

微博 Android 平台 SDK 文档

编号：WEIBO_ANDROID_SDK

版本：WEIBO_ANDROID_SDK V3.1.1

修订记录：

| 时间 | 文档版本 | 修订人 | 备注 |
|------------|-------|-----|---|
| 2012/7/20 | 2.0.0 | | 初稿 |
| 2012/8/2 | 2.0.0 | | |
| 2013/4/17 | 2.1.0 | | 新增分享微博 |
| 2013/9/12 | 2.3.0 | | 新增登入登出按钮、好友邀请接口 |
| 2013/11/11 | 2.4.0 | | 重大版本变更： <ul style="list-style-type: none">➤ 授权、分享等代码的重构➤ 代码、文档等规范化➤ 开源 OpenAPI 部分代码 |
| 2014/3/7 | 2.5.0 | | <ul style="list-style-type: none">➤ 重构网络模块代码➤ 提供同步和异步的网络请求接口➤ 提供网络模块常用接口，如获取用户信息➤ 提供网络请求后的数据结构，如 User、微博信息流等数据结构➤ 修正若干 BUG➤ 简化文档 |
| 2014/11/19 | 3.0.0 | | <ul style="list-style-type: none">➤ 优化网页授权➤ 优化网页分享➤ 增加社会化评论组件➤ 增加社会化关注组件 |
| 2015/6/1 | 3.1.1 | | <ul style="list-style-type: none">➤ 增加手机短信注册入口 修改文档说明➤ 增加微博支付接口调用说明➤ |

目录

| | |
|---------------------------------|---|
| 目录..... | 2 |
| 1. 概述及名词解释..... | 1 |
| 1.1 认证授权..... | 1 |
| 1.2 名词解释..... | 1 |
| 2. 功能列表..... | 1 |
| 2.1 认证授权..... | 1 |
| 2.2 微博分享..... | 2 |
| 2.3 登录/注销按钮..... | 2 |
| 2.4 开放接口..... | 2 |
| 3. 运行示例代码..... | 3 |
| 3.1 导入工程..... | 3 |
| 3.2 修改 debug.keystore..... | 3 |
| 3.3 编译运行..... | 4 |
| 4. 微博 SDK 及 DEMO 工程目录结构及分析..... | 5 |
| 4.1 闭源部分结构分析..... | 5 |
| 4.2 开源部分结构分析..... | 6 |
| 4.3 Demo 部分结构分析..... | 6 |
| 5. 集成前准备..... | 7 |
| 5.1 申请应用程序的 APP_KEY..... | 7 |
| 5.2 注册应用程序的包名和签名..... | 7 |
| 5.3 选择应用的集成方式..... | 8 |

| | |
|---|----|
| 5.4 在应用中添加 SDK 所需要的权限..... | 8 |
| 6. 授权分享等示例代码分析 | 9 |
| 6.1 认证授权 | 9 |
| 6.2 分享微博 | 12 |
| 6.3 一键登录/注销按钮..... | 16 |
| 7. OpenAPI 示例代码分析 | 19 |
| 7.1 用户信息接口 | 19 |
| 7.2 邀请好友接口..... | 20 |
| 8. 社会化组件 | 21 |
| 8.1 关注组件 (AttentionComponentView) | 21 |
| 8.2 评论组件 (CommentComponentView) | 22 |
| 9. 支付组件 | 23 |
| 9.1 支付示例 | 23 |
| 9.2 支付参数说明 | 24 |
| 9.3 支付接口错误返回码说明..... | 25 |
| 9.4 签名机制 | 26 |
| 10 问题 | 28 |
| 10.1 如何使用签名工具获取应用的签名？ | 28 |
| 10.2 如何实现 LinkCard 效果？ | 28 |
| 10.3 如何实现附件栏集成分享？ | 28 |
| 10.4 第三方如何申请接口权限？ | 29 |

1. 概述及名词解释

1.1 认证授权

新浪微博 Android SDK 为开发者提供了 OAuth2.0 授权认证，并集成 SSO 登录功能，使第三方应用无需了解复杂的验证机制即可进行授权登录操作，并提供微博分享功能，第三方应用可直接通过微博客户端进行分享。

本文档介绍了新浪微博 Android SDK 的三种授权方式，各种分享、获取用户信息等常用接口，并给出简单的示例分析，帮助第三方开发者快速集成应用。

1.2 名词解释

| 名词 | 注解 |
|------------------------|--|
| 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。 |
| RedirectURI | 第三方应用授权回调页面。 授权回调页对移动客户端应用来说对用户是不可见的，所以定义为何种形式都将不影响，但是没有定义将无法使用 SDK 认证登录。建议使用默认回调页 https://api.weibo.com/oauth2/default.html 可以在“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。 |
| Scope | Scope 是 OAuth2.0 新版授权页的一个功能，通过 scope 平台将开放更多的微博核心功能给开发者，同时也加强用户隐私保护，提升用户体验，用户在新 OAuth2.0 授权页中有权利选择赋予应用的功能。 |
| AccessToken | 表示用户身份的 Token，用于微博 OpenAPI 的调用。 |
| OAuth2.0 Web 授权 | 通过 WebView 进行授权，并返回 Token 信息。 |
| SSO 授权 | 通过唤起微博客户端进行授权，并返回 Token 信息。 |
| Code 授权 | 通过应用的 APP_ID 和 APP_KEY 来获取 Token，不需要应用的包名和签名 |

2. 功能列表

2.1 认证授权

目前微博 SDK 为开发者提供三种授权方式：

- SSO 授权：仅当手机安装新浪微博。客户端时使用 SSO 授权登陆，实现见 6.1.1
- Web 授权：在没有客户端的情况下，可直接使用该授权，实现见 6.1.2
- SSO+Web 授权：如果手机端安装了新浪微博客户端的话会默认发起 SSO 授权，反之则进行 Web 授权（推荐使用），实现见 6.1.3

2.2 微博分享

通过微博 SDK，第三方应用能够分享文字、图片、视频、音乐等内容，目前分享有三种方式：

➤ 有微博客户端情况

1. 通过第三方应用唤起微博客户端进行分享（该分享方式为第三方客户端通常的使用方式），[实现见 6.2.1](#)
2. 通过微博客户端唤起第三方应用进行分享（该分享方式需要合作接入，参考 <http://t.cn/aex4JF>），[实现见 6.2.2](#)

➤ 无微博客户端情况，[实现见 6.2.3](#)

3. 通过 OpenAPI 进行分享，直接使用 StatusesAPI 中的 upload、update 或 uploadUrlText 函数进行分享，或直接使用 AsyncWeiboRunner#requestAsync 方法，自己进行拼接参数进行 HTTP 请求实现分享。

