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 V2.3.0

修订记录

时间	文档版本	修订人	备注
2012/7/20	2.0.0		初稿
2012/8/02	2.0.0		
2013/4/17	2.1.0		新增分享微博
2013/9/06	2.3.0		新增登入登出按钮、好友邀请接口

目录

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。

3. 登入登出按钮

微博登入按钮主要是简化用户进行 SSO 登陆，实际上，它内部是对 SSO 认证流程进行了简单的封装。

微博登出按钮主要一键登出的功能，帮助开发者主动取消用户的授权。

4. 好友邀请接口

好友邀请接口，支持登录用户向自己的微博互粉好友发送私信邀请、礼物。

三、 集成步骤及示例分析（认证授权）

1. 新建工程，导入 SDK JAR 包

添加 JAR 包：

1. 将 SDK 的 JAR 包放到工程的 libs 目录下
2. 添加 JAR 包：工程->右键->properties->java build path -> libraries -> add external jar

DEMO 工程如下图：

2. 创建微博 API 接口类对象

在 MainActivity 中，创建微博 API 接口类对象 mWeibo。

```
mWeibo = Weibo.getInstance(ConstantS.APP_KEY, ConstantS.REDIRECT_URL, ConstantS.
```

其构造函数需要设置 appkey 和应用回调页，以及 scope 权限。在 DEMO 中，我们把这些有用的常量封装到了 ConstantS 类中，如下图：

```
3  public interface ConstantS {
4
5      // 应用的key 请到官方申请正式的appkey替换APP_KEY
6      public static final String APP_KEY      = "2045436852";
7
8      // 替换为开发者REDIRECT_URL
9      public static final String REDIRECT_URL = "http://www.sina.com";
10
11     // 新支持scope：支持传入多个scope权限，用逗号分隔
12     public static final String SCOPE =
13             "email,direct_messages_read,direct_messages_write,"
14             + "friendships_groups_read,friendships_groups_write,statuses_to_me_read,"
15             + "follow_app_official_microblog," + "invitation_write";
```

对于移动客户端应用来说，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。

3. 注册应用程序的包名和签名

应用程序包名：向开放平台注册应用程序的包名；

应用程序签名：向开放平台注册应用程序的签名（该签名是通过官方提供的签名工具生成的 MD5 值）

下图即为注册应用程序的包名和签名的页面，可以在新浪微博开放平台->我的应用->应用信息->应用基本信息处找到，点击编辑按钮即可注册。

如何使用签名工具获取 Android 签名的 MD5 值？

1. 签名工具地址：https://github.com/mobileresearch/weibo_android_sdk/blob/master/app_signatures.apk
2. 使用方式：安装该工具后，输入应用程序的包名，点击生成按钮，即可获得 MD5 签名，如下图所示。（*请注意：要签名的第三方应用程序必须安装在该设备上才能够生成对应的 MD5 签名*）

4. 实现 WeiboAuthListener 接口

授权成功后可在 onComplete 函数中获得 access_token、expires_in 信息。具体如何保存和使用 access_token 信息由开发者自行处理，目前 Demo 中是将其保存到 SharedPreferences 中（请查看 AccessTokenKeeper 类）

```
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: \r\n access_token: "+ token + "\r\n" + "expires_in: " + expires_in+"\r\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();
    }

}
```

5. 调用 authorize 方法，认证并授权

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

```
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，我们把这些有用的常量封装到了 ConstantS 类中，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      = "2045436852";
```

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，也只会在内存创建一份）。

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

}
```

2. 注册到新浪微博

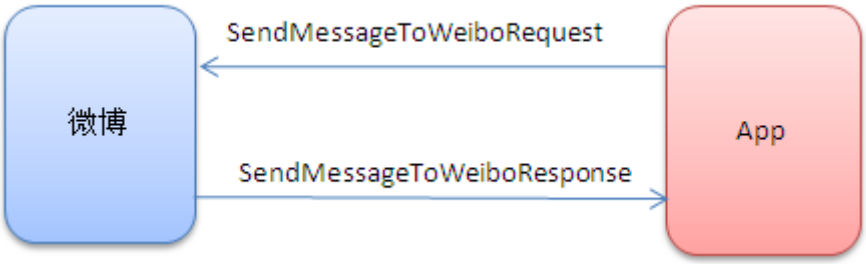
初始化后，调用 registerApp 即可注册到新浪微博。

