

ShareSDK 使用说明

(by sharesdk.cn)

集成

描述获取和集成 ShareSDK 的基本流程

接口调用

详细介绍 ShareSDK 各种接口的调用方法

快捷分享

举例说明如何使用最快的办法集成 ShareSDK

插件服务

介绍与 ShareSDK 的插件服务框架有关的资料

常见问题

对一些常见问题的答疑

欢迎访问我们的 **Wiki**
获取更详细的信息

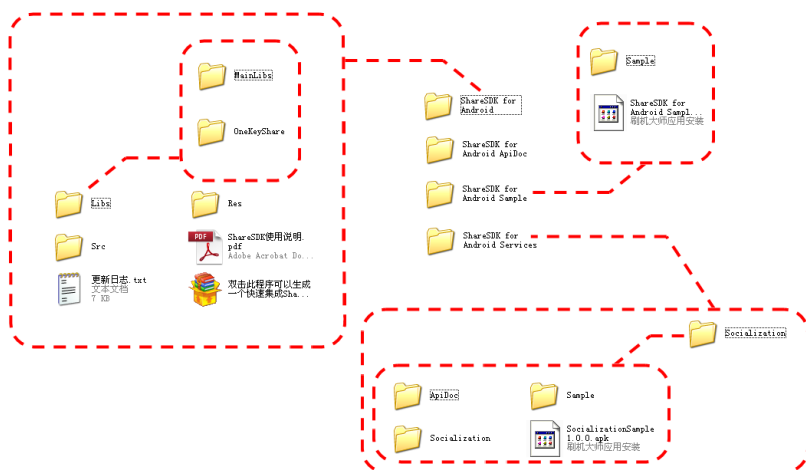
集成 ShareSDK

本章内容包括：

- 获取 ShareSDK
- 导入 ShareSDK
- 添加应用信息
- 配置 **AndroidManifest.xml**
- 添加代码
- 微信、易信的签名和注册
- 平台对应列表
- 应用信息注册地址列表

获取 ShareSDK

为了集成 ShareSDK，您首先需要到 [ShareSDK](#) 官方网站注册并且创建应用，获得 ShareSDK 的 Appkey，然后到 SDK 的[下载页面](#)下载 SDK 的压缩包，解压以后可以得到如下图的目录结构：



其中 ShareSDK 在“ShareSDK for Android”目录下，此目录中的“Libs”包含“MainLibs”和“OnekeyShare”，分别是 ShareSDK 的核心库和“快捷分享”的源码库，说明文档也在“ShareSDK for Android”目录下，集成 ShareSDK 前请务必仔细阅读。“ShareSDK for Android ApiDoc”包含 Mainlibs 和 OnekeyShare 的 JavaDoc，供开发者开发时查阅。“ShareSDK for Android Sample”包含 ShareSDK 的功能演示代码和 apk 文件，Sample 源码的 JavaDoc 也在“ShareSDK for Android ApiDoc”中。“ShareSDK for Android Services”包含 ShareSDK 已经发布的“插件服务”，暂时我们只提供了“评论与赞”服务，压缩包中已经提供了此插件服务的依赖库、示例代码、JavaDoc

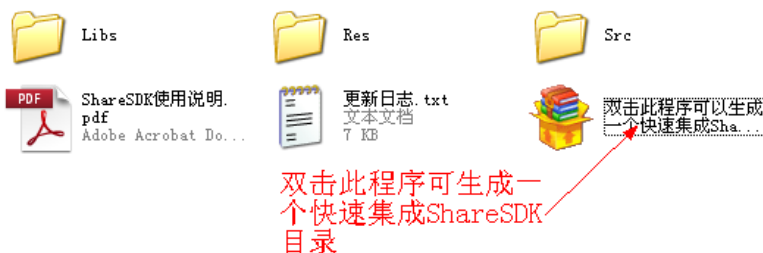
和 apk 文件。

导入 ShareSDK

ShareSDK 有两种集成方式：直接复制 jar 包到目标项目 libs 目录和项目引用。

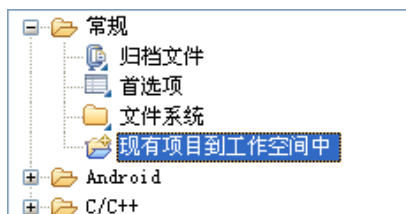
如果使用复制 jar 的方法，需要除了复制 MainLibs/libs 中的 jar 外，还需要复制 MainLibs/res 中图片和 strings，否则会出现授权时找不到资源的问题。而且如果您的项目也集成了快捷分享，还需要复制 OneKeyShare/src 中的源码、OneKeyShare/res 中的资源和 OneKeyShare/libs 下的 mframework.jar 到您的项目中。由于操作繁琐，所以 ShareSDK **建议开发者使用的是项目引用的方式**。对于直接复制 jar 和资源的方式，可以参考 ShareSDK 的 Sample 项目。

由于直接复制 jar 包和资源的集成方式比较麻烦，Share SDK 提供了快速集成的程序，在 Windows 下可以直接双击执行。这个程序会自动搜集集成 SDK 所需要的 jar 和资源，完成以后可以通过复制目标目录中的文件到您项目中覆盖，再修改快捷分享项目中对源码文件对 R 文件的引用就行了。极大简化了“直接复制 jar 和资源”这种集成方式的操作步骤。

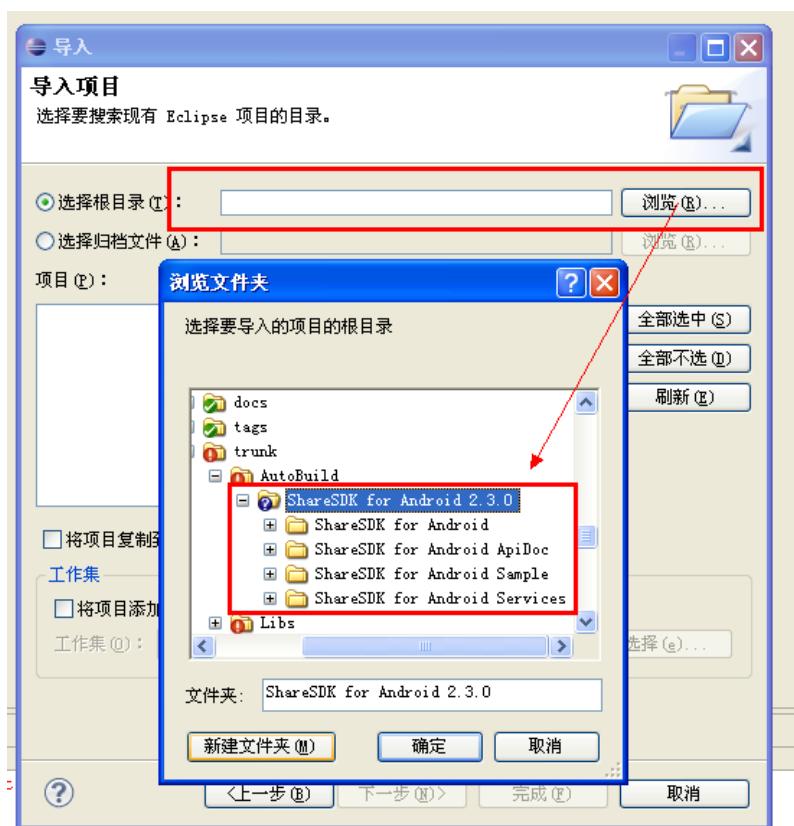


下面演示如何通过项目引用的方式集成 ShareSDK:

