# 【Unity】中实现微信、QQ、微博帐号登录（Android环境）（官方SDK）

对于一些没有过安卓开发经验的开发者来说，为了实现安卓第三方登陆肯定是费尽了脑汁，不知道从何着手解决，为了让开发者少走弯路，下面就给大家介绍下Unity中使用官方SDK实现微信、QQ、微博帐号登录的方法，一起来看看吧。

一、新建安卓工程并导入jar文件

　　注意写好自己的包名，微信登录和微博登录都要用到包名获取签名，把获取到的签名上传到开放平台。

　　填好包名后一直点next就可以了，点击finish完成创建。

　　创建完成后，新建一个module类，同样写好自己的包名。

二、QQ登录

QQ登录时最简单的一个，因为不需要签名。在qq开放平台下载好sdk，然后copy到libs文件夹下。

　　点击Add as Library就可以了，微信和微博的jar包导入也是一样，Unity的classes.jar也需要导入，后面就不赘述了。

　　通过调用Tencent类的login函数发起登录/校验登录态。

　　该API具有两个作用：

　　（1）如果开发者没有调用mTencent实例的setOpenId、setAccessToken API，则该API执行正常的登录操作；

　　（2）如果开发者先调用mTencent实例的setOpenId、setAccessToken API，则该API执行校验登录态的操作。如果登录态有效，则返回成功给应用，如果登录态失效，则会自动进入登录流程，将最新的登录态数据返回给应用。

建议开发者在每次应用启动时调用一次该API(先调用setOpenId、setAccessToken)，以确保每次打开应用时用户都是有登录态的。

public class MainActivity extends UnityPlayerActivity {

private Tencent mTencent;

private BaseUiListener mBaseUiListener=new BaseUiListener();

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

mTencent = Tencent.createInstance("你的APPID", this.getApplicationContext());

}

/\*\*qq登录代码 Unity里调用\*/

public void QQlogin() {

//登录代码。。。

if (!mTencent.isSessionValid()) {

mTencent.login(this, "", mBaseUiListener);

}

}

/\*\*qq请求回调\*/

private class BaseUiListener implements IUiListener {

@Override

public void onComplete(Object o) {

JSONObject response=(JSONObject)o;

//把请求到的参数传给unity

UnityPlayer.UnitySendMessage("ThirdLoginGameobject","QQCallBack",o.toString());

}

@Override

public void onError(UiError uiError) { }

@Override

public void onCancel() { }

}

/\*\*

\* qq登录页面返回处理

\* @param requestCode

\* @param resultCode

\* @param data

\*/

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

Tencent.onActivityResultData(requestCode,resultCode,data,mBaseUiListener);

}

　　然后再mainfest里配置权限。

"android.permission.INTERNET">

"android.permission.ACCESS\_NETWORK\_STATE">

"com.tencent.tauth.AuthActivity" android:nohistory="true" android:launchmode="singleTask">

"android.intent.action.VIEW">

"android.intent.category.DEFAULT">

"android.intent.category.BROWSABLE">

"tencent你的AppId">

SDK\_V2.0引入了AssistActivity，开发者需在androidManifest.xml中注册。代码如下：

"com.tencent.connect.common.AssistActivity"

android:theme="@android:style/Theme.Translucent.NoTitleBar" android:configchanges="orientation|keyboardHidden|screenSize">

　　现在就可以成功拉起QQ授权页面并登陆了。

三、微信登录

　　新建一个名为wxapi的包和WXEntryActivity的类。

WXEntryActivity的代码如下：

public class WXEntryActivity extends Activity implements IWXAPIEventHandler {

private static final String WEIXIN\_ACCESS\_TOKEN\_KEY = "wx\_access\_token\_key";

private static final String WEIXIN\_OPENID\_KEY = "wx\_openid\_key";

private static final String WEIXIN\_REFRESH\_TOKEN\_KEY = "wx\_refresh\_token\_key";

public static WeChatCode mWeChatCode;

public WeChatCode getWeChatCode() {

return mWeChatCode;