2.3 登录/注销按钮

微博 SDK 目前提供了两类登录按钮：一种是一键登陆按钮，一种是登陆/注销按钮，两者都是调用 SSO 登录接口。

2.4 开放接口

微博 SDK 目前提供了一个 OpenAPI 接口调用框架，并封装了一些简单的开放接口，以供大家参考：

| | | |
|--------------------------|-------|---------------------------------|
| LogoutAPI | 注销接口 | 授权回收接口，取消用户的授权 |
| InviteAPI | 邀请接口 | 好友邀请接口，向自己的微博互粉好友发送私信邀请、礼物 |
| StatusesAPI | 微博接口 | 获取微博信息、删除微博、发送微博、获取微博官方表情等接口 |
| FavoritesAPI | 收藏接口 | 获取收藏列表、收藏信息、收藏标签，增删改收藏列表等接口 |
| CommentsAPI | 评论接口 | 获取评论列表、评论信息，删除、回复评论等接口 |
| FriendshipsAPI | 关系接口 | 获取关注列表、关注用户 Id、粉丝列表，关注或取消关注等接口 |
| GroupAPI | 分组接口 | 获取好友分组列表、好友分组的微博列表，增删改好友分组等接口 |
| LocationAPI | 地理信息 | 获取坐标、返回实际位置信息等接口 |
| PlaceAPI | 位置服务 | 获取用户、用户与好友、某个位置点、周边的位置动态等接口 |
| ShortUrlAPI | 微博短链接 | 转换长短链接，获取链接点击数、来源、微博内容、评论数等接口 |
| AccountAPI | 账号接口 | 获取登录用户的信息、API 访问频率限制、学校列表、退出等接口 |
| ActivityInvokeAPI | 微博相关页 | 调起微博客户端发送微博、查看周边人、打开个人资料等页面接口 |
| CommonAPI | 公共服务 | 获取城市列表、国家列表、时区配置表等接口 |
| RegisterAPI | 注册接口 | 根据用户填写的信息验证用户填写的昵称是否可用接口 |
| SearchAPI | 搜索接口 | 搜索用户、微博、学校、公司、应用时的联想搜索建议等接口 |
| SuggestionsAPI | 推荐接口 | 获取热门用户列表、感兴趣的人、推荐相关微博、精品推荐等接口 |
| TagsAPI | 标签接口 | 获取用户的标签列表、创建或删除标签等接口 |
| TrendsAPI | 话题接口 | 获取话题列表、热门话题、关注或取消关注某话题等接口 |
| UsersAPI | 用户接口 | 获取用户信息、用户最新的一条微博信息等接口 |

3. 运行示例代码

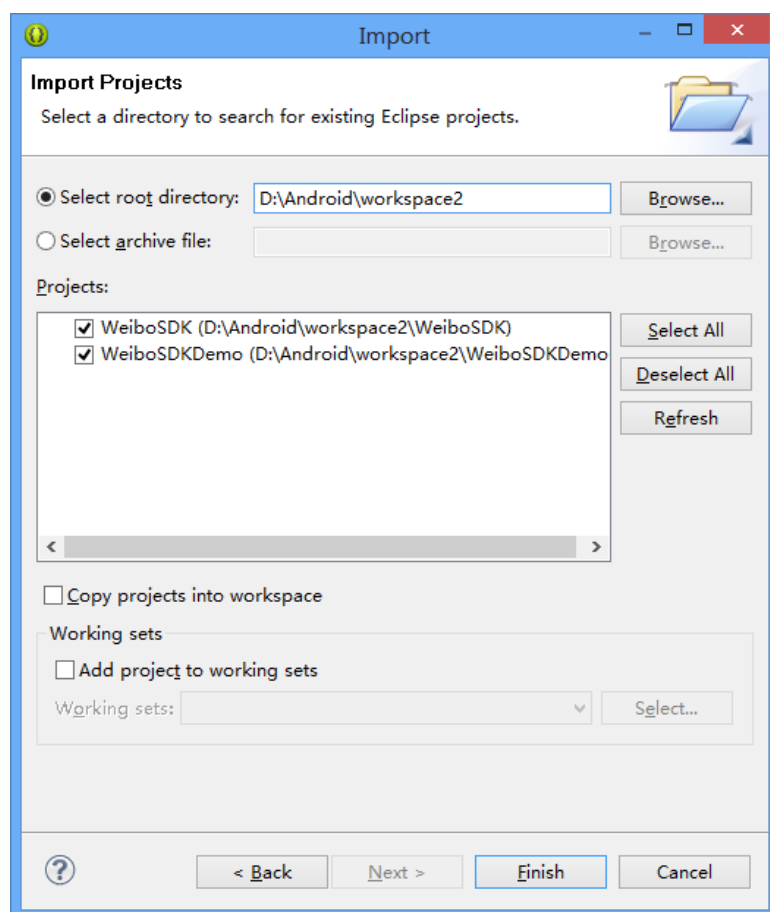
为了方便第三方应用更快的集成微博 SDK，更清晰的了解目前微博 SDK 所提供的功能，可在 GitHub（https://github.com/mobilesearch/weibo_android_sdk）下载整个示例工程以及对应的 APK 安装包。运行工程可以通过以下两种方式进行运行：

- 直接安装 WeiboSDKDemo.apk 至手机进行运行
- 在 Eclipse 中导入并运行 WeiboSDKDemo 工程

我们简要描述一下运行 Demo 工程的步骤：

3.1 导入工程

在 Eclipse 中，点击 File→Import→Existing Projects into Workspace，输入正确的路径，导入工程，如下图：



注意：目前整个工程全采用中文注释，为了防止乱码滋生，请修改文本编码方式为 UTF-8。

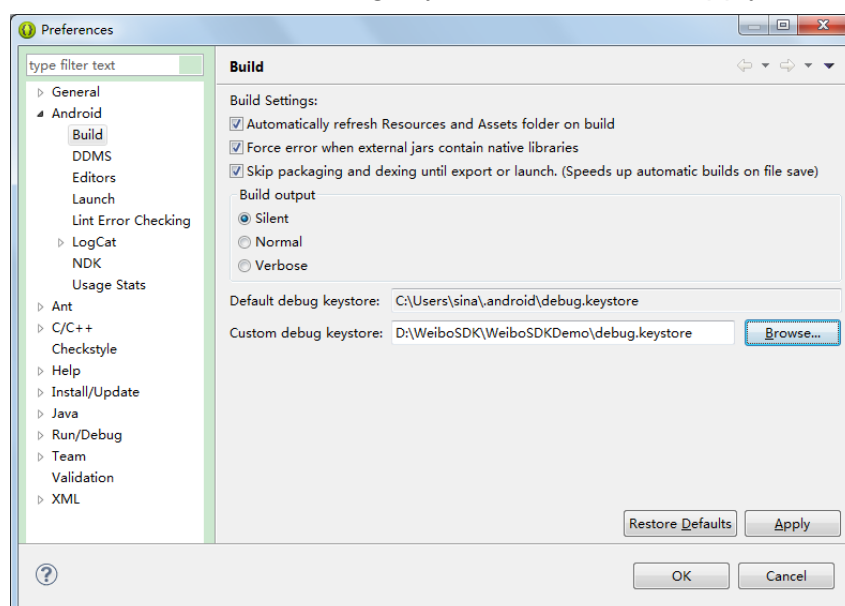
更改方式：Eclipse→Window→General→Workspace→Text file encoding。

3.2 修改 debug.keystore

MD5 工具是根据 keystore 来生成签名的，不同的 keystore 生成的签名是不一样的。此 Demo 的签名是用官网提供的 keystore 生成的，若要顺利运行 Demo 程序，需要进行设置或是替换 keystore，两种方法选择一种操作即可：

方式一：替换 keystore：把 Android 默认的 debug.keystore（在 C:\Users\XXXXX\android 目录下）替换成官方在 GitHub 上提供的 debug.keystore。

方式二：在 Eclipse 中设置工程 keystore：在 Eclipse 中点击 “Windows→Preferences→Android→Build”，在 Custom debug keystore 中选择 Demo 中的 debug.keystore，如下图，点击 Apply→OK，Demo 即可正常运行。