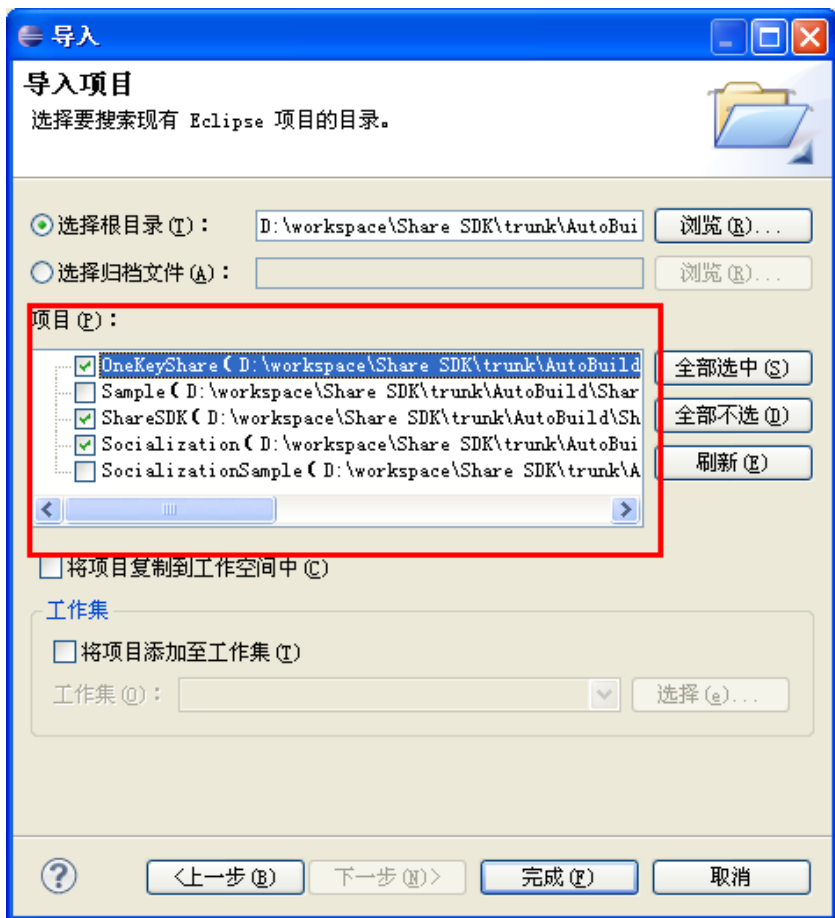
1)将 ShareSDK 下的 Libs 目录解压到 Eclipse 的工作空间，然后在 Eclipse 中，选择“文件”－“导入”，并选择“现有的工作空间”：



并在“下一步”的页面中，选择上文的解压目录：



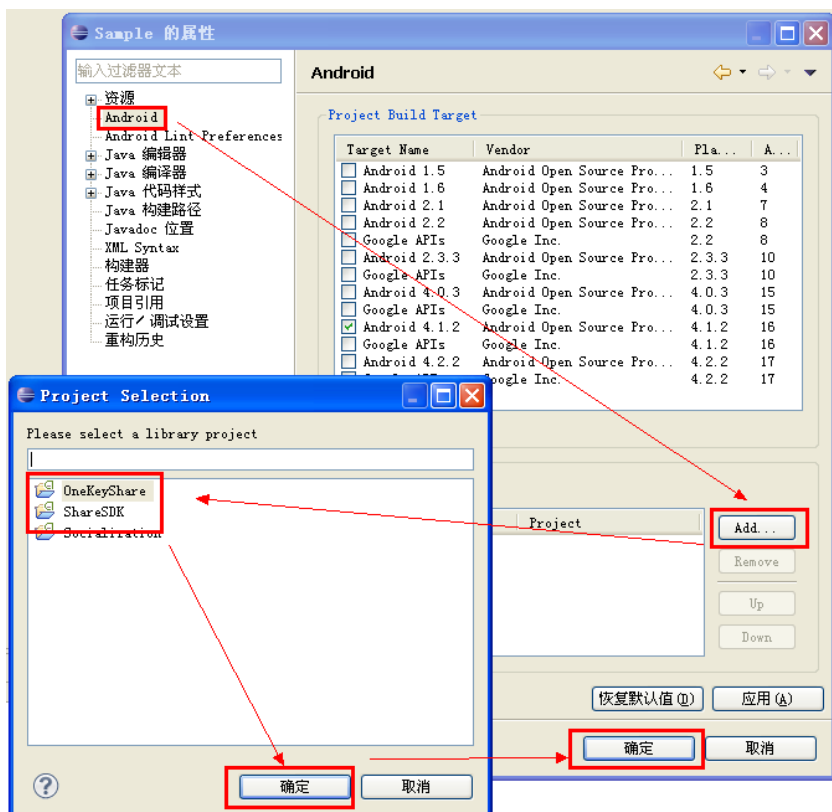
点击“确定”，Eclipse 会自动找到“OnekeyShare”、“Sample”、“ShareSDK（即 MainLibs）”、“Socialization（即评论与赞的依赖项目）”和“SocializationSample”目录下的两个项目：



如果您不需要快捷分享功能，可以取消 **OneKeyShare** 项目的勾选，如果您不需要集成评论与赞服务，可以取消 **Socialization** 项目的勾选。否则请按“确定”，导入项目。下面简单说明 MainLibs 和 OnekeyShare 的集成，Socialization 的集成类似于 OnekeyShare，将在后文详细说明。

2) 被导入的三个项目默认是“Is Library”的，而且 OneKeyShare 也默认引用了 MainLibs，所以您并不需要为这两个项目做更多的设置。但是 **如果您修改了它们的相对路径，使之不在同一目录下，则需要手动修改其引用方式**。具体操作和下面步骤类似。

3) 右键您的项目，选择“属性”。在弹出的窗口左侧选择“Android”，然后转至右下侧的“Library”中点击“Add”，如果您的项目希望集成 ShareSDK 的 GUI 分享功能，请选择“**OneKeyShare**”，否则选择“MainLibs”即可。保存设置并退出属性窗口。



完成上述步骤，“刷新”并“清理”一下您的项目，若未提示错误，ShareSDK 已经导入到您的项目中。接下来就是配置不同平台的应用信息和注册授权以及快捷分享的 Activity 了。

对于 1.2.0 以前已经集成 ShareSDK，现在要升级的开发者，可以参考“复制 jar 和资源文件”的方式来升级 SDK，也可以将您 libs 中对 ShareSDK 所引用的 jar 删除，然后依照上述的步骤重新引用。由于 ShareSDK 2.0 并不兼容 1.x 的版本，因此会有一些代码需要改动。**主要的修改原则是将所有包含“Weibo”名称的字段和方法，以“Platform”来替换；AbstractWeibo 被拆分为两个，对于 ShareSDK 而言的静态方法被分配到类 ShareSDK 中，其他的和具体平台操作有关的成员，被分配到类 Platform 中。**更多的详细内容，可以参考我们 Wiki 上的 [Android 2.0 升级指南](#)。

ShareSDK 的快捷分享只提供源码，并没有编译好的 jar。MainLibs 的 libs 目录中包含的是 ShareSDK 的核心类库，不同的 jar 对应不同的平台，其对应关系可以参考后文的平台对应列表。对于大部分的开发者，这些平台并不都是需要的，这时候可以直接删除此目录下不需要平台的 jar 包即可，直到后续需求变更，需要添加新的平台，再复制新平台的 jar 到此目录下就行了。

添加应用信息

ShareSDK 有不同的应用信息配置方式。**可以在 ShareSDK 的[应用管理后台](#)中配置，或者通过“assets/ShareSDK.xml”文件来配置，或者通过“setPlatformDevInfo(String, HashMap<String, Object>)”方法，指定平台名称，根据 ShareSDK.xml 文件中此平台应用信息的各个字段传入其值。**开发者可以自

