

微博 Android 平台 SDK 文档

北京新潮讯捷信息技术有限公司

编号：WEIBO_ANDROID_SDK

版本：WEIBO_ANDROID_SDK V2.1.0

修订记录

时间	文档版本	修订人	备注
2012/7/20	2.0.0	罗棚	初稿
2012/8/02	2.0.0	张晓伟	
2013/4/17	2.1.0	唐庆杰	新增分享微博

目录

微博 Android 平台 SDK 文档 1

 一、 概述 3

 名词解释..... 3

 二、 流程 3

 三、 集成步骤及示例分析（认证授权） 3

 1. 通过 Oauth2.0 授权获得 AccessToken 3

 四、 集成步骤及示例分析（分享微博） 6

 1. 初始化 SDK 6

 2. 注册到新浪微博..... 7

 3. 发送请求消息给微博..... 7

 4. 接收微博请求消息..... 9

一、概述

微博 SDK 为开发者提供访问 oauth2.0 授权认证，并集成 sso 登录功能，使第三方应用可通过新浪微博官方客户端快速通过 OAuth2.0 授权，并完成用户登录操作。提供微博分享功能，可直接通过微博客户端分享微博。

本文档将对使用 SDK 时所用的一些参数、接口进行说明，并分析一个简单示例，帮助第三方方便的使用 SDK（一些不使用的接口只做简单说明）。

名词解释

AppKey	分配给每个第三方应用的 app key。用于鉴权身份，显示来源等功能。
AccessToken	表示用户身份的 token，用于微博 API 的调用。
Expire in	过期时间，用于判断登录是否过期。
RedirectURI	应用回调页面，可在新浪微博开放平台->我的应用->应用信息->高级应用->授权设置->应用回调页中找到。

二、流程

1: 认证授权流程

使用带 SSO 功能的 SDK 进行登录，只需调用登录接口，并完成回调方法对接收登录结果即可。SDK 中自动完成对是否进行 SSO 登录的判断，若支持，则唤起微博客户端，用户确认后返回请求的应用，SDK 对请求结果进行解析，最后交给第三方实现的回调方法进行处理；否则，SDK 将通过内置浏览器请求登录，用户输入用户名密码提交后，仍是 SDK 解析请求结果，并由第三方应用实现相应的回调方法进行最后处理。

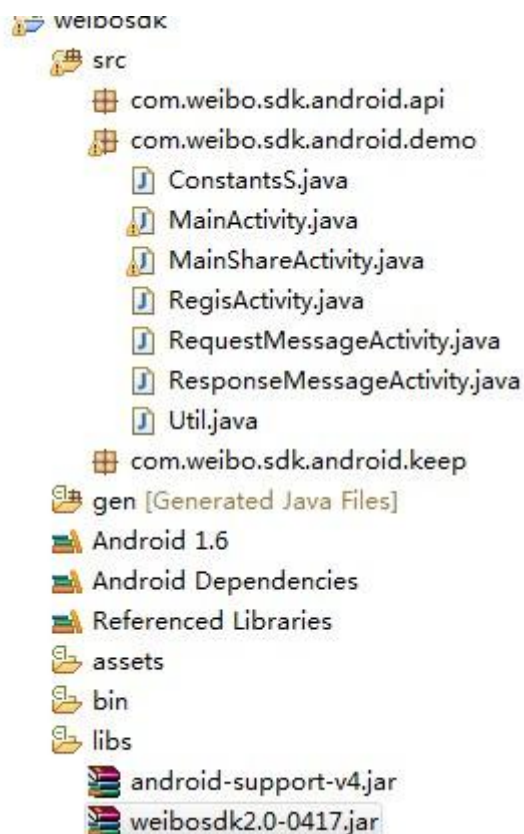
2: 微博分享流程

微博分享分两种场景：一是从第三方应用分享信息到微博；二是微博主动呼起第三方应用，并提取信息返回到微博客户端，进行分享。

三、集成步骤及示例分析（认证授权）

1. 通过 OAuth2.0 授权获得 AccessToken

Demo 工程:(注, 请用 debug.keystore 替换本地的 debug.keystore, 否则 demo 无法正常运行)



第一步 in MainActivity:

通过 Weibo 的构造函数设置应用回调页和 appkey。

```
mWeibo = Weibo.getInstance(APP_KEY, REDIRECT_URL);
```

对于 Mobile 客户端应用来说，是不存在 Server 的，故此处的应用回调页地址只要与新浪微博开放平台->我的应用->应用信息->高级应用->授权设置->应用回调页中的 url 地址保持一致就可以了，如图所示：