注意：这一步是必须的，如果没有替换，demo 程序在运行时将无法正确的授权成功。用户在替换前，最好先备份一下原始的 debug.keystore。GitHub 中 debug.keystore 是新浪官方的，除了编译运行官方 DEMO 外，不要直接使用它，出于安全的考虑，用户应该为自己的应用提供一份 keystore。

3.3 编译运行

在编译前，先 Clean 一下工程，然后右键点击工程→Run As→Android Application，即可运行。默认的编译 target 是 Android 2.3.3，第三方可以自行修改成合适的版本。

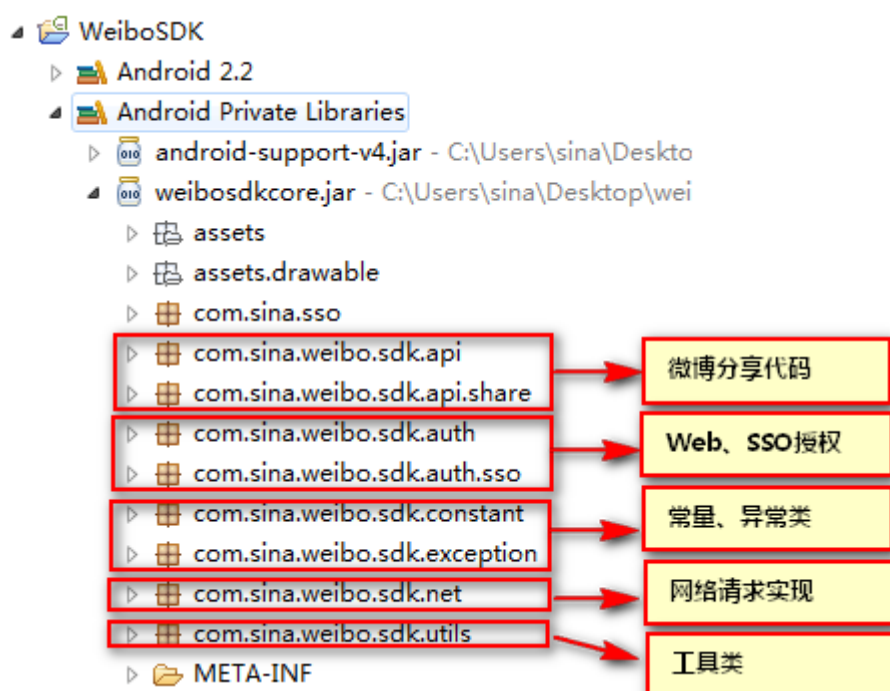
4. 微博 SDK 及 DEMO 工程目录结构及分析

微博 SDK 目前以是**部分开源**的形式提供给第三方开发者的，简单来说，可以分为以下三部分：

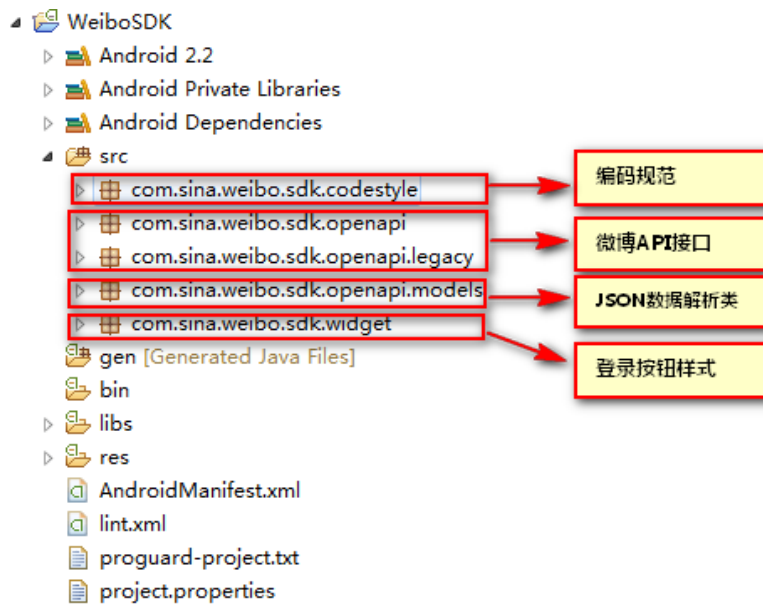
- **闭源部分**：[weibosdkcore.jar](#)，该 JAR 包集成了微博授权、SSO 登录以及分享等核心功能。另外，在 V2.5.0 中，我们将网络模块框架也加入其中，方便开发者进行 OpenAPI 的网络请求。
- **开源部分**：[WeiboSDK 工程 \(Library \)](#)，该工程引用了 [weibosdkcore.jar](#)，这里面主要是对 OpenAPI 进行了简单的封装，第三方可以参考使用流程，模仿并添加自己需要的接口，利用 OpenAPI 接口获取用户信息，分享微博等。
- **Demo 部分**：[WeiboSDKDemo 工程](#)，该工程引用了 [WeiboSDK 工程](#)，提供了目前微博所支持的部分功能的示例代码。

注意：第三方在使用时，如果只需要实现授权和分享功能，可直接使用 [weibosdkcore.jar](#)；如果想使用其它功能，可直接导入 [WeiboSDK 工程](#)。如何导入 [WeiboSDK 工程](#)请详见：[集成步骤及示例分析](#)

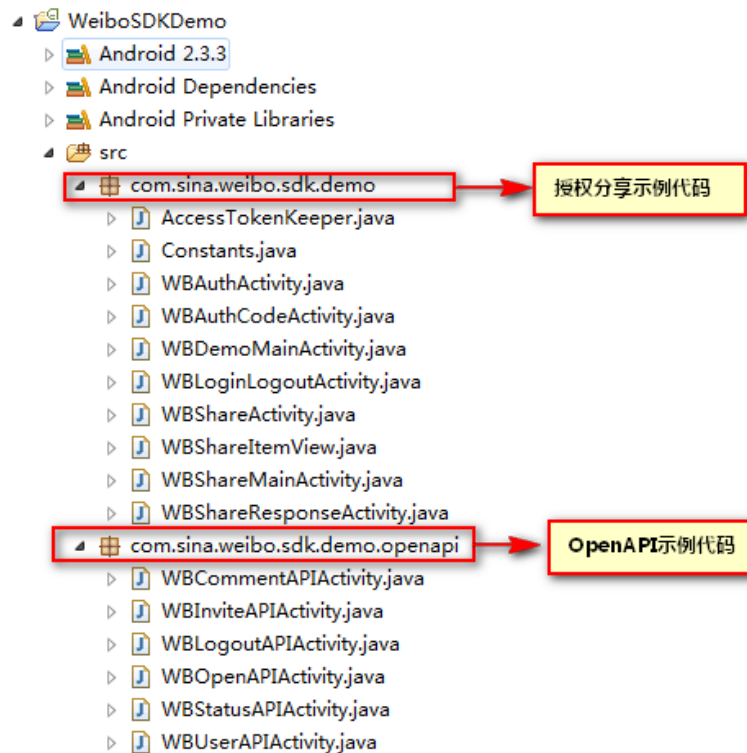
4.1 闭源部分结构分析



4.2 开源部分结构分析



4.3 Demo 部分结构分析



5. 集成前准备

5.1 申请应用程序的 APP_KEY

用户首先需要在微博开放平台上对应用进行注册，并获取 APP_KEY，添加应用的授权回调页（Redirect URI）。详情请仔细阅读：[移动客户端接入](http://t.cn/aex4JF)（<http://t.cn/aex4JF>）