行选择，不过这三种设置方式各有区别：第一种方式优先级最高，可以实现“动态配置应用信息”的功能，但是一旦脱离网络，ShareSDK 可能无法运作；第二种方式是优先级最低的方式，但是它最为方便、集中；最后使用代码设置的方式是最灵活的方式，开发者可以在代码里面写死应用信息，也可以通过私有协议，从自己的服务器上动态获取应用信息，其优先级居第一种和第二种之间。下面介绍一下如何使用“assets/ShareSDK.xml”配置您项目的应用信息列表。

首先参考**应用信息注册地址列表**章节，到您即将集成的社交平台上注册您的应用，并获取应用信息。

获取信息以后，**将这些信息都存放在您项目的“assets/ShareSDK.xml”中**。请到“Res”中将 ShareSDK.xml 复制到您项目的“assets”下，打开文件，然后根据不同的平台的数据，如下面的例子所示，替换您在这个平台上的开发者信息：

```
<ShareSDK
    AppKey="填写您在 ShareSDK 上注册到的 AppKey" />

<SinaWeibo
    SortId="此平台在您分享列表中的位置，整型，数值越大越靠后"
    AppKey="填写您在新浪微博上注册到的 AppKey"
    AppSecret="填写您在新浪微博上注册到的 AppSecret"
    Id="自定义字段，整型，用于您项目中对此平台的识别符"
    RedirectUrl="填写您在新浪微博上注册的 RedirectUrl"
    ShortLinkConversationEnable="布尔值，标记是否作短链转换"
    Enable="布尔值，标记此平台是否有效" />
```

ShareSDK.xml 以 XML 格式存储数据，每一个平台一个块，除了社交平台外，开发者在 ShareSDK 注册应用时得到的 Appkey 需要填写在块“ShareSDK”中，如果此 Appkey 不是

开发者自己的 Appkey，则将来在 ShareSDK 应用后台的统计数据将不正确。ShareSDK 的每一个平台都具备 SortId、Id、ShortLinkConversationEnable 和 Enable 四个字段，除此之外的字段（如新浪微博的 AppKey、AppSecret、RedirectUrl 等字段）需要到目标平台上注册应用以后得到，请正确填写这些字段的数据，否则 ShareSDK 无法完成授权，则后续的其它操作也将无法执行。关于应用信息不同字段的更详细解释，可以参考 ShareSDK.xml 文件头部的说明。

配置 AndroidManifest.xml

不同的集成度需要在 AndroidManifest.xml 中添加的内容不一样。但是首先您需要添加下面的权限列表：

```
<uses-permission android:name="android.permission.GET_TASKS" />
<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.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.MANAGE_ACCOUNTS"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
```

这些权限将允许您的项目和 ShareSDK 判断当前应用是否“前置”、获取连接网络的权限、获取您的设备网络状态的权限、实现 https 安全连接的权限、读取手机设备状态的权限和保存必要配置的权限。一般来说，即便不集成 ShareSDK，大部分的项目也都会注册申请这些权限。

其次，为了授权操作可以顺利完成，需要在 application

下注册下面的 Activity:

```
<activity
    android:name="cn.sharesdk.framework.ShareSDKUIShell"
    android:theme="@android:style/Theme.Translucent.NoTitleBar"
    android:configChanges="keyboardHidden|orientation|screenSize"
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateHidden|adjustResize" />
```

ShareSDKUIShell 的路径是固定的，一定要在“cn.sharesdk.framework”下，因为它在 Share-Core 中。

最后，如果您的项目集微信的三个平台，则需要要在 application 下注册下面的回调 Activity:

```
<activity
    android:name=".wxapi.WXEntryActivity"
    android:theme="@android:style/Theme.Translucent.NoTitleBar"
    android:configChanges="keyboardHidden|orientation|screenSize"
    android:exported="true"
    android:screenOrientation="portrait" />
```

而如果您的项目集易信的两个平台，则需要要在 application 下注册下面的回调 Activity:

```
<activity
    android:name=".yxapi.YXEntryActivity"
    android:theme="@android:style/Theme.Translucent.NoTitleBar"
    android:configChanges="keyboardHidden|orientation|screenSize"
    android:exported="true"
    android:screenOrientation="portrait" />
```

这两个类的路径是需要根据您项目的包名来确定，如果路径错误，您将收不到微信和易信客户端的操作回调，因此

ShareSDK 也无法给予您操作回调。为了避免出错，请使用相对路劲的方式，**直接复制上面的代码到您的 AndroidManifest.xml 中即可。**

添加代码

集成 ShareSDK 需要至少在两个地方添加代码，包括：

1) 打开您项目的入口 Activity，在其 onCreate 中插入下面的代码：

```
ShareSDK.initSDK(this);
```

这行代码会初始化 ShareSDK，此后对 ShareSDK 的操作都依次为基础。如果**不在所有 ShareSDK 的操作之前调用这行代码，会抛出空指针异常。**

2) 在您项目的出口 Activity 的 onDestroy 方法的第一行插入下面的代码：

```
ShareSDK.stopSDK(this);
```

这行代码会结束 ShareSDK 的统计功能并释放资源。**如果这行代码没有被调用，那么“应用启动次数”的统计将不会准确，**因为应用可能从来没有被关闭。

initSDK 是可以重复调用的，其实 ShareSDK 建议在您不确定的时候调用这个方法，来保证 ShareSDK 被正确初始化。**而 stopSDK 一旦调用了，就必须重新调用 initSDK 才能使用 ShareSDK 的功能**，否则会出现空指针异常。

至于如何调用授权、获取资料、分享等操作，请参考后续**接口调用和快捷分享**等章节。

微信、易信的签名和注册

对于 ShareSDK 的大部分平台，只要您参考上面的集成步骤引入到您的项目以后就可以开始进行工作了。但是对于微信和易信两个客户端（共五个平台）却还不行。因为这两个客户端的开放平台要求 **android** 第三方开发者需要提供自己项目的包名和签名，否则其客户端将拒绝第三方应用的分享操作。下面简要叙述微信开放平台注册应用和添加这些信息的操作流程，易信开放平台的操作与之类似：

1) 您需要到[微信开放平台](#)上注册一个开发者账号，然后利用这个账号添加一个*移动应用*。



应用创建出来以后，您会得到一个 **AppId**，将这个 Id 添加到

assets/ShareSDK.xml 中。

移动应用

微信开放平台目前对于移动应用开放的API包括：
从移动应用中分享信息给用户的微信好友。

AppID

AppID 是你的应用与微信通信的标识。每个应用对应一个唯一的AppID。

wx790694568c8

恭喜你，你已经成功获得你的专有AppID，使用它，你可以开始使用微信SDK进行相应的开发工作了。

当安装你的应用的微信用户尝试打开来自你的应用的信息时，微信将引导用户下载安装。（iOS、Android平台至少需要填写一个，当任一平台的任一项信息被填写，该平台下的其他项也必须填写）

☐ iOS平台应用

☐ Android平台应用

勾选这里，填写Android平台的信息

2)添加 Android 平台的信息。这些信息包括：**下载地址**、**应用签名**和**包名**。

☒ Android平台应用

下载地址 (Android)

Google Play™或其他应用商店中的下载页面URL。不允许使用apk包下载URL。

http://www.shareSDK.cn

应用签名 (Android)

用于对当前应用进行二次身份校验，开发者可以使用**签名生成工具**直接从安装当前应用的手机中获取。应用签名由开发者的keytool应用的keystore文件决定。

下载这个apk并安装到你的手机

包名 (Android)

这个是你项目的包名
请不要乱填

应用在一台设备上的唯一标识，在manifest文件里面声明，该包名应和正式发布应用的包名一致。例如，微信的包名为com.tencent.mm

com.example.ihatwechat

上图中的下载地址 **不可以是应用 apk 文件的直接地址**，而包名则是您**应用的包名**。至于应用签名，事实上您应用签名文件（keystore 文件）的 **MD5 值**。有多种计算方式：