}

public void setWeChatCode(WeChatCode weChatCode) {

mWeChatCode = weChatCode;

}

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// 微信事件回调接口注册

MyApplication.sApi.handleIntent(getIntent(), this);

}

//微信组件注册初始化

public static IWXAPI initWeiXin(Context context, @NonNull String weixin\_app\_id) {

if (TextUtils.isEmpty(weixin\_app\_id)) {

Toast.makeText(context.getApplicationContext(), "app\_id 不能为空", Toast.LENGTH\_SHORT).show();

}

IWXAPI api = WXAPIFactory.createWXAPI(context, weixin\_app\_id, true);

api.registerApp(weixin\_app\_id);

return api;

}

/\*\*登录微信\*/

public static void loginWeixin(Context context, IWXAPI api, WeChatCode wechatCode) {

// 判断是否安装了微信客户端

// if (!api.isWXAppInstalled()) {

// Toast.makeText(context.getApplicationContext(), "您还未安装微信客户端！", Toast.LENGTH\_SHORT)

// .show();

// return;

// }

mWeChatCode=wechatCode;

// 发送授权登录信息，来获取code

SendAuth.Req req = new SendAuth.Req();

// 应用的作用域，获取个人信息

req.scope = "snsapi\_userinfo";

/\*\*

\* 用于保持请求和回调的状态，授权请求后原样带回给第三方

\* 为了防止csrf攻击（跨站请求伪造攻击），后期改为随机数加session来校验

\*/

req.state = "app\_wechat";

api.sendReq(req);

}

// 微信发送请求到第三方应用时，会回调到该方法

@Override

public void onReq(BaseReq req) {

switch (req.getType()) {

case ConstantsAPI.COMMAND\_GETMESSAGE\_FROM\_WX:

break;

case ConstantsAPI.COMMAND\_SHOWMESSAGE\_FROM\_WX:

break;

default:

break;

}

}

// 第三方应用发送到微信的请求处理后的响应结果，会回调到该方法

@Override

public void onResp(BaseResp resp) {

switch (resp.errCode) {

// 发送成功

case BaseResp.ErrCode.ERR\_OK:

// 获取code

String code = ((SendAuth.Resp) resp).code;

mWeChatCode.getResponse(code);

break;

}

}

/\*\*\* 返回code的回调接口\*/

public interface WeChatCode {

void getResponse(String code);

}

}

MainActivity中添加如下代码：

private static final String WEIXIN\_ACCESS\_TOKEN\_KEY = "wx\_access\_token\_key";

private static final String WEIXIN\_OPENID\_KEY = "wx\_openid\_key";

private static final String WEIXIN\_REFRESH\_TOKEN\_KEY = "wx\_refresh\_token\_key";

微信登录 unity中调用该方法

public void weiLogin() {

WXEntryActivity.loginWeixin(MainActivity.this, MyApplication.sApi, new WXEntryActivity.WeChatCode() {

@Override

public void getResponse(String code) {

// 通过code获取授权口令access\_token

getAccessToken(code);

Log.i("获取token成功",code.toString());

// 从手机本地获取存储的授权口令信息，判断是否存在access\_token，不存在请求获取，存在就判断是否过期

String accessToken = (String) ShareUtils.getValue(MainActivity.this, WEIXIN\_ACCESS\_TOKEN\_KEY,"none");

String openid = (String) ShareUtils.getValue(MainActivity.this, WEIXIN\_OPENID\_KEY, "");

if (!"none".equals(accessToken)) {

// 有access\_token，判断是否过期有效

isExpireAccessToken(accessToken, openid);

} else {

// 没有access\_token

getAccessToken(code);

Log.i("获取token成功",code.toString());

}

}

});

Log.i("登录成功","aaaaaaaaaaaaaaaaaaaaaaaaaa");