5.2 注册应用程序的包名和签名

对应用授权前，需要在微博开放平台上注册应用程序的包名和签名后。

注意：包名和签名未注册，或者签名注册不正确，都会导致无法授权。
手机时间不对也会造成授权不成功。

- 应用程序包名：指 AndroidManifest.xml 文件中，package 标签所代表的内容。
- 应用程序签名：该签名是通过官方提供的签名工具生成的 MD5 值（[获取签名方式见附录 8.1](#)）

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。

应用基本信息

应用类型：

普通应用 - 客户端

应用名称：

我的安卓客户端应用

该名称也用于来源显示，不超过10个汉字或20个字母

应用平台：

手机

[查看移动客户端接入指南](#)

☐ iPhone ☒ Android ☐ BlackBerry

☐ Windows Phone ☐ Symbian

☐ WebOS ☐ Other

1

* Android包名：

com

2

* Android签名：

702465el

通过下载使用平台提供的[签名工具](#)获取

Debug签名：

用户根据需要填写，可以不填。

3

Android下载地址：

http://

5.3 选择应用的集成方式

在集成微博 SDK 前，有两种方式来集成微博 SDK：

- 直接导入 weibosdkcore.jar：*适用于只需要授权、分享、网络请求框架功能的项目*
- 引用 WeiboSDK 工程 (Library)：*适用于微博授权、分享，以及需要登陆按钮、调用 OpenAPI 的项目*
-

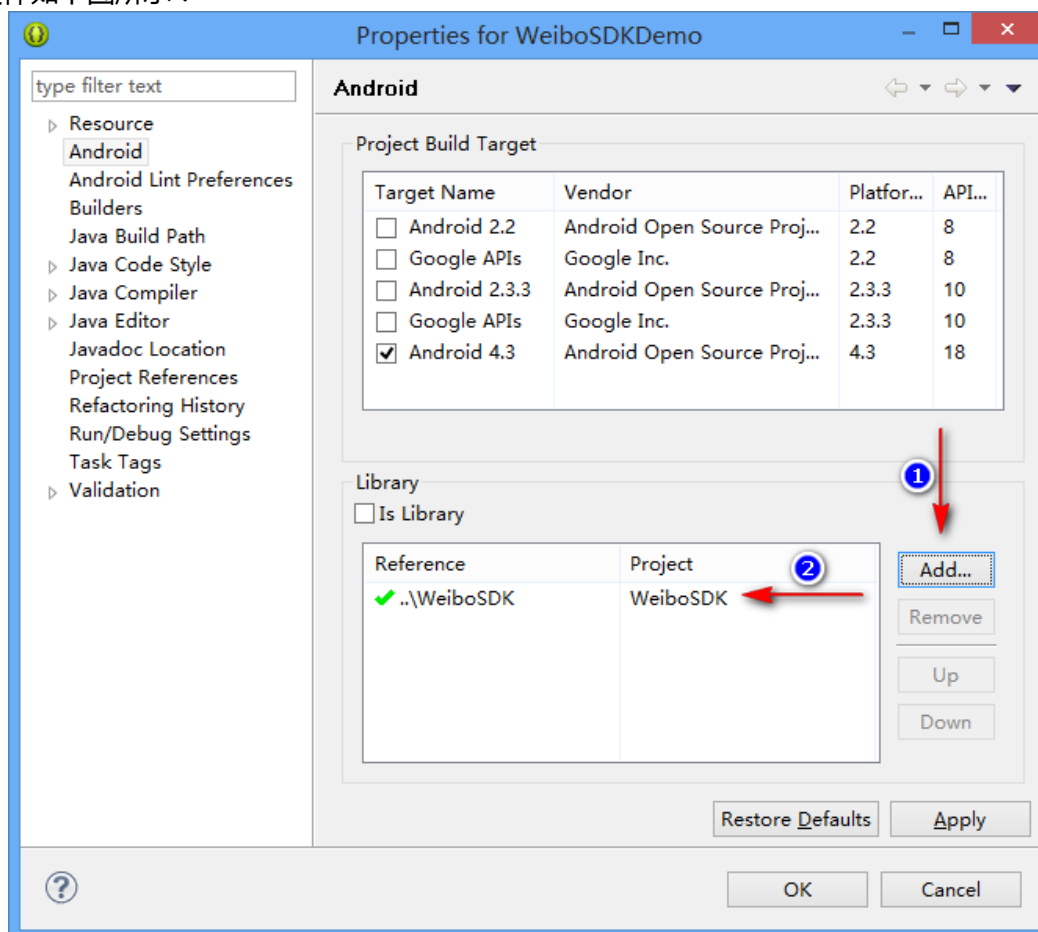
下面简述下两种导入方式的实现：

方式一：直接导入 weibosdkcore.jar

- ☞ 将官方的提供 SDK JAR 包 (weibosdkcore.jar) 放到工程的 libs 目录下
- ☞ 添加 JAR 包：工程→右键→properties→java build path→libraries→add external jar

方式二：引用 WeiboSDK 工程 (Library)

- ☞ 将 WeiboSDK 工程整个目录拷贝到你自己的工程相同的目录下
- ☞ 在你自己的工程中,添加 WeiboSDK 工程的引用：工程→右键→properties→Android→Add→选择工程，具体如下图所示：



注意：无论使用哪一种方式，都需要先将 demo 中 lib 目录下的对应三个 so 文件目录 拷贝到你的目标工程中
Demo 工程中的 weibosdkcore.jar 与 github 上的 weibosdkcore_v3.x.x.jar 包是同一个。

5.4 在应用中添加 SDK 所需要的权限

打开 AndroidManifest.xml 文件，将 SDK 需要的权限添加到该文件中：

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

6. 授权分享等示例代码分析

6.1 认证授权

目前微博 SDK 提供四种授权方式，这四种授权方式都需要先实现以下三步：

1) 替换成自己应用的 APP_KEY 等参数

若您使用的是 Demo 工程，需要将工程中 Constants 类的 APP_KEY、Redirect_URL、SCOPE 域替换成自己应用对应的参数，如需要实现好友邀请接口，需要在 SCOPE 参数中添加"invitation_write"值。

```
public interface Constants {
    public static final String APP_KEY = "2045436852"; // 应用的APP_KEY
    public static final String REDIRECT_URL = "http://www.sina.com"; // 应用的回调页
    public static final String SCOPE = // 应用申请的高级权限
        "email,direct_messages_read,direct_messages_write,"
        + "friendships_groups_read,friendships_groups_write,statuses_to_me_read,"
        + "follow_app_official_microblog," + "invitation_write";
}
```

2) 创建微博授权类对象

在 WBAuthActivity 类中，创建微博授权类对象，将应用的信息保存：

```
mAuthInfo = new AuthInfo(this, Constants.APP_KEY,
                          Constants.REDIRECT_URL, Constants.SCOPE);
```

3) 实现 WeiboAuthListener 接口

授权成功后，SDK 会将 access_token、expires_in、uid 等通过 Bundle 的形式返回，在 onComplete 函数中，可以获取该信息。具体如何保存和 Token 信息由开发者自行处理。

注意：当您注册的应用程序签名不正确时，SDK 会将 Code 返还给用户，请确保应用的签名是正确的。

```
class AuthListener implements WeiboAuthListener {
    @Override
    public void onComplete(Bundle values) {
        mAccessToken = OAuth2AccessToken.parseAccessToken(values); // 从 Bundle 中解析 Token
        if (mAccessToken.isSessionValid()) {
            AccessTokenKeeper.writeAccessToken(WBAuthActivity.this, mAccessToken); //保存Token
            .....
        } else {
            // 当您注册的应用程序签名不正确时，就会收到错误Code，请确保签名正确
            String code = values.getString("code", "");
            .....
        }
    }
    .....
}
```