（1）使用微信提供的[签名计算工具](#)，来签名：

如果您希望通过这个方法来计算签名，则需要下载微信的签名计算工具。此处需要注意的是下载回来的文件一定是“apk”的，如果您下载回来的文件是 zip 或者后缀名，请修改其后缀名为“apk”，并复制到您的手机上安装。

然后为您的应用导出一个**签过名的 apk**。请注意，这个签名应该是您将来**发布应用时的签名**，也就是所谓的“正式包签名”。这里需要解释一下，微信并不要求具体使用的签名文件的性质，但是修改签名经常会导致问题，所以为了避免不必要的麻烦，ShareSDK 强烈建议您使用正式包的签名。

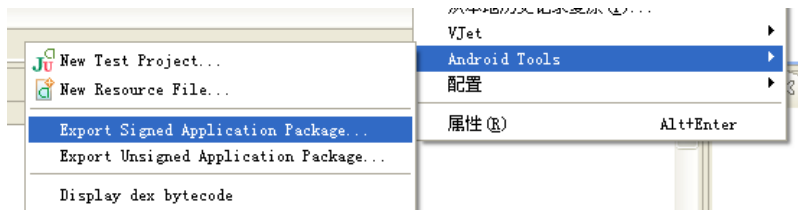
安装导出的 apk 文件，然后运行微信的签名计算工具。并在其中输入您应用的包名，微信会自动计算您应用所签 keystore 的 MD5：



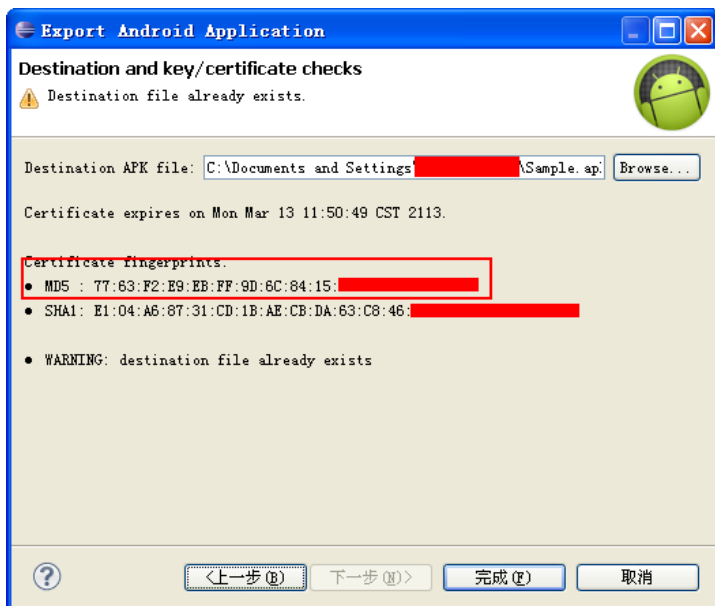
转至微信开放平台，将计算好的结果填写到其中并保存。

(2) 利用 Eclipse 导出签名时显示的签名 MD5 值：

利用此方法，要求您下载**最新版本的 Eclipse Android 开发插件 (ADT)**。确保此条件后，右键您的项目，选择“Android Tools” — “Export Signed Application Package...”：



此时会弹出一个导出签名 apk 的窗口，输入您应用正式签名和密码以后，会在导出页面中，显示您签名的 MD5 值。将其复制出来，调整为小写，并去除多余的分割符。转至微信开放平台，将计算好的结果填写到其中并保存即可。



(3) Linux (Ubuntu) 下，利用 Shell 直接计算：

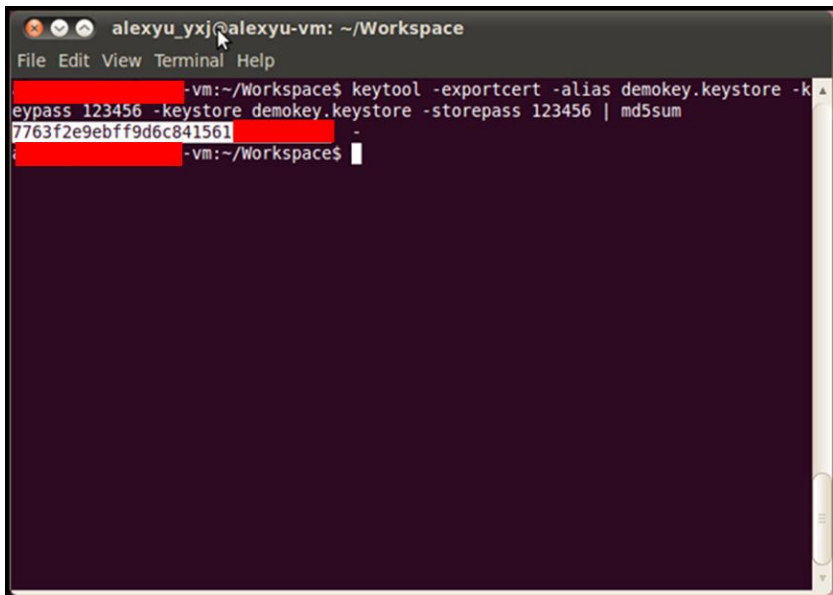
如果您使用 Ubuntu 操作系统，可以直接利用下面的 Shell 命令计算出签名文件的 MD5 值：

```
keytool -exportcert -alias <keystore 别名> -keypass <别名 密码>  
-keystore <keystore 路径> -storepass <keystore 密码> | md5sum
```

如：

```
keytool -exportcert -alias demokey.keystore -keypass 123456 -keystore  
demokey.keystore -storepass 123456 | md5sum
```

其实出如下：



```
alex_yxj@alex-yu-vm: ~/Workspace  
File Edit View Terminal Help  
-vm:~/Workspace$ keytool -exportcert -alias demokey.keystore -k  
eypass 123456 -keystore demokey.keystore -storepass 123456 | md5sum  
7763f2e9ebff9d6c841561 -  
-vm:~/Workspace$
```

将其复制出来，转至微信开放平台，将计算好的结果填写到其中并保存即可。

完成上述的注册操作以后，将您的项目提交微信开放平台审核，**审核通过以后**，您的项目就可以开始调用微信客户端进行分享了。**分享时，需要利用上文提到的 keystore 来给您的 apk 签名，签名以后放到您的机器上、安装，以后才能调用分享。**不能直接在 Eclipse 上 run，因为签名和包名都已经捆绑，若不是注册时填写的签名，微信客户端将拒绝请求。

为了演示分享功能 Sample 包中携带了一个"**demokey.keystore**"的 keystore 文件，其密码是“**123456**”，如果您在调试 Sample 项目的时候集成了微信的平台，就需要使用这个 keystore 对 apk 进行签名，之后才能尝试微信的分享。但是请注意：**这个 keystore 已经和“cn.sharesdk.demo”这个包名捆绑**，如果您的项目不使用这个包名，是不能直接使用这个 keystore 的。

平台对应列表

下面是 ShareSDK 当前所支持平台及其 jar 包的对应表^[1]:

平台	Jar 包
新浪微博	ShareSDK-SinaWeibo.jar
腾讯微博	ShareSDK-TencentWeibo.jar
QQ	ShareSDK-QQ.jar
QQ 空间	ShareSDK-QZone.jar
微信好友 ^[2]	ShareSDK-Wechat.jar
微信朋友圈 ^[2]	ShareSDK-Wechat-Moments.jar
微信收藏 ^[2]	ShareSDK-Wechat-Favorite.jar
Facebook	ShareSDK-Facebook.jar
Twitter	ShareSDK-Twitter.jar
人人网	ShareSDK-Renren.jar
开心网	ShareSDK-KaiXin.jar