}

/\*\*\* 微信登录获取授权口令 \*/

private void getAccessToken(String code) {

String url = "https://api.weixin.qq.com/sns/oauth2/access\_token?" +

"appid=" + AppConst.WEIXIN\_APP\_ID +

"&secret=" + AppConst.WEIXIN\_APP\_SECRET +

"&code=" + code +

"&grant\_type=authorization\_code";

// 网络请求获取access\_token

OkHttpUtils.get().url(url).build().execute(new StringCallback() {

@Override

public void onError(Call call, Exception e, int id) {

e.printStackTrace();

}

@Override

public void onResponse(String response, int id) {

// 判断是否获取成功，成功则去获取用户信息，否则提示失败

processGetAccessTokenResult(response);

}

});

}

/\*\*\* 微信登录处理获取的授权信息结果

\* @param response 授权信息结果

\*/

public void processGetAccessTokenResult(String response) {

// 验证获取授权口令返回的信息是否成功

if (validateSuccess(response)) {

// 使用Gson解析返回的授权口令信息

// WXAccessTokenInfo tokenInfo = mGson.fromJson(response, WXAccessTokenInfo.class);

// 保存信息到手机本地

// saveAccessInfotoLocation(tokenInfo);

// 获取用户信息

// getUserInfo(tokenInfo.getAccess\_token(), tokenInfo.getOpenid());

} else {

// 授权口令获取失败，解析返回错误信息

// WXErrorInfo wxErrorInfo = mGson.fromJson(response, WXErrorInfo.class);

}

}

/\*\*

\*微信登录获取tokenInfo的WEIXIN\_OPENID\_KEY，WEIXIN\_ACCESS\_TOKEN\_KEY，WEIXIN\_REFRESH\_TOKEN\_KEY保存到shareprephence中

\* @param tokenInfo

\*/

private void saveAccessInfotoLocation(WXAccessTokenInfo tokenInfo) {

ShareUtils.saveValue(MyApplication.mContext,WEIXIN\_OPENID\_KEY,tokenInfo.getOpenid());

ShareUtils.saveValue(MyApplication.mContext,WEIXIN\_ACCESS\_TOKEN\_KEY,tokenInfo.getAccess\_token());

ShareUtils.saveValue(MyApplication.mContext,WEIXIN\_REFRESH\_TOKEN\_KEY,tokenInfo.getRefresh\_token());

}

/\*\* \* 验证是否成功\*

\* @param response 返回消息

\* @return 是否成功

\*/

private boolean validateSuccess(String response) {

String errFlag = "errmsg";

return (errFlag.contains(response) && !"ok".equals(response))

|| (!"errcode".contains(response) && !errFlag.contains(response));

}

/\*\*\* 微信登录判断accesstoken是过期

\* @param accessToken token

\* @param openid 授权用户唯一标识

\*/

private void isExpireAccessToken(final String accessToken, final String openid) {

String url = "https://api.weixin.qq.com/sns/auth?" +

"access\_token=" + accessToken +

"&openid=" + openid;

OkHttpUtils.get().url(url).build().execute(new StringCallback() {

@Override

public void onError(Call call, Exception e, int id) {

}

@Override

public void onResponse(String response, int id) {

if (validateSuccess(response)) {

// accessToken没有过期，获取用户信息

getUserInfo(accessToken, openid);

//把请求到的数据传递到unity

UnityPlayer.UnitySendMessage("与安卓交互的C#脚本挂载的物体名","函数名",response.toString());

Toast.makeText(getApplicationContext(),response.toString(),Toast.LENGTH\_LONG).show();

} else {

// 过期了，使用refresh\_token来刷新accesstoken

refreshAccessToken();

}

}

});

}

/\*\* \* 微信登录刷新获取新的access\_token \*/

private void refreshAccessToken() {

// 从本地获取以存储的refresh\_token

final String refreshToken = (String) ShareUtils.getValue(this, WEIXIN\_REFRESH\_TOKEN\_KEY, "");

if (TextUtils.isEmpty(refreshToken)) {

return;