6.1.1 仅通过微博客户端 SSO 登录方式授权

```
mSsoHandler = new SsoHandler(WBAuthActivity.this, mWeiboAuth);
```

- 1) 创建 SsoHandler 对象
- 2) 调用 SsoHandler# authorizeClientSso 方法

```
mSsoHandler.authorizeClientSso(new AuthListener());
```

- 3) 重写 Activity#onActivityResult 方法，调用 SsoHandler# authorizeCallBack

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (mSsoHandler != null) {
        mSsoHandler.authorizeCallBack(requestCode, resultCode, data);
    }
}
```

6.1.2 仅通过 Web 页面方式授权

- 1) 创建 SsoHandler 对象

```
mSsoHandler = new SsoHandler(WBAuthActivity.this, mWeiboAuth);
```

- 2) 调用 SsoHandler# authorizeWeb 方法

```
mSsoHandler.authorizeWeb(new AuthListener());
```

Web授权需要在AndroidManifest.xml中，注册授权页面

```
<activity
    android:name="com.sina.weibo.sdk.component.WeiboSdkBrowser"
    android:configChanges="keyboardHidden|orientation"
    android:windowSoftInputMode="adjustResize"
    android:exported="false" >
</activity>
```

3) Manifest 中注册 web 页面 (activity)

6.1.3 all In one 方式授权

注：此种授权方式会根据手机是否安装微博客户端来决定使用 sso 授权还是网页授权，如果安装有微博客户端 则调用微博客户端授权，否则调用 Web 页面方式授权

1) 创建 SsoHandler 对象

```
mSsoHandler = new SsoHandler(WBAuthActivity.this, mWeiboAuth);
```

2) 调用 SsoHandler# authorize 方法

```
mSsoHandler.authorize(new AuthListener());
```

3) 重写 Activity#onActivityResult 方法，调用 SsoHandler# authorizeCallBack

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (mSsoHandler != null) {
        mSsoHandler.authorizeCallBack(requestCode, resultCode, data);
    }
}
```

6.1.4 通过手机短信授权登录

注：该方式由新浪提供短信下发通道 供开发者使用。调用流程与 sso 以及 all in one 方式类似，具体调用过程参见调用 demo 示例。

1) 创建 SsoHandler 对象

```
mSsoHandler = new SsoHandler(WBAuthActivity.this, mWeiboAuth);
```

2) 调用 SsoHandler#registerOrLoginByMobile 方法

注：第一个参数是String类型，用来控制“短信授权登录”页面的标题，传空时默认为“验证码登录”

```
mSsoHandler.registerOrLoginByMobile("", new AuthListener());
```

3) 重写 Activity#onActivityResult 方法，调用 SsoHandler# authorizeCallBack

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (mSsoHandler != null) {
        mSsoHandler.authorizeCallBack(requestCode, resultCode, data);
    }
}
```

4) 从 listener 对象中获取用户输入的 手机号码 和验证授权信息

```
class AuthListener implements WeiboAuthListener {
    @Override
    public void onComplete(Bundle values) {
        mAccessToken = OAuth2AccessToken.parseAccessToken(values); // 从 Bundle 中解析 Token
        String phoneNum = mAccessToken.getPhoneNum(); // 用户输入的电话号码
        if (mAccessToken.isSessionValid()) {
            AccessTokenKeeper.writeAccessToken(WBAuthActivity.this, mAccessToken); // 保存
Token
            .....
        } else {
            // 当您注册的应用程序签名不正确时，就会收到错误Code，请确保签名正确
            String code = values.getString("code", "");
            .....
        }
    }
}
.....
}
```

6.2 分享微博

6.2.1 从第三方应用唤起微博客户端进行分享

1) 分享前准备工作

在进行微博分享前，需要在 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 : ACTION_SDK_REQ_ACTIVITY , 如下所示 :

```
<activity
    android:name="com.sina.weibo.sdk.demo.WBShareActivity"
    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>
<activity
    android:name="com.sina.weibo.sdk.component.WeiboSdkBrowser"
    android:configChanges="keyboardHidden|orientation"
    android:windowSoftInputMode="adjustResize"
    android:exported="false" >
</activity>
```

2) 分享实现

① 在 onCreate 函数创建微博分享接口实例, 并进行注册, 请确保先注册, 后分享

```
mWeiboShareAPI = WeiboShareSDK.createWeiboAPI(this, Constants.APP_KEY);
mWeiboShareAPI.registerApp(); // 将应用注册到微博客户端
```

② 创建要分享的内容

```
private TextObject getTextObj() {
    TextObject textObject = new TextObject();
    textObject.text = getSharedText();
    return textObject;
}
```

③ 通过 IWeiboShareAPI#sendRequest 唤起微博客户端发博器进行分享


```
private void sendMultiMessage(boolean hasText, boolean hasImage, boolean hasWebpage,
    boolean hasMusic, boolean hasVideo, boolean hasVoice) {
    WeiboMultiMessage weiboMessage = new WeiboMultiMessage(); //初始化微博的分享消息
    if (hasText) {
        weiboMessage.textObject = getTextObj();
    }
    SendMultiMessageToWeiboRequest request = new SendMultiMessageToWeiboRequest();
    request.transaction = String.valueOf(System.currentTimeMillis());
    request.multiMessage = weiboMessage;
    mWeiboShareAPI.sendRequest(request); //发送请求消息到微博，唤起微博分享界面
}
```

④ 实现 IWeiboHandler#Response 接口，接收分享后微博返回的数据

```
protected void onNewIntent(Intent intent) {
    super.onNewIntent(intent);
    mWeiboShareAPI.handleWeiboResponse(intent, this); //当前应用唤起微博分享后，返回当前应用
}
@Override
public void onResponse(BaseResponse baseResp) { //接收微客户端博请求的数据。
    switch (baseResp.errCode) {
        case WBConstants.ErrorCode.ERR_OK: .....
            break;
        case WBConstants.ErrorCode.ERR_CANCEL: .....
            break;
        case WBConstants.ErrorCode.ERR_FAIL: .....
            break;
    }
}
```

3) 分享成功后效果

微博分享成功后，可以打开微博客户端查看效果，如下图所示：[LinkCard 效果实现请参考附录 8.2。](#)

🔗 分享新闻链接：



📺 分享视频链接：



6.2.2 从微博客户端唤起第三方应用进行分享

通过微博客户端进行分享，即[附件栏集成分享（见附录 8.3）](#)，如果**没有进行过商务合作**，是无法进行该分享的。

1) 分享前准备工作

通过微博唤起第三方应用进行分享前，同样需要在 AndroidManifest.xml 中，在需要接收消息的 Activity（被微博唤起的程序的类）里声明对应的 Action：ACTION_SDK_RESP_ACTIVITY，如下所示：

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

2) 分享的实现步骤

① 在 onCreate 函数创建微博分享接口实例

```
mShareWeiboAPI = WeiboShareSDK.createWeiboAPI(this, Constants.APP_KEY); //创建分享实例
mShareWeiboAPI.handleWeiboRequest(getIntent(), this); // 处理微博客户端发送过来的请求
```

② 实现 IWeiboHandler#Request 接口，接收并处理微博客户端发送过来的请求