邮件	ShareSDK-Email.jar
信息	ShareSDK-ShortMessage.jar
搜狐微博	ShareSDK-SouhuMicroBlog.jar
网易微博	ShareSDK-NetEaseMicroBlog.jar
豆瓣	ShareSDK-Douban.jar
有道云笔记	ShareSDK-YouDao.jar
印象笔记	ShareSDK-Evernote.jar
Linkedin	ShareSDK-Linkedin.jar
Google+	ShareSDK-GooglePlus.jar
FourSquare	ShareSDK-Foursquare.jar
搜狐随身看	ShareSDK-SohuSuishenkan.jar
Flickr	ShareSDK-Flickr.jar
Pinterest	ShareSDK-Pinterest.jar
Tumblr	ShareSDK-Tumblr.jar
Dropbox	ShareSDK-Dropbox.jar
Instagram	ShareSDK-Instagram.jar
Vkontakte	ShareSDK-Vkontakte.jar
易信好友 ^[3]	ShareSDK-Yixin.jar
易信朋友圈 ^[3]	ShareSDK-Yixin-Moments.jar

注：

[1] 由于所有平台都依赖于 ShareSDK-Core.jar，故此表略去每一项中的此包，以免冗余

[2] 微信的三个平台（微信好友、微信朋友圈和微信收藏）除了依赖 ShareSDK-Core.jar 外，还依赖 ShareSDK-Wechat-Core.jar，使用的时候需注意不要遗漏。

[3] 易信的两个平台（易信好友和易信朋友圈）除了依赖 ShareSDK-Core.jar 外，还依赖 ShareSDK-Yixin-Core.jar，使用的时候需注意不要遗漏。

应用信息注册地址列表

下面是 ShareSDK 不同平台应用信息注册网站的地址^[1]:

平台	网站
新浪微博	http://open.weibo.com
腾讯微博	http://dev.t.qq.com
QQ 空间 ^[2]	http://connect.qq.com/intro/login/
微信好友 ^[3]	http://open.weixin.qq.com
Facebook	https://developers.facebook.com
Twitter	https://dev.twitter.com
人人网	http://dev.renren.com
开心网	http://open.kaixin001.com
搜狐微博	http://open.t.sohu.com
网易微博	http://open.t.163.com
豆瓣	http://developers.douban.com
有道云笔记	http://note.youdao.com/open/developguide.html#app
印象笔记	https://dev.evernote.com/
Linkedin	https://www.linkedin.com/secure/developer?newapp=
FourSquare	https://developer.foursquare.com/
搜狐随身看	https://open.sohu.com/
Flickr	http://www.flickr.com/services/
Pinterest	http://developers.pinterest.com/
Tumblr	http://www.tumblr.com/developers
Dropbox	https://www.dropbox.com/developers
Instagram	http://instagram.com/developer#
Vkontakte	http://vk.com/dev
易信好友 ^[4]	http://open.yixin.im/

注：

- [1]** Email、信息和 Google+不需要注册即可使用
- [2]** QQ 客户端分享和 QQ 空间使用相同的应用信息配置页面
- [3]** 微信好友、微信朋友圈和微信收藏使用相同的注册信息
- [4]** 易信好友和易信朋友圈使用相同的注册信息

接口调用

本章内容包括：

- 接口调用的原则
- 获取平台列表和实例
- 设置操作监听
- 授权及其页面的自定义
- **Single Sign-On**
- 获取用户资料及第三方登录
- 分享到指定平台
- **ShareParams** 字段释义
- 接收微信客户端消息
- 关注指定用户
- 获取关注列表
- 自定义接口的使用
- 平台数据库的操作

虽然 ShareSDK 提供了“快捷分享”的 UI 辅助功能，但是 ShareSDK 本身是可以独立操作的。为了后续可以更好地介绍快捷分享功能，本章作为说明文档的核心内容，将详细介绍 ShareSDK 所有重要的接口和功能。

接口调用的原则

对于 ShareSDK 的大部分操作，如：授权、获取资料、分享等等。其接口调用都有固定的格式。下面使用伪代码进行描述：

0、调用 *initSDK* 方法

- 1、获取平台实例
- 2、为平台实例添加操作回调
- 3、调用平台实例的方法实现对应的操作

不管您在什么情况下使用 ShareSDK, *initSDK* **都需要在任何操作之前被调用**，正因如此，ShareSDK 强烈建议您在入口 Activity 的 onCreate 方法中就调用 initSDK。此外，您也可以在任何您不确定的时候重复调用此方法，这个方法可以无限重复调用，并不会造成太多效率上的损耗。具体的调用说明，可以参考[添加代码](#)章节的相关说明。

获取平台列表和实例

intiSDK 以后就可以尝试获取平台实例了。“平台实例”是 cn.sharesdk.framework.Platform 子类的实例。事实上 Platform 是不能被实例化的，但是它是 ShareSDK 的核心，利用它就可以完成所有的操作，您甚至不需要知道具体某个平台是怎么操作的。ShareSDK 已经发布了很多社会化组件平台

工具，但并不是每一个平台您都需要。因此您可以参考**集成 ShareSDK** 中的**相关章节**，选择您需要平台，以缩减不必要的包体积。

获取已经集成到您项目中的平台实例有两种办法：

```
// 获取已经注册到 SDK 的平台实例列表
Platform[] platformList = ShareSDK.getPlatformList(context)

// 获取单个平台
Platform plat = ShareSDK.getPlatform(context, TencentWeibo.NAME);
```

第一种办法可以获取已经集成到您项目中的所有平台，通过这个平台数组，您可以结合循环操作，对所有平台进行统一的操作。而第二种方法相对常用，它只是根据您给定的平台名称获取一个平台的实例，然后单独对这个平台进行操作。

设置操作监听

获取平台实例以后还不能立刻执行操作，大部分的操作都是以 **void** 返回值的，因为这些操作内部都将开启线程，不能在线程间等到返回值。故 **ShareSDK 的操作，大量使用了回调**。下面是设置操作回调的方法：

```
Platform weibo = ShareSDK.getPlatform(context, TencentWeibo.NAME);
weibo.setPlatformActionListener(new PlatformActionListener() {

    public void onError(Platform platform, int action, Throwable t) {
        // 操作失败的处理代码
    }
});
```

```
public void onComplete(Platform platform, int action,
    HashMap<String, Object> res) {
    // 操作成功的处理代码
}

public void onCancel(Platform platform, int action) {
    // 操作取消的处理代码
}

});
```

PlatformActionListener 是 ShareSDK 统一的操作回调。它有三个方法：onComplete、onError 和 onCancel。分别表示：成功、失败和取消。一般来说，“取消”的事件很少出现，但是授权的时候会有；**错误的时候，可以通过对其 *Throwable* 参数 *t* 执行 “*t.printStackTrace()*” 可以得到错误的堆栈。**这个堆栈十分重要，如果您向 ShareSDK 的客服提交 bug，请带上这个堆栈，以便他们查询异常。成功的时候，会将操作的结果通过其 HashMap<String, Object>参数 res 进行返回，**返回时的 *res* 已经解析，可以根据不同平台的 *api* 文档，从 *res* 中得到返回体的数据。**

每一个操作的返回，不管成功失败，都会有一个 action，表示这个操作的类型，其值可以参考 [API 文档](#) 的相关说明。**大部分的回调方法都处于网络线程，因此可以简单默认为回调方法都不在主线程，因此如果要在这些方法进行 UI 方面的操作，一定要通过 *Handler* 发送一个消息给外部来处理。**