}

// 拼装刷新access\_token的url请求地址

String url = "https://api.weixin.qq.com/sns/oauth2/refresh\_token?" +

"appid=" + AppConst.WEIXIN\_APP\_ID +

"&grant\_type=refresh\_token" +

"&refresh\_token=" + refreshToken;

// 请求执行

OkHttpUtils.get().url(url).build().execute(new StringCallback() {

@Override

public void onError(Call call, Exception e, int id) {

System.out.println("刷新获取新的access\_token信息失败！！！");

// 重新请求授权

weiLogin();

}

@Override

public void onResponse(String response, int id) {

// 判断是否获取成功，成功则去获取用户信息，否则提示失败

processGetAccessTokenResult(response);

}

});

}

/\*\* \* 微信token验证成功后，联网获取用户信息

\* @param access\_token

\* @param openid

\*/

private void getUserInfo(String access\_token, String openid) {

String url = "https://api.weixin.qq.com/sns/userinfo?" +

"access\_token=" + access\_token +

"&openid=" + openid;

OkHttpUtils.get().url(url).build().execute(new StringCallback() {

@Override

public void onError(Call call, Exception e, int id) {

System.out.println("联网获取用户信息失败！！！");

}

@Override

public void onResponse(String response, int id) {

// 解析获取的用户信息

// WXUserInfo userInfo = mGson.fromJson(response, WXUserInfo.class);

System.out.println("获取用户信息String是：：：：：："+response);

}

});

}

　　写了一个类来存APPID和SECRET

　　新建一个MyApplication类来初始化微信登录和网络请求

public class MyApplication extends Application {

public static IWXAPI sApi;

public static Context mContext;

@Override

public void onCreate() {

super.onCreate();

mContext=this;

//初始化微信

sApi = WXEntryActivity.initWeiXin(this, AppConst.WEIXIN\_APP\_ID);

initOkHttp();

}

//封装okhttp框架的初始化配置

private void initOkHttp() {

HttpsUtils.SSLParams sslParams = HttpsUtils.getSslSocketFactory(null, null, null);

CookieJarImpl cookieJar = new CookieJarImpl(new PersistentCookieStore(getApplicationContext()));

OkHttpClient okHttpClient = new OkHttpClient.Builder()

.addInterceptor(new LoggerInterceptor("TAG"))

.cookieJar(cookieJar)

.connectTimeout(20000L, TimeUnit.MILLISECONDS)

.readTimeout(20000L,TimeUnit.MILLISECONDS)

.writeTimeout(20000L,TimeUnit.MILLISECONDS)

.sslSocketFactory(sslParams.sSLSocketFactory, sslParams.trustManager)

//其他配置

.build();

OkHttpUtils.initClient(okHttpClient);

}

}

mainfest里初始化MyApplication

　　只要填写正确的APPID以及签名正确，就可以拉起微信登录页面了。

四、微博登录

　　微博登录不光要导入jar包，libs里面的.so文件也要导入，这里只讲客户端授权。

　　从官方demo中复制AccessTokenKeeper这个类到你的工程中。写一个Constance类来存APPKEY，把这三个类也从官方的demo中复制过来，下面贴上我删减后的代码：

public class WBAuthActivity extends Activity {

private static final String TAG = "weibosdk";

private AuthInfo mAuthInfo;

/\*\* 封装了 "access\_token"，"expires\_in"，"refresh\_token"，并提供了他们的管理功能 \*/

private Oauth2AccessToken mAccessToken;

/\*\* 注意：SsoHandler 仅当 SDK 支持 SSO 时有效 \*/

private SsoHandler mSsoHandler;

/\*\*\* @see {@link Activity#onCreate}\*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

mAuthInfo = new AuthInfo(this, Constants.APP\_KEY, Constants.REDIRECT\_URL, Constants.SCOPE);

mSsoHandler = new SsoHandler(WBAuthActivity.this, mAuthInfo);