```
@Override
protected void onNewIntent(Intent intent) {
    .....
    mShareWeiboAPI.handleWeiboRequest(intent, this); // 处理微博客户端发送过来的请求
}

@Override
public void onRequest(BaseRequest baseRequest) {
    // 保存从微博客户端唤起第三方应用时，客户端发送过来的请求数据对象
    mBaseRequest = baseRequest;
    .....
}
```

③ 创建要分享的内容，实现代码同 6.2.1 中创建分享内容一样。

④ 通过 IWeiboShareAPI#sendResponse 发送应答数据给微博客户端

```
private void responseSingleMessage(boolean hasText, boolean hasImage,
    boolean hasWebpage, boolean hasMusic, boolean hasVideo) {
    WeiboMessage weiboMessage = new WeiboMessage();
    // 1. 初始化微博的分享消息
    if (hasText) {
        weiboMessage.mediaObject = getTextObj();
    }

    .....
    // 2. 初始化从微博到第三方的消息请求
    ProvideMessageForWeiboResponse response = new ProvideMessageForWeiboResponse();
    response.transaction = mBaseRequest.transaction;
    response.reqPackageName = mBaseRequest.packageName;
    response.message = weiboMessage;
    // 3. 发送响应消息到微博
    mShareWeiboAPI.sendResponse(response);
}
```

6.2.3 使用 OpenAPI 进行分享

1) 在 onCreate 函数创建微博分享接口实例

```
statusesAPI = new StatusesAPI(accessToken); //创建微博分享接口实例
```

2) 实现 RequestListener 接口，接收并处理微博分享后结果

3) 调用 StatusAPI#update、update 或 uploadUrlText 函数发送微博

```
statusesAPI.update("要分享的内容", "0.0", "0.0", mListener);
```

注意：如果应用未通过审核，调用 OpenAPI 中的函数需要添加测试账号，在“应用信息->测试帐号”中添加，否则不能调用函数，会报测试账号已超过上限的 ErrorCode。

6.3 一键登录/注销按钮

6.3.1 一键登录按钮

目前提供了以下三种样式，如下图：



其中，第一种为默认样式，对于第三种样式，目前不提供按下的效果。对于一键登录，使用步骤如下：
使用步骤如下：

- 1) 在需要集成的 Activity 的布局文件中，添加按钮：

```
<!-- 默认效果，带图片文字的登陆按钮 -->
<com.sina.weibo.sdk.widget.LoginButton
    android:id="@+id/login_button_default"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/com_sina_weibo_sdk_login_button_with_account_text" />
```

用户可根据需要，将 drawable 替换成其它两种：

```
@drawable/com_sina_weibo_sdk_login_button_with_frame_logo
@drawable/com_sina_weibo_sdk_login_button_with_original_logo
```

- 2) 在对应的 Activity 中，为该控件设置授权认证信息和 listener。

mLoginListener 为 WeiboAuthListener，请参考：[示例分析：授权认证](#)

```
AuthInfo authInfo = new AuthInfo(this, Constants.APP_KEY, Constants.REDIRECT_URL,
    Constants.SCOPE); // 创建授权认证信息
mLoginBtnDefault = (LoginButton) findViewById(R.id.login_button_default);
mLoginBtnDefault.setWeiboAuthInfo(authInfo, mLoginListener); // 为按钮设置授权认证信息
```

- 3) 当用户点击该按钮时，会进行 SSO 登陆，登陆完成后返回应用 Activity，需要在 Activity#onActivityResult 中调用 LoginButton#authorizeCallBack 函数 整个登陆过程结束。授权成功后，用户可选择保存自己的 Token。

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (mLoginBtnDefault != null) {
        mLoginBtnDefault.onActivityResult(requestCode, resultCode, data);
    }
}
```

6.3.2 登录/注销按钮

该按钮目前提供了以下样式，蓝色和银白色，如下图左所示：



其中，第一种为默认样式，第三方开发者可自行修改。登录过程的实现和一键登录一样。使用步骤如下：

1) 在需要集成的 Activity 的布局文件中，添加按钮：

```
<!-- 默认效果，其 style 为: @style/com_sina_weibo_sdk_loginview_default_style -->
<com.sina.weibo.sdk.widget.LoginoutButton
    android:id="@+id/login_out_button_default"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    style="@style/com_sina_weibo_sdk_loginview_default_style" />
```

在对应的 Activity 中，为该控件设置授权认证信息和 listener。与一键登录按钮完全一样，不再赘述。
当用户点击该按钮时，会进行 SSO 登陆，其流程与一键登录按钮完全一样，不再赘述。

2) 用户可根据需要，将为其添加 style 样式，目前有两种选：

```
@style/com_sina_weibo_sdk_loginview_default_style
@style/com_sina_weibo_sdk_loginview_silver_style
```

注意：该 style 文件定义在 WeiboSDK/res/values/styles.xml 下，第三方可根据需要修改或重写。

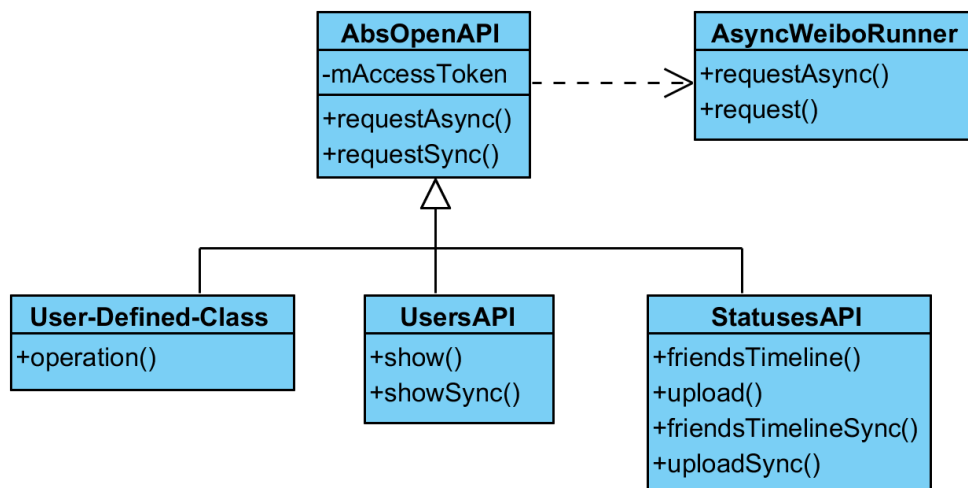
当登录完后，该按钮会变成注销，方便用户注销。如上图右所示。

值得注意一点的是，如果该按钮已从当前 Activity 移除后（如 Activity 销毁后），下次再进入该 Activity 时，该按钮不会自动变成注销（由于 V2.4 SDK 未保存 Token 信息，以后可能会考虑保存所有 Session 相关信息），需要用户手动调用 LoginLogoutButton# setLogoutInfo 设置 Token 信息后，才会变成注销。

7. OpenAPI 示例代码分析

在 V2.5.0 ,我们重构了网络模块 ,提供了一个简单易用的 OpenAPI 接口调用框架 ,并封装了一些简单的开放接口 ,如发布微博、获取用户信息等 ,用户可根据自己需要进行调用。另外 ,我们还提供了相应的 JSON 数据解析 ,以供第三方开发者直接调用相应的 parse()函数解析 JSON 数据来获取对象。

OpenAPI 接口框架的类图如下 :



如上图所示 ,AbsOpenAPI 做为一个基类 ,提供**同步请求**和**异步请求**两种接口。对于我们的各个类型的接口 ,只需要从其继承过来 ,拼接调用参数 ,调用对应方法即可。