设置完操作回调，就可以调用不同的操作了。下面就根据这些操作和应用场景，结合例子来说明其使用方法。

授权及其页面的自定义

授权是您接触 ShareSDK 的第一个操作，不管您选择的是“手动授权”还是“自动授权”。

对于大部分的应用来说，手动授权是没有必要的，但是如果您只是想做一个“**账号系统**”，或者是说您的应用不需要注册，只需要是微博的用户，就能登录，那么这个方法还是十分有用途的。

下面是新浪微博授权操作的例子：

```
Platform weibo = ShareSDK.getPlatform(context, SinaWeibo.NAME);
weibo.setPlatformActionListener(paListener);
weibo.authorize();
```

调用 `authorize` 方法，会弹出一个基于 `ShareSDKUIShell` 的授权页面，填写账号和密码以后，会执行授权操作。**这个方法的操作回调 `paListener` 并不实际带回什么数据**，只是通过回调告知外部成功或者失败。但是每一个平台都具备一个 `PlatformDb` 的成员，这里面存储了此平台的授权信息。可以参考章节**平台数据库的操作**的说明，通过方法 `getToken`、`getUserId` 等方法，获取授权用户在此平台上的授权信息。并由此建立“账户系统”。

然后说一下授权页面的自定义方法。`ShareSDK` 的所有 GUI 都基于 `ShareSDKUIShell`，但是 `ShareSDKUIShell` 只是一个容器，真正显示授权页面主体和逻辑是几个 `FakeActivity` 的子类。这些控件在 `ShareSDK` 之外是不能修改的，但是考虑到很多开发者希望可以修改这个页面的布局、逻辑等等代码，因此 `ShareSDK` 为此提供了一个自定义授权页面的方法，操作如下：

1) 打开 AndroidManifest.xml，并如下修改 ShareSDKUI Shell 的注册信息：

```
<activity
    android:name="cn.sharesdk.framework.ShareSDKUIShell"
    android:theme="@android:style/Theme.Translucent.NoTitleBar"
    android:configChanges="keyboardHidden|orientation"
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateHidden|adjustResize" >
    <!--
        Adapter 继承自
        cn.sharesdk.framework.authorize.AuthorizeAdapter，
        可以监听到页面的生命周期，也能获取页面的各种 UI 控件。
        开发者可以通过继承 AuthorizeAdapter，重写其方法，
        并获取各种 UI 来自定义这个页面的行为。
    -->
    <meta-data android:name="Adapter"
        android:value="cn.sharesdk.demo.MyAdapter" />
</activity>
```

其中的 cn.sharesdk.demo.MyAdapter 是一个继承自 cn.sharesdk.framework.AuthorizeAdapter 的类。*开发者可以根据自己的需要，修改 Adapter 为任何类名和路径，但是一定要继承自 AuthorizeAdapter。*当然，如果您不想修改授权页面，可以不设置这个 Adapter。

2) 创建 MyAdapter 这个类，然后添加自定义代码，如：

```
public class MyAdapter extends AuthorizeAdapter {
    public void onCreate() {
        System.out.println("> ShareSDKUIShell created!");
    }
}
```

```

        System.out.println("> PlatName: " + getPlatformName ());
        System.out.println("> TitleLayout: " + getTitleLayout());
        System.out.println("> WebBody: " + getWebBody());
        getTitleLayout().getTvTitle().setText("This is MyAdapter");
    }

    public void onDestroy() {
        System.out.println("> ShareSDKUIShell will be destroyed.");
    }
}

```

上面的 **onCreate**、**onResume**、**onStop**、**onDestroy** 等方法都在 **ShareSDKUIShell** 的生命周期不停时段被调用，如 **onCreate**、**onDestroy** 分别在授权页面被创建（但还没有显示），以及即将被关闭时被调用。也就是说可以通过这些方法参与授权页面的生命周期。而 **getPlatformName**、**getTitleLayout** 和 **getWebBody** 分别返回授权平台的名称、授权页面标题栏控件和授权页面主体部分的 **WebView**，开发者可以通过平台名称和控件实例，修改整个页面的布局甚至行为。更多关于自定义授权页面的案例，可以参考 ShareSDK BBS 中的[相关主题](#)。

请注意，注册在 **Adapter** 下的 **AuthorizeAdapter** 子类不能被混淆！否则 **ShareSDK** 无法找到这个类。

Single Sign-On

新浪微博、腾讯微博、QQ 空间、人人网、Facebook 和 Dropbox 已经提供了 **Single Sign-On (SSO)** 的授权方式。就是利用这些平台的手机客户端来完成授权。由于 SSO 的授权方式对于用户来说更加便捷，因此各大平台均建议开发者优先使用这一种授权方式。**ShareSDK** 提供 SSO 的授权实现，并

*且默认情况下是使用的。*但是如果您想关闭 SSO 功能，可以调用类似于下面的代码来关闭：

```
Facebook facebook = new Facebook(context);
facebook.SSOSetting(true); // true 表示不使用 SSO 方式授权
facebook.setPlatformActionListener(paListener);
facebook.authorize();
```

为方法 SSOSetting 传递参数 true，表示不使用 SSO 方式授权。

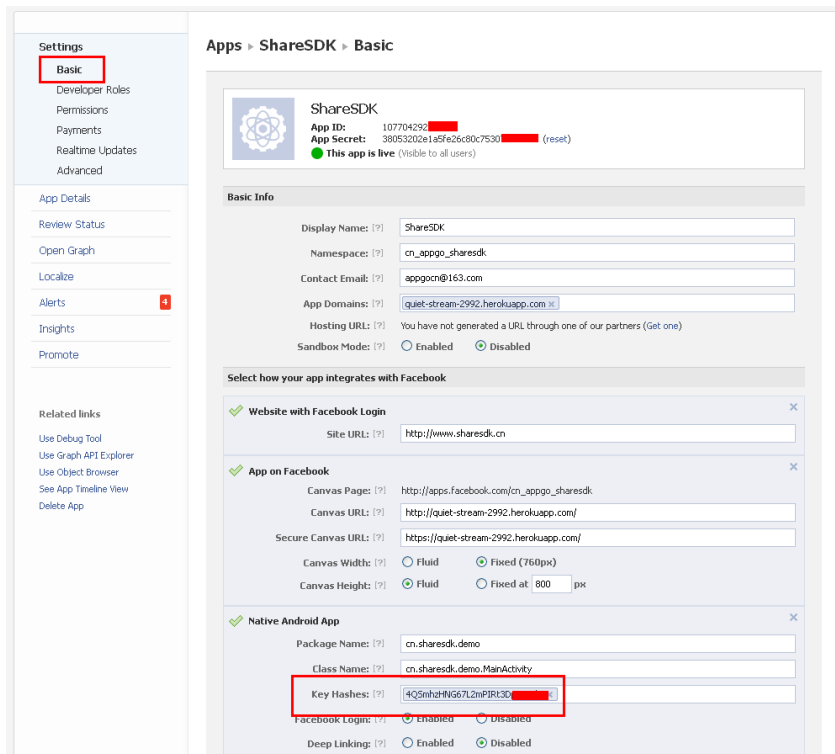
新浪微博

新浪官方微博客户端自版本 4.0 以后升级了 SSO 授权验证机制，会在授权的时候验证第三方应用的包名与签名的有效性。这些数据会在开放平台上注册应用时要求开发者填写，操作步骤类似于微信开放平台上的操作，可以参考[前文对应的说明](#)，或者阅读[新浪微博官方文档](#)的说明。

为了简化开发者的集成步骤，ShareSDK 已经将原来的“com.sina.sso.RemoteSSO.aidl”文件集成到其对应的平台 jar 包中，开发者不需要再复制此包到自己的 src 里，但是**混淆脚本里面依然需要 keep 这个类**。