// SSO 授权, 仅客户端

WBLogin();

// 从 SharedPreferences 中读取上次已保存好 AccessToken 等信息，

// 第一次启动本应用，AccessToken 不可用

mAccessToken = AccessTokenKeeper.readAccessToken(this);

if (mAccessToken.isSessionValid()) {

updateTokenView(true);

}

}

/\*\*当 SSO 授权 Activity 退出时，该函数被调用。

\* @see {@link Activity#onActivityResult}

\*/

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

// SSO 授权回调

// 重要：发起 SSO 登陆的 Activity 必须重写 onActivityResults

// ++++++++++++++++++

//如果是sso授权登陆，必须先执行下面的回调，然后才会执行监听器里的 回调

if (mSsoHandler != null) {

mSsoHandler.authorizeCallBack(requestCode, resultCode, data);

}

}

private void WBLogin() {

mSsoHandler.authorizeClientSso(new AuthListener());

}

/\*\*

\* 微博认证授权回调类。

\* 1. SSO 授权时，需要在 {@link #onActivityResult} 中调用 {@link SsoHandler#authorizeCallBack} 后，

\* 该回调才会被执行。

\* 2. 非 SSO 授权时，当授权结束后，该回调就会被执行。

\* 当授权成功后，请保存该 access\_token、expires\_in、uid 等信息到 SharedPreferences 中。

\*/

class AuthListener implements WeiboAuthListener {

@Override

public void onComplete(Bundle values) {

// 从 Bundle 中解析 Token

mAccessToken = Oauth2AccessToken.parseAccessToken(values);

UnityPlayer.UnitySendMessage("ThirdLoginGameobject","WeiboCallBack",values.toString());

//从这里获取用户输入的 电话号码信息

// String phoneNum = mAccessToken.getUid();

if (mAccessToken.isSessionValid()) {

// 显示 Token

updateTokenView(false);

} else {

// 以下几种情况，您不会收到 Code：

// 1. 当您未在平台上注册的应用程序的包名与签名时；

// 2. 当您注册的应用程序包名与签名不正确时；

// 3. 当您在平台上注册的包名和签名与您当前测试的应用的包名和签名不匹配时。

}

}

@Override

public void onCancel() { }

@Override

public void onWeiboException(WeiboException e) { }

}

/\*\*显示当前 Token 信息。

\* @param hasExisted 配置文件中是否已存在 token 信息并且合法

\*/

private void updateTokenView(boolean hasExisted) { }

}

public class WBAuthCodeActivity extends Activity {

private static final String TAG = "WBAuthCodeActivity";

/\*\*

\* WeiboSDKDemo 程序的 APP\_SECRET。

\* 请注意：请务必妥善保管好自己的 APP\_SECRET，不要直接暴露在程序中，此处仅作为一个DEMO来演示。

\*/

/\*\* 通过 code 获取 Token 的 URL \*/

private static final String OAUTH2\_ACCESS\_TOKEN\_URL = "https://open.weibo.cn/oauth2/access\_token";

/\*\* 微博 Web 授权接口类，提供登陆等功能 \*/

private AuthInfo mAuthInfo;

/\*\* 获取到的 Code \*/

private String mCode;

/\*\* 获取到的 Token \*/

private Oauth2AccessToken mAccessToken;

/\*\*@see {@link Activity#onCreate}\*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// setContentView(R.layout.activity\_auth\_code);

// 初始化微博对象

mAuthInfo = new AuthInfo(this, Constants.APP\_KEY, Constants.REDIRECT\_URL, Constants.SCOPE);

}

/\*\*\* 微博认证授权回调类。\*/

class AuthListener implements WeiboAuthListener {

@Override

public void onComplete(Bundle values) {

if (null == values) {

return;

}

String code = values.getString("code");

if (TextUtils.isEmpty(code)) {

return;