为了适应各种不同的需求 ,我们的每个 OpenAPI 同时提供了同步和异步的网络请求接口。

如 : StatusesAPI#upload 即为异步 API , StatusesAPI#uploadSync 即为同步 API。

注 : 同步接口适用于第三方开发者有自己的异步请求机制。

7.1 用户信息接口

1) 通过 Token 初始化 UsersAPI 接口

```
mUsersAPI = new UsersAPI(mAccessToken); // 获取用户信息接口
```

2) 实现异步请求接口回调 , 并在回调中直接解析 User 信息

```
private RequestListener mListener = new RequestListener() {
    @Override
    public void onComplete(String response) {
        if (!TextUtils.isEmpty(response)) {
            // 调用 User#parse 将JSON串解析成User对象
            User user = User.parse(response);
            .....
        }
    }
}
```

3) 调用接口

```
long uid = Long.parseLong(mAccessToken.getUid());
mUsersAPI.show(uid, mListener);
```

7.2 邀请好友接口

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>

[邀请好友接口权限开通方法见附录 8.4](#)，集成该接口步骤如同注销 Token 一样。

1) 实现 RequestListener 接口

2) 创建邀请接口参数

在初始化 InviteApi 类的实例时，需要设置以下内容：

| 参数名称 | 作用 |
|-------------------------------|---|
| InviteAPI.KEY_TEXT (必填) | 要回复的私信文本内容。文本大小必须小于 300 个汉字 |
| InviteAPI.KEY_URL (可选) | 邀请点击后跳转链接。默认为当前应用地址 |
| InviteAPI.KEY_INVITE_LOGO(可选) | 邀请 Card 展示时的图标地址，大小必须为 80px X 80px，仅支持 PNG、JPG 格式 |

实现代码如下：

```
JSONObject jsonObject = new JSONObject();
try {
    jsonObject.put(InviteAPI.KEY_TEXT, "这个游戏太好玩了，加入一起玩吧");
    jsonObject.put(InviteAPI.KEY_URL,
        "http://app.sina.com.cn/appdetail.php?appID=770915");
    jsonObject.put(InviteAPI.KEY_INVITE_LOGO,
        "http://hubimage.com2us.com/hubweb/contents/123_499.jpg");
} catch (JSONException e) {
    e.printStackTrace();
}
```

3) 创建 InviteAPI 类的实例，并调用其 InviteAPI#sendInvite 方法：

```
new InviteAPI(accessToken).sendInvite(uid, jsonObject, mInviteRequestListener);
```

8. 社会化组件

8.1 关注组件 (AttentionComponentView)

创建请求参数

```

/**
 * 创建请求参数（如果用户已经授权，并且有 token）
 * @param appKey
 * @param token
 * @param attentionUid 需要 关注/取消关注 的用户 UID
 * @param attentionScreenName 需要 关注/取消关注 的用户昵称(attentionUid 和 attentionScreenName 两者选其一就行)
 * @param listener 如果想获取授权信息，需要传出授权回调 Listener
 * @return
 */
public static RequestParam createRequestParam(String appKey,
                                             String token, String attentionUid, String attentionScreenName,
                                             WeiboAuthListener listener)

/**
 * 创建请求参数（如果用户没有授权）
 * @param appKey
 * @param attentionUid 需要 关注/取消关注 的用户 UID
 * @param attentionScreenName 需要 关注/取消关注 的用户昵称(attentionUid 和 attentionScreenName 两者选其一就行)
 * @param listener 如果想获取授权信息，需要传出授权回调 Listener
 * @return
 */
public static RequestParam createRequestParam(String appKey, String attentionUid, String attentionScreenName,
                                             WeiboAuthListener listener)

```

设置请求参数

```
public void setAttentionParam(RequestParam param)
```


8.2 评论组件 (CommentComponentView)

创建请求参数

```
/**
 * 创建请求参数（如果用户已经授权，并且有 token）
 * @param appKey
 * @param token
 * @param commentTopic 评论的话题
 * @param commentContent 评论的内容
 * @param category 评论的内容的分类
 * @param listener 如果想获取授权信息，需要传出授权回调 Listener
 * @return
 */
public static RequestParam createRequestParam(String appKey,
                                             String token, String commentTopic, String commentContent, Category category,
                                             WeiboAuthListener listener)

/**
 * 创建请求参数（如果用户没有授权）
 * @param appKey
 * @param commentTopic 评论的话题
 * @param commentContent 评论的内容
 * @param category 评论的内容的分类
 * @param listener 如果想获取授权信息，需要传出授权回调 Listener
 * @return
 */
public static RequestParam createRequestParam(String appKey, String commentTopic, String commentContent,
                                             Category category, WeiboAuthListener listener)
```

设置请求参数

```
public void setCommentParam(RequestParam param)
```

9. 支付组件

微博支付需要单独申请，开通请联系：wb_pay_kf@vip.sina.com

9.1 支付示例

1) 初始化支付环境

```
// 初始化环境
IWeiboShareAPI mWeiboShareAPI = WeiboShareSDK.createWeiboAPI(this, Constants.APP_KEY, false);
// 注册第三方应用到客户端中 注册不成功则无法支付
mWeiboShareAPI.registerApp();
// 微博当前版本是否支持支付
```

2) 生成订单参数

```
// 生成可以支付的订单参数，格式如下
String order =
"source=1941657700&seller_id=3292350247&sign_type=md5&notify_url=http%3A%2F%2Fwww.baidu.com&
out_trade_no=test001427189083&subject=%E6%8E%A5%E5%85%A5%E6%94%B6%E9%93%B6%E5%8F%B0%E5%AE%9E
%E4%BE%8B&pay_type=1&total_fee=0.01&body=%E6%8E%A5%E5%85%A5%E6%94%B6%E9%93%B6%E5%8F%B0%E5%AE
%9E%E4%BE%8B&show_url=http%3A%2F%2Fwww.baidu.com&it_b_pay=3d&extra=%E9%A2%9D%E5%A4%96%E9%80%
8F%E4%BC%A0%E5%8F%82%E6%95%B0&sign=f3b18ea552afa60d68338c306cac2604&is_route=1";
```

3) 呼起支付

```
if (isSupportPay) {
    mWeiboShareAPI.launchWeiboPay(order);
}
```

注:无法正常呼起多为订单参数不正确或者第三方 APP 没有得到认证

9.2 支付参数说明

| 参数 | 参数说明 | 类型 | 是否必填 | 备注 |
|---------------------|----------|--------|------|---|
| source | AppKey | String | 是 | 在微博开放平台申请应用时分配的 AppKey，调用接口时候代表应用的唯一身份 |
| seller_id | 商户微博 UID | int | 是 | |
| sign_type | 签名方式 | String | 是 | 目前签名仅支持 md5 |
| sign | 签名 | String | 是 | 根据 url 参数以及密钥生成，详见“3.签名机制” |
| notify_url | 后台回调地址 | String | 是 | 用来通知商户支付结果，判断支付成功请仅以此接口为准，最长 255 位 |
| out_trade_no | 商户订单号 | String | 是 | 商户网站唯一订单号，最长 64 位 |
| subject | 商品名称 | String | 是 | 该参数最长为 128 个汉字 |
| pay_type | 支付业务类型 | int | 是 | 根据业务规划给出参数列表，如 1=实物商品售卖，2=虚拟商品售卖，3=捐助，4=权益等 |
| price | 商品单价 | float | 否 | 单位为 RMB-Yuan. 取值范围 [0.01,9999999999.00],精确到小数点后俩位. 规则: price 和 quantity 能替代 total_fee 。即存在 price 和 quantity 就不能存在 total_fee ; 存在 total_fee 就不能存在 price 和 quantity |
| quantity | 购买数量 | int | 否 | 规则: price 和 quantity 能替代 total_fee 。即存在 price 和 quantity 就不能存在 total_fee ; 存在 total_fee 就不能存在 price 和 quantity |
| total_fee | 交易金额 | float | 否 | 单位为 RMB-Yuan.取值范围 [0.01,9999999999.00],精确到小数点后俩位 |
| body | 商品描述 | String | 否 | 该参数最长为 500 个汉字 |