注：appkey 和 redirect_uri 在开放平台（<https://open.weibo.cn/>）上获取。

第二步注册应用程序的包名和签名：

packagename: 向开放平台注册应用程序的包名；

key_hash: 向开放平台注册应用程序的签名(经过 hash 后的值, 可通过 app_signatures.apk

获取)

目前只能邮件给相关人员进行人工注册。联系方式：蒋斌(jiangbin1@staff.sina.com.cn)。

邮件内容：

- 1: appkey
- 2: packagename
- 3: key_hash

第三步：

实现 WeiboDialogListener 接口。授权成功后可在 onComplete 函数中获得 accesstoken 信息。具体如何保存、使用 accesstoken 信息由开发者自行处理。

```
class AuthDialogListener implements WeiboAuthListener {

    @Override
    public void onComplete(Bundle values) {
        String token = values.getString("access_token");
        String expires_in = values.getString("expires_in");
        MainActivity.accessToken = new OAuth2AccessToken(token, expires_in);
        if (MainActivity.accessToken.isSessionValid()) {
            String date = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss").format(new java.util.Date(MainActivity.accessToken.getExpiresTime()));
            mText.setText("认证成功: \n\n access_token: " + token + "\n\n" + "expires_in: " + expires_in + "\n\n有效期: " + date);

            apiBtn.setVisibility(View.VISIBLE);
            AccessTokenKeeper.keepAccessToken(MainActivity.this, accessToken); //将OAuth2AccessToken保存到SharedPreferences中
            Toast.makeText(MainActivity.this, "认证成功", Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public void onError(WeiboDialogError e) {
        Toast.makeText(getApplicationContext(), "Auth error: " + e.getMessage(),
            Toast.LENGTH_LONG).show();
    }

    @Override
    public void onCancel() {
        Toast.makeText(getApplicationContext(), "Auth cancel", Toast.LENGTH_LONG).show();
    }

    @Override
    public void onWeiboException(WeiboException e) {
        Toast.makeText(getApplicationContext(), "Auth exception: " + e.getMessage(),
            Toast.LENGTH_LONG).show();
    }
}
```

第四步：

调用 authorize 方法，弹出授权对话框,进行授权。授权成功后即可获得 accesstoken。

```
mWeibo.authorize(MainActivity.this, new AuthDialogListener());
```



四、 集成步骤及示例分析（分享微博）

1. 初始化 SDK

首先,从新浪微博官方申请正式的 appKey,只有真实的 appkey 才能完成微博分享功能,如下图。

```
public interface Constants {  
    //应用的key 请到官方申请正式的appkey替换APP_KEY  
    public static final String APP_KEY="3583664417";  
}
```

初始化接口可以放在 Activity 的 onCreate 里,此初始化为采用单例模式(即使在几个 Activity 里都用到 SDK 初始化,也只会内存创建一份)。

```
// 新浪微博分享的开放接口
IWeiboAPI weiboAPI;
private void initWeiboSDK() {
    // 初始化SDK
    weiboAPI = WeiboSDK.createWeiboAPI(this, Constants.APP_KEY);
}
}
```

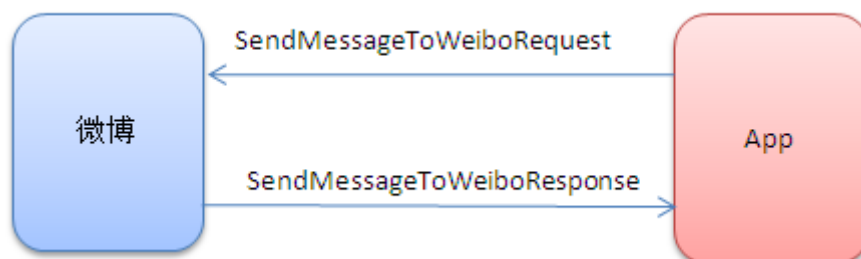
2. 注册到新浪微博

初始化后，调用 registerApp 即可注册到新浪微博。

```
private void regWeibo() {
    //注册到新浪微博
    weiboAPI.registerApp();
}
}
```

3. 发送请求消息给微博

发送请求消息给微博的消息模型如下图所示，是由三方应用发起，分享到微博，然后微博把处理结果返回给三方应用。



第三方请求的代码如下，以发送文本消息为例：

```

/**
 * 三方到微博文本消息
 */
private void reqTextMsg() {
    // 初始化微博的分享消息
    WeiboMessage weiboMessage = new WeiboMessage();
    // 放文本消息
    weiboMessage.mediaObject = getTextObj();
    // 初始化从三方到微博的消息请求
    SendMessageToWeiboRequest req = new SendMessageToWeiboRequest();
    req.transaction = String.valueOf(System.currentTimeMillis()); // 用transaction唯一标识一个请求
    req.message = weiboMessage;
    // 发送请求消息到微博
    weiboAPI.sendRequest(this, req);
}
/**
 * 文本消息构造方法
 */
@return
*/
private TextObject getTextObj() {
    TextObject textObject = new TextObject();
    textObject.text = title.getText().toString();
    return textObject;
}