}

mCode = code;

}

@Override

public void onCancel() { }

@Override

public void onWeiboException(WeiboException e) {

}

}

/\*\*异步获取 Token。

\* @param authCode 授权 Code，该 Code 是一次性的，只能被获取一次 Token

\* @param appSecret 应用程序的 APP\_SECRET，请务必妥善保管好自己的 APP\_SECRET，

\* 不要直接暴露在程序中，此处仅作为一个DEMO来演示。

\*/

public void fetchTokenAsync(String authCode, String appSecret) {

WeiboParameters requestParams = new WeiboParameters(Constants.APP\_KEY);

requestParams.put(WBConstants.AUTH\_PARAMS\_CLIENT\_ID, Constants.APP\_KEY);

requestParams.put(WBConstants.AUTH\_PARAMS\_CLIENT\_SECRET, appSecret);

requestParams.put(WBConstants.AUTH\_PARAMS\_GRANT\_TYPE, "authorization\_code");

requestParams.put(WBConstants.AUTH\_PARAMS\_CODE, authCode);

requestParams.put(WBConstants.AUTH\_PARAMS\_REDIRECT\_URL, Constants.REDIRECT\_URL);

// 异步请求，获取 Token

new AsyncWeiboRunner(getApplicationContext()).requestAsync(OAUTH2\_ACCESS\_TOKEN\_URL, requestParams, "POST", new RequestListener() {

@Override

public void onComplete(String response) {

// 获取 Token 成功

Oauth2AccessToken token = Oauth2AccessToken.parseAccessToken(response);

if (token != null && token.isSessionValid()) {

mAccessToken = token;

String date = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(

new java.util.Date(mAccessToken.getExpiresTime()));

String format = token.toString();

} else { }

}

@Override

public void onWeiboException(WeiboException e) { }

});

}

}

WBShareMainActivity继承Activity为空即可，在MainActivity的onCreate里写上LogUtil.sIsLogEnable = true;

MainActivity里加上微博登录代码

//微博登录 Unity里调用

public void WBLogin() {

startActivity(new Intent(MainActivity.this, WBAuthActivity.class));

}

mainfest里加上 即可。

　　把MainActivity里的代码整合到一起后，build

　　把libs里的classes.jar删掉，把红框圈起来的classes.jar拖到libs文件夹里，然后把libs，res和AndroidMainfest这三个文件拖到Unity工程的Plugins/Android文件夹下。

　　然后在unity创建三个按钮并添加点击事件

public void OnQQBtnClick(ButtonScript obj, object args, object param1, object param2)

{

#if UNITY\_IPHONE

c\_thirdQQLogin();

#elif UNITY\_ANDROID

AndroidJavaClass jc = new AndroidJavaClass("com.unity3d.player.UnityPlayer");

AndroidJavaObject jo = jc.GetStatic("currentActivity");

jo.Call("QQlogin");

#endif

}

public void OnWeChatBtnClick(ButtonScript obj, object args, object param1, object param2)

{

#if UNITY\_IPHONE

c\_thirdLogin();

#elif UNITY\_ANDROID

AndroidJavaClass jc = new AndroidJavaClass("com.unity3d.player.UnityPlayer");

AndroidJavaObject jo = jc.GetStatic("currentActivity");

jo.Call("weiLogin");

#endif

}

public void OnWeiboBtnClick(ButtonScript obj, object args, object param1, object param2)

{

#if UNITY\_IPHONE

c\_thirdWeiboLogin();

#elif UNITY\_ANDROID

AndroidJavaClass jc = new AndroidJavaClass("com.unity3d.player.UnityPlayer");

AndroidJavaObject jo = jc.GetStatic("currentActivity");

jo.Call("WBLogin");

#endif

}

　　发布安卓应用的时候记得签名，不然微信和微博登录会报错。

---------------------

原文：https://blog.csdn.net/swj524152416/article/details/55261781