| 参数 | 参数说明 | 类型 | 是否必填 | 备注 |
|--------------------|--------|--------|------|---|
| show_url | 页面回跳地址 | String | 否 | 用来通知商户支付结果, 用户在网页支付成功页面, 点击“返回商户”时的回跳地址, 最长 255 位 |
| it_b_pay | 超时时间 | String | 否 | 交易请求自创建时间起, 自动关闭的时间, 取值范围: 1m(分钟)、1h(小时)、1c(当天 0 点)、15d(天), 不接受小数点, 默认 3d |
| is_route | 必传值为 1 | String | 是 | |
| req_channel | 请求渠道 | String | 是 | 1: 跨端支付 |
| extra | 额外参数 | String | 否 | 发起支付时透传的参数, 可为不包含"="、"&"等特殊字符的字符串 |

9.3 支付接口错误返回码说明

| 返回错误码 error_code | 含义 error | 说明 |
|---------------------|-------------|----------------------|
| 1000000 | Succeeded | 成功 |
| 220001 | 操作失败 | 操作失败 |
| 220002 | 非法请求 | 非法请求 |
| 220003 | 请登录 | 登录失败 |
| 220004 | 验证签名失败 | 验证签名失败 |
| 220005 | xxxx | 参数验证错误, 以 error 描述为准 |
| 220006 | 无效的订单号 | 无效的订单号 |
| 220007 | 用户信息错误 | 用户信息错误 |
| 220008 | 无效的 uid | 无效的 uid |

| 返回错误码 error_code | 含义 error | 说明 |
|---------------------|-------------|---------|
| 220009 | 无效的价格 | 无效的价格 |
| 220010 | 无效 id | 无效 id |
| 220011 | 无效 time | 无效 time |
| 220012 | 无效平台 | 无效平台 |
| 220013 | 数据已生成 | 数据已生成 |
| 220014 | 数据为空 | 数据为空 |
| 220015 | 无效支付渠道 | 无效支付渠道 |
| 220016 | 无效操作 | 无效操作 |
| 220017 | 无效商家 | 无效商家 |

9.4 签名机制

9.4.1 生成待签名的字符串

9.4.1.1 需要参与签名的参数

在请求参数列表中，除去 sign，sign_type 两个参数外，其他需要使用到的参数皆是要签名的参数。（个别接口中参数 sign_type 也需要参与签名）

在通知返回参数列表中，除去 sign，sign_type 两个参数外，凡是通知返回回来的参数皆是要签名的参数。

9.4.1.2 需生成待签名字符串

对数组里的每一个值从 a 到 z 的顺序排序，若遇到相同首字母，则看第二个字母，以此类推。排序完成之后，再把所有数组值以 “&” 字符连接起来，组成的字符串便是待签名的字符串。

9.4.1.3 注意

没有值的参数无需传递，也无需包含到待签名数据中。

根据 HTTP 协议要求，传递参数中的值中如果存在特殊字符（如：&，@等），那么该值需要做 URL Encoding，这样请求接收方才能接收到正确的参数值。这种情况下，待签名数据应该是原生值而不是 encoding 之后的值。

9.4.2 签名

9.4.2.1 MD5 签名

在 MD5 签名时，需要私钥参与签名。MD5 的私钥是以英文字母和数字组成的字符串。商户提交完资料审核后，会随同开发文档等一起发送到指定邮箱。

9.4.2.2 请求时签名

当拿到请求时的待签名字符串后，需要把私钥直接拼接有待签名字符串后面，形成新的字符串，利用 MD5 的签名函数对这个新的字符串进行签名运算，从而得到 32 位签名结果字符串（该字符串赋值于参数 sign）。

9.4.2.3 通知返回时验证签名

当获得通知返回时的待签名字符串后，同理，需要把私钥直接拼接有待签名字符串后面，形成新的字符串，利用 MD5 的签名函数对这个新的字符串进行签名运算，从而得到 32 位签名结果字符串。此时，这个新的字符串需要与微博支付通知返回参数中的参数 sign 的值进行验证是否相等，来判断签名是否验证通过。

10 问题

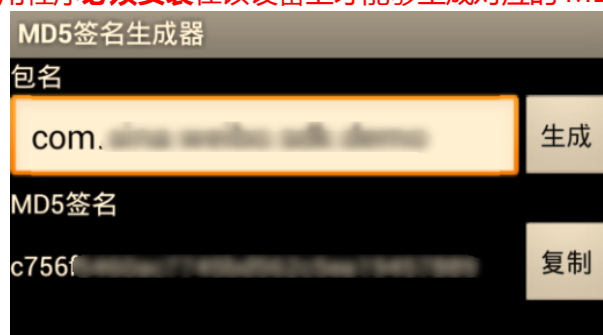
10.1 如何使用签名工具获取应用的签名？

☞ 签名工具下载地址：

https://github.com/mobileresearch/weibo_android_sdk/blob/master/app_signatures.apk

☞ 使用方式：首先要安装您需要签名的应用，然后再安装该工具，安装完后，输入您的应用程序的包名，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



10.2 如何实现 LinkCard 效果？

6.2.1 (3) 中分享视频所示效果是 LinkCard 效果，如下图左所示。分享普通链接是没有这种效果的。如果你想要你分享出去的内容展示成 LinkCard 的样子，**其前提条件是你分享的链接支持 LinkCard**。即 *LinkCard 的分享开关，是基于分享出来的链接的域名*。打个比方，开启 v.youku.com 的 LinkCard 开关，从而发布器中带这个链接的，都能呈现为 LinkCard，该功能并不是针对于某个特定的应用，而是针对于特殊的网内容的。如果某个网站需要进行 LinkCard 商务合作，请联系 BD。详情请阅读：移动客户端接入（<http://t.cn/aex4JF>）

10.3 如何实现附件栏集成分享？

关于第三方应用注册到微博客户端效果，如下图右所示：

如果优质的第三方应用也想被显示在此处，也需要进行商务合作，默认情况下是不被显示在此处的。详情请阅读：移动客户端接入（<http://t.cn/aex4JF>）

注意：如果没有进行过商务合作，是无法进行该分享的。



10.4 第三方如何申请接口权限？

请参考：<http://open.weibo.com/wiki/好友邀请>

具体实现步骤：

- 1) 申请好友邀请接口使用权限，通过邮件 `open_api@sina.com` 申请，需要详细描述应用的功能、服务等信息。
- 2) 在 OAuth2 的 Scope 域中添加好友邀请权限，主要是在 demo 工程的 Constants 类的 SCOPE 变量添加 "invitation_write" 值。
- 3) 获取互粉好友，调用 `friendships/friends/bilateral` 接口，获取当前用户微博互粉好友列表 Uid。
- 4) 发送邀请，调用 `messages/invite` 接口实现向指定好友发送邀请。