Facebook

Facebook 的 SSO 要求您到自己的应用详情页面中，如上文签名类似，*注册您的包名和签名* SHA 值的 Base64 编码结果。具体流程如下：



1) 如上图。在 facebook 开发者平台注册、登录。然后进入您的应用详情页面。在应用配置信息界面中选择左侧的“Settings” — “Basic”。

2) 参考 facebook [开发文档](#)中的示例代码，计算出您签名 keystore 文件 SHA 值的 Base64 编码结果。

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    try {
        PackageInfo info = getPackageManager().getPackageInfo(
            "com.facebook.samples.loginhowto",
            PackageManager.GET_SIGNATURES);
        for (Signature signature : info.signatures) {
            MessageDigest md = MessageDigest.getInstance("SHA");
            md.update(signature.toByteArray());
            Log.d("KeyHash:", Base64.encodeToString(md.digest(), Base64.DEFAULT));
        }
    } catch (NameNotFoundException e) {

    } catch (NoSuchAlgorithmException e) {

    }
    ...
}

```

3) 回到应用详情的右侧，填写“Native Android App”项目中的应用包名，类名，和应用签名的 hash 值并保存。

更多详细的信息，可以参考文档 [“Facebook Login Flow for Android”](#)。为了演示方便，ShareSDK 发布的时候，其 Sample 项目已经在 Facebook 上完成了上述的注册，如果您需要调试 Sample 项目的 Facebook 功能，请使用 Sample 目录下的 ***demokey.keystore*** 来给 Sample 签名。

Dropbox

Dropbox 的 SSO 也有额外的要求。需要如下面示例，对 ShareSDKUIShell 添加一个 intent-filter:

```

<activity
    android:name="cn.sharesdk.framework.ShareSDKUIShell"
    android:theme="@android:style/Theme.Translucent.NoTitleBar"
    android:configChanges="keyboardHidden|orientation|screenSize"
    android:windowSoftInputMode="stateHidden|adjustResize" >

```



```
<!--
    Dropbox 的 SSO 功能需要在此处添加一个对
    ACTION_VIEW 事件的过滤器，其中的 scheme 是
    “db-” 前缀再开发者应用的加上 appKey。如果
    此过滤器不设置，则不能完成 SSO 功能授权
-->
<intent-filter>
    <data android:scheme="db-7janx53ilz11gbs" />
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.BROWSABLE"/>
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter>
</activity>
```

其中的 scheme 需要根据您的应用信息作修改，其他的内容无须改变。

SSO 只是授权方式的区别，不管是授权数据的管理或者是后续的其他操作，由于都与授权方式无关，因此不受 SSO 影响。

获取用户资料及第三方登录

如果您想获取用户资料，您可以参考下面的代码实现：

```
String account = "3189087725";
Platform weibo = ShareSDK.getPlatform(context, SinaWeibo.NAME);
weibo.setPlatformActionListener(paListener);
weibo.showUser(account);
```

示例代码获取了新浪微博的实例，然后调用 showUser 方法

获取了账号为“3189087725”用户的资料，其结果将通过操作回调 `paListener` 返回给外部代码。如果 `account` 为 `null`，则表示获取授权账户自己的资料。

获取用户资料的最主要用途是实现**第三方平台登录**的功能。一般来说，如果您的应用已经拥有自己的登录/注册功能了，但是您还希望提供用户一种利用已有的微博等第三方平台的账号快速登录到您的系统，那么您可以选择简单的“**授权-登录**”，或者“**获取用户资料-注册-登录**”。

第一种方法会让您的系统一直依赖第三方平台，其操作方式如下：

- 1、点击您应用的“登录”按钮
- 2、调用 `authorize` 引导用户授权
- 4、成功使用 `getDb().getUserId()` 来获取此用户在此平台上的 `id`
- 5、如果 `id` 不为空，就视为用户已经登录

而第二种方法您的应用需要有自己的账号系统。操作如下：

- 1、点击您应用的“登录”按钮
- 2、通过用户指定的平台，使用 `getDb().getUserId()` 来得到用户在此平台上的 `id`
- 3、如果 `id` 不为空，则提交给您的登录接口，否则调用 `showUser` 请求用户的资料
- 4、服务器接收到 `id` 以后判断用户是否已经注册，若已注册，认为登录成功，否则引导客户端进入注册流程
- 5、客户端进入注册流程以后，将从 `showUser` 得到的资料填写到注册页面，用户完善资料以后，将其 `id` 和资料一并提交给您应用的服务器
- 6、如果注册成功，引导用户进入客户端应用

获取资料前 ShareSDK 会自行判断平台是否已经授权, 若未授权, 会自行执行授权操作。

分享到指定平台

ShareSDK 的分享功能由 Platform 的 share(ShareParams params)方法完成上。请参考 [Wiki 相关文档](#) 上的说明。 下面分别使用新浪微博和 QQ 空间作为例子, 举例如何利用此方法实现分享功能。

1) 新浪微博:

```
Platform.ShareParams sp = new SinaWeibo.ShareParams();
sp.text = "测试分享的文本";
sp.imagePath = "/mnt/sdcard/测试分享的图片.jpg";

Platform weibo = ShareSDK.getPlatform(context, SinaWeibo.NAME);
weibo.setPlatformActionListener(paListener); // 设置分享事件回调
// 执行图文分享
weibo.share(sp);
```

2) QQ 空间

```
QZone.ShareParams sp = new QZone.ShareParams();
sp.title = "测试分享的标题";
sp.titleUrl = "http://sharesdk.cn"; // 标题的超链接
sp.text = "测试分享的文本";
sp.imageUrl = "http://www.someserver.com/测试图片网络地址.jpg";
sp.comment = "我对此分享内容的评论";
sp.site = "发布分享的网站名称";
sp.siteUrl = "发布分享网站的地址";
```