```
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 中，在需要接收消息的 Activity（唤起微博主程序的类）里声明对应的 Action：*com.sina.weibo.sdk.action.ACTION_SDK_REQ_ACTIVITY*，如下图：

```
<activity
    android:name="com.weibo.sdk.android.demo.RequestMessageActivity"
    android:configChanges="keyboardHidden|orientation"
    android:screenOrientation="portrait">
    <intent-filter>
        <action android:name="com.sina.weibo.sdk.action.ACTION_SDK_REQ_ACTIVITY" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

对应的 Activity 类需要实现以下内容：

1. 实现 IWeiboHandler.Response 接口

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

2. 在 onCreate 与 onNewIntent 里加入 mWeiboAPI.responseListener(...)

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

当从微博发博器界面返回到该 Activity 时，接口函数 IWeiboHandler.Response.onResponse(...)会被调用，用户可以从该函数内获取成功或失败，以及取消的信息，如下图所示。

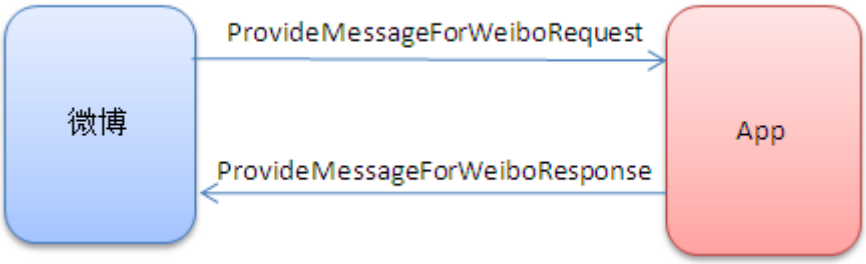
```
/**
 * 从本应用->微博->本应用
 * 接收响应数据，该方法被调用。
 * 注意：确保{@link #onCreate(Bundle)} 与{@link #onNewIntent(Intent)}中，
 * 调用 mWeiboAPI.responseListener(intent, this)
 */
@Override
public void onResponse(BaseResponse baseResp) {
    switch (baseResp.errCode) {
    case com.sina.weibo.sdk.constant.Constants.ErrorCode.ERR_OK:
        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(Activity act, BaseRequest 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。

五、 集成步骤及示例分析（登入登出按钮）

1. 登入按钮集成

目前登入按钮只是提供 SSO 授权的封装，并没有提供现成的样式，所以第三方在集成的时候，可以自定义按钮的样式。

1. 在需要集成的 Activity 的布局文件中，添加按钮：

```
<com.weibo.sdk.view.LoginButton
    android:id="@+id/login_bt"
    android:layout_width="fill_parent"
    android:layout_height="40dp"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:layout_marginTop="10dp"
    android:background="@drawable/login_button"
    android:text="登入"/>
```

2. 在对应的 Activity 中，为该控件设置 Activity 对象：

```
mLoginBt = (LoginButton) findViewById(R.id.login_bt);
mLoginBt.setCurrentActivity(this);
```

3. 当用户点击该按钮时，会进行 SSO 登陆，登陆完成后，返回该 Activity，此时，我们需要在 onActivityResult(…)中调用 mSsoHandler.authorizeCallBack(requestCode, resultCode, data)函数，整个登陆才能结束。当调用完该函数后，我们可以通过 LoginButton 实例获取 access_token，默认情况下，我们已通过 AccessTokenKeeper 将其保存到 SharedPreference 里面。

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // TODO Auto-generated method stub
    super.onActivityResult(requestCode, resultCode, data);
    SsoHandler mSsoHandler = mLoginBt.getSsoHandler();
    if (mSsoHandler != null) {
        mSsoHandler.authorizeCallBack(requestCode, resultCode, data);

        TextView tv = (TextView) findViewById(R.id.result);
        Oauth2AccessToken token = AccessTokenKeeper.readAccessToken(getApplicationContext());
        if (token != null && !TextUtils.isEmpty(token.getToken())) {
            tv.setText(token.getToken());
        }
    }
}
```

2. 登出按钮集成

登出按钮的集成比较简单，由于我们已通过 AccessTokenKeeper 将其保存到 SharedPreference 里面，因此，LogoutButton 内部，我们先从 SharedPreference 获取 access_token，然后进行登出操作。登出操作只需要在 Activity 的布局文件中，添加按钮即可，如下图：

```
<com.weibo.sdk.view.LogoutButton
    android:id="@+id/logout_bt"
    android:layout_width="fill_parent"
    android:layout_height="40dp"
    android:layout_marginLeft="20dp"
    android:layout_marginRight="20dp"
    android:layout_marginTop="20dp"
    android:background="@drawable/login_button"
    android:text="登出"/>
```

六、 集成步骤及示例分析（好友邀请）

1. 好友邀请接口集成

该接口支持登录用户向自己的微博互粉好友发送私信邀请、礼物。
该接口的详细内容可参见：http://open.weibo.com/wiki/2/messages/invite

集成该接口步骤也是非常简单，只需要创建一个 InviteApi 类的实例，并调用其 sendInvite() 方法即可，如下图：

```
mInviteButton = (Button) findViewById(R.id.invite);
mInviteButton.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        Bundle bundle = new Bundle();
        bundle.putString(InviteApi.KEY_TEXT, "songxiaoxue 邀请接口");
        bundle.putString(InviteApi.KEY_URL, "http://weibo.com/u/1846671692?wvr=5&");
        String uid = "3769295315";//"2371687607";
        new InviteApi(getApplicationContext(), bundle, uid).sendInvite();
    }
});
```

在初始化 InviteApi 类的实例时，需要设置以下内容：

1. InviteApi.KEY_TEXT：要回复的私信文本内容。文本大小必须小于 300 个汉字。
2. InviteApi.KEY_URL：邀请点击后跳转链接。默认为当前应用地址。

注意事项：

1. 应用引导用户使用微博登录，采用 OAuth2 的 Scope 授权方式获得用户“发送私信邀请给你的微博好友”权限，对应的 scope 权限参数为：invitation_write；
2. 每用户每小时通过每应用可邀请 10 个互粉好友；
3. 必须由用户主动发起。

具体请参考 Scope 权限说明：http://open.weibo.com/wiki/Scope