```

微博响应三方的请求，需要配置条件：

在 AndroidManifest.xml 的需要接收消息的 Activity 里声明

```

<intent-filter>
    <action android:name="com.sina.weibo.sdk.action.ACTION_SDK_REQ_ACTIVITY" />

    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>

```

在 Activity 类需要实现 IWeiboHandler.Response

```

public class RequestMessageActivity extends Activity implements
    OnClickListener, IWeiboHandler.Response {

```

在 Activity 的 onCreate 与 onNewIntent 里加入

```
weiboAPI.responseListener(getIntent(), this);
```

```

@Override
protected void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
    weiboAPI.responseListener(getIntent(), this);
}

```

即可接收 onResponse 消息，如下图所示。


```

@Override
public void onResponse(BaseResponse baseResp) {
    switch(baseResp.errCode){
        case com.sina.weibo.sdk.constant.Constants.ErrorCode.ERR_OK:
            Toast.makeText(this, "成功!!", Toast.LENGTH_LONG).show();
            break;
        case com.sina.weibo.sdk.constant.Constants.ErrorCode.ERR_CANCEL:
            Toast.makeText(this, "用户取消!!", Toast.LENGTH_LONG).show();
            break;
        case com.sina.weibo.sdk.constant.Constants.ErrorCode.ERR_FAIL:
            Toast.makeText(this, baseResp.errMsg + ":失败!!", Toast.LENGTH_LONG).show();
            break;
    }
}

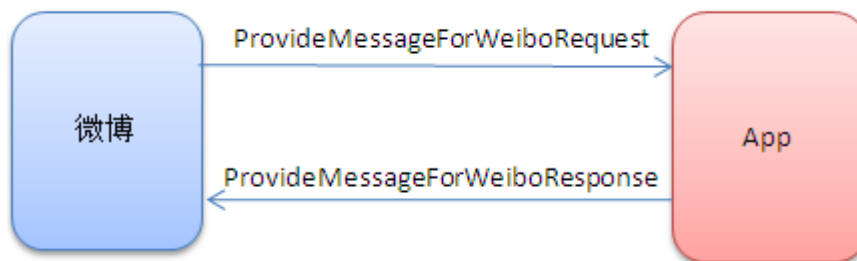
```

注意：

weiboAPI.sendRequest(this, req);接口会检查用户系统是否安装微博，如果没安装，会提示下载安装，更多请查看微博 SDK 开发手册。

4. 接收微博请求消息

发送响应消息给微博的消息模型如下图所示，是由微博应用发起，经过三方，分享到微博。



第三方接收微博的请求（需要配置条件与微博响应三方的请求相同），需要注意的是 mBundle 必须带到新 Activity。

```

@Override
public void onRequest(BaseRequest baseReq) {
    //可以启动一个新activity,根据实际情况自己处理
    Intent intent = new Intent(this, ResponseMessageActivity.class);
    intent.putExtras(mBundle); //mBundle必须带到新Activity
    startActivity(intent);
}

```

响应微博的消息代码，以发送网页为例：

```

private void respWebpageMsg() {
    // 初始化微博的分享消息
    WeiboMessage weiboMessage = new WeiboMessage();
    // 多媒体（网页）消息
    weiboMessage.mediaObject = getWebpageObj();

    // 初始化从三方到微博的消息请求
    ProvideMessageForWeiboResponse resp = new ProvideMessageForWeiboResponse();
    resp.transaction = new ProvideMessageForWeiboRequest(mBundle).transaction;
    Log.i("msg", resp.transaction);
    resp.message = weiboMessage;
    // 发送请求消息到微博
    weiboAPI.sendResponse(resp);
}

/**
 * 多媒体（网页）消息构造方法
 *
 * @return 多媒体（网页）消息对象
 */
private WebpageObject getWebpageObj() {
    WebpageObject mediaObject = new WebpageObject();
    mediaObject.identify = Util.generateId(); // 创建一个唯一的ID
    mediaObject.title = webpageTitle.getText().toString();
    mediaObject.description = webpageContent.getText().toString();
    // 设置bitmap类型的图片到视频对象里
    BitmapDrawable bitmapDrawable = (BitmapDrawable) webpageImage
        .getDrawable();
    mediaObject.setThumbImage(bitmapDrawable.getBitmap());
    mediaObject.actionUrl = webpageUrl.getText().toString();
    return mediaObject;
}

```

此处分成两个方法，respWebpageMsg() 创建响应消息，getWebpageObj() 是获取网页消息对象。

