JSONException e) {

ToastUtils.toastMessage(context, "预取号失败 :" + jsonObject.toString()); }

} });

};

**4、拉起授权页**

. 拉起授权页 requestToken

**方法描述**

在需要登录的地方调用 requestToken 接口拉起一键登录授权页，待用户点一键登录授权后获取运营商 token ，获取成功后即可请求服务端换取本机手机号码。

**public** void requestToken(OneLoginThemeConfig oneLoginThemeConfig, AbstractOneLoginListener listener)

**参数说明**

|  |  |  |
| --- | --- | --- |
| **参数** | **类型** | **说明** |
| oneLoginThemeConﬁg | OneLoginThemeConﬁg | 自定义全局配置接口，用 来配置授权页面 UI 样式 |
| listener | AbstractOneLoginListener | 回调监听器，需要开发者 自己实现 |

**代码示例**

取号并获取免密登录的 token ，通过接口进行校验，并获取登录信息。

**if** (!OneLoginHelper.with().isPreGetTokenResultValidate()) {

*//预取号已过期或者失效，* *需要重新预取号, 此处可显示自定义加载对话框*

**return**; }

OneLoginHelper .with()

.requestToken(**new** OneLoginThemeConfig.Builder().build(), **new** AbstractOneLoginListener() {

@Override

**public** void onResult(JSONObject jsonObject) { **try** {

int statusResult = jsonObject.getInt("status");

*// status=200 为取号成功，* *其他返回码请参考返回码章节*

**if** (statusResult != 200) {

Toast.makeText(getApplicationContext(), jsonObject.toString(), Toast.LENGTH\_SHORT).show();

**return**; }

**new** Thread(() -> {

JSONObject json = **new** JSONObject(); **try** {

json.put("process\_id", jsonObject.getString("process\_id")); json.put("token", jsonObject.getString("token"));

json.put("authcode", jsonObject.optString("authcode")); *//必选参数* } **catch** (JSONException e) {

e.printStackTrace(); }

String result = HttpUtils.requestNetwork(<---获取用户登录信息的接口 --->, json);

*///*

*///* **TODO** *判断* *result 是否有效* *///*

}).start();

} **catch** (JSONException e) {

e.printStackTrace();

}

} });

**5、关闭授权页**

. 关闭授权页 dismissAuthActivity

**方法描述**

用户主动关闭授权页。 SDK 除了返回按钮触发关闭以外，默认是不 ﬁnsih 授权页的，需要开发者**在回 调结束后自行实现关闭授权页**。

**public** void dismissAuthActivity()

**代码示例**

OneLoginHelper.with().dismissAuthActivity();