```
Platform qzone = ShareSDK.getPlatform (context, QZone.NAME);  
qzone. setPlatformActionListener (paListener); // 设置分享事件回调  
// 执行图文分享  
qzone.share(sp);
```

ShareParams 是 Skare SDK 实现分享的数据体，Platform.ShareParams 本身提供的字段十分有限，因此不是建议使用的数据体。ShareSDK 为每一个社交平台提供一个字段更丰富的数据体，可以通过“平台名称.ShareParams”的方式初始化，如上面例子中的“SinaWeibo.ShareParams”和“QZone.ShareParams”。

执行分享前 ShareSDK 会自行判断平台是否已经授权，若未授权，会自行执行授权操作。

分享时需要注意一些常见的问题：**绝大部分的微博平台：如新浪微博、腾讯微博、网易微博等都是不支持短时间内重复分享相同文本内容的。**因此如果分享的文本重复，会有错误返回。

新浪微博

新浪微博支持分享网络图片，但是这个需要高级微博写入权限，因此如果您需要分享网络图片，请申请权限以后，将图片 Url 设置给 SinaWeibo.ShareParams.imageUrl 即可。下面详细描述一下申请此权限的方法：

- 1) 进入您的应用详情，并选择“接口管理” — “申请权限”。
- 2) 在右侧展开的页面中展开“微博高级写入接口”，会看到“statuses/upload_url_text”。勾选分组左侧的复选框。
- 3) 滚动页面到底部，填写申请理由，然后“提交申请”。



完成上述步骤已经，您的申请就会进入新浪微博的审核程序，一般 3 个工作日以后可以得到审核结果。当接口权限被审核通过，您的应用就具备发送网络图片的功能了。

人人网和 QQ 空间

与此同时，**人人网和 QQ 空间已经支持分享本地图片功能，可以通过 `imagePath` 字段为其设置本地图片**。不过这些平台都需要特定的权限才能分享本地图片，因此请确保您的应用已经申请了足够的权限。请注意：对于人人网和 QQ 空

间来说, *imageUrl* 存在的时候, 原来的 *imagePath* 将被忽略, 但是新浪微博刚好相反, 故须要特别注意。

微信和其微信朋友圈

微信和微信朋友圈 的分享则要求在您的项目中注册它的一个回调 **Activity**, 并将回调 **Actiity** 添加到您的代码中。其操作如下:

1) 打开 **AndroidManifest.xml**, 参考配置 **AndroidManifest.xml** 章节的说明, 配置回调 **WXEntryActivity**。

2) 复制 SDK 解压目录下 **Src** 中的 **wxapi** 包到您的项目中。

大部分平台分享时可以@此平台上的其他用户, 但不同平台关注时需要提供的字段和权限并不一样。比如新浪微博@其他人时提供的 **account** 是其 **screen_name** (即显示出来的名称), 腾讯微博提供的是用户名, **twitter** 和 **facebook** 都使用 **uid**, 但 **facebook** 需要在 **uid** 前后加中括号 (如: **uid** 是“123”, 则@他人时提供的是 “[123]”)。而且 **facebook** 比较麻烦, 还需要申请权限, 否则@操作无效。

印象笔记

由于部分开发者要求印象笔记在目标设备已经安装了印象笔记客户端的时候, 直接使用印象笔记客户端来分享, 因此最新版本的 **ShareSDK** 已经提供了此项支持。开发者可以通过到印象笔记的应用信息中修改字段 “**ShareByAppClient**” 的值来实现: **true**, 表示优先使用客户端分享。

不过 **ShareByAppClient** 只会印象分享方式, 而不能影响授权方式, 印象笔记暂无 **SSO** 功能, 因此如果您使用印象笔记完成授权, 依然会跳往 **Web** 授权方式。

ShareParams 字段释义

ShareSDK 的每一个平台都有自己的 ShareParams，由于平台差异，不同平台之间的 ShareParams 字段数量不一样，但是同样名称的字段都表示相同的含义，下面是当前 ShareSDK 所有平台 ShareParams 的字段名称和代表的含义：

字段名称	描述
text	待分享的文本
imagePatd	待分享的本地图片。如果目标平台使用客户端分享，此路径不可以在/data/data 下面
filePatd	待分享的文件路径。这个用在 Dropbox 和微信中
title	分享内容的标题
notebook	存放笔记的笔记本，如果不存在，会创建。一般用在印象笔记等“笔记类”平台中
stack	印象笔记中的字段，用于归类笔记本
publish	印象笔记中的字段，用来规定笔记是否公开
resource	印象笔记中的字段，数组，暂时支持 String 类型和 File 类型的元素。如果想一次发送多个文本和图片，需要使用这个字段
tags	标签，数组，部分平台支持为分享内容设置标签，可以使用这个字段
isPublic	flickr 的字段，表示是否公开
isFriend	flickr 的字段，表示是否公开给自己的朋友
isFamily	flickr 的字段，表示是否公开给自己的家人
safetyLevel	flickr 的字段，表示安全级别：1 为安全级、2 为辅导级、3 为限制级
contentType	flickr 的字段，表示相册类型：1 为相片、2 为屏幕截图、3 为其他
hidden	flickr 的字段，表示是否隐藏图片
venueName	foursquare 的字段，表示分享位置的名称
venueDescription	foursquare 的字段，表示分享位置的描述
latitude	分享位置的维度
longitude	分享位置的经度

imageUrl	待分享的网络图片
comment	对分享内容的评价。区别于 text ，评论一般共应用的用户自己填写，部分平台支持此字段
titleUrl	分享内容标题的链接地址
url	分享内容的 url、在微信和易信中也使用为视频文件地址
address	邮箱地址或者短信电话号码，一般在邮箱或者短信中使用
site	QQ 空间的字段，标记分享应用的名称
siteUrl	QQ 空间的字段，标记分享应用的网页地址
friendsOnly	VK 的字段，表示是否只有朋友可见
gropuld	VK 的字段，表示图片所属的组 ID。如果不设置此字段，ShareSDK 会将组 ID 设置为 ShareSDK 所在的组
extInfo	微信的字段，分享应用时，可以选择分享二进制文件或者脚本，此字段用来设置分享应用中的脚本
shareType	微信和易信的字段，分享内容的类型：分别为 Platform.SHARE_TEXT（分享文本），Platform.SHARE_IMAGE（分享图片），Platform.SHARE_WEBPAGE（分享网页，既图文分享），Platform.SHARE_MUSIC（分享音频），Platform.SHARE_VIDEO（分享视频），Platform.SHARE_APPS（分享应用，仅微信支持），Platform.SHARE_FILE（分享文件，仅微信支持）Platform.SHARE_EMOJI（分享表情，仅微信支持）
musicUrl	微信和易信的字段，分享音频时的音频文件网络地址
imageData	微信和易信的字段，各类分享内容中的图片 bitmap 对象，可以替代 imagePath 或者 imageUrl
author	有道云笔记的字段，表示分享内容的作者

上述字段中和本地路径有关的，全部使用“Path”结尾，如 imagePath；和网络路径有关的，使用“Url”结尾，如

imageUrl。本地图片最好使用后缀名，否则 **POST** 文件时，可能无法正确计算文件的 **MIME** 值，从而导致发送失败。

接收微信客户端消息

ShareSDK 支持让您的应用接收到微信和易信客户端发送过来的消息，但由于当前易信在此功能上还没有提供实际入口（虽然保留了接口），因此本章节仅描述第三方应用接收微信发送过来的请求 app 消息和应用发起的信息：



上面的图片展示的分别是从小聊天页面打开 ShareSDK 的 Sample 和通过点击微信上的“分享应用”信息，让 ShareSDK 的 Sample 执行一个脚本（弹出一个 Toast 提示）。

为了完成上面的操作，您需要在原来“`.wxapi.WXEntryActivity`”中重载两个方法：

名称	说明
<code>onGetMessageFromWXReq</code>	此方法用于处理微信发出的向第三方应用请求 <code>app message</code>
<code>onShowMessageFromWXReq</code>	此方法用于处理处理微信向第三方应用发起的消息

比如处理微信向第三方应用发出请求 `app message` 这种类型的消息，在 `WXEntryActivity` 中重写方法 `onGetMessageFromWXReq`。然后添加当微信用户点击聊天界面中您的应用的图标的时候应该执行的代码（如打开您应用中的某一个页面），于是微信客户端会在适当的时候执行这些代码，即使您的应用从来都没有启动过。

下面是上面第二个图片（向第三方应用发起的消息）的示例代码：

```
public void onShowMessageFromWXReq(WXMediaMessage msg) {  
    if (msg == null || msg.mediaObject == null) {  
        return;  
    }  
    if (!(msg.mediaObject instanceof WXAppExtendObject)) {  
        return;  
    }  
    WXAppExtendObject obj = (WXAppExtendObject) msg.mediaObject;  
    Toast.makeText(this, obj.extInfo, Toast.LENGTH_SHORT).show();  
}
```

代码中的各个对象的说明可以在[微信的开发文档](#)中查阅。此方法会在用户点击“分享应用”的消息时被触发，可以执行

的代码几乎不受限制，但是本例仅仅只是弹出一个 Toast 的提示而已。

关注指定用户

当前的 ShareSDK 已经实现了 *新浪微博* 和 *腾讯微博* 的关注功能，下面的代码演示腾讯微博的关注操作：

```
String account = "shareSDK";
Platform weibo = ShareSDK.getPlatform(
    context, TencentWeibo.NAME);
weibo.setPlatformActionListener(paListener);
weibo.followFriend(account);
```

如果重复关注。paListener 会有错误回调，因此最好处理此异常，提示“重复关注”，或者直接提示“关注成功”。

执行关注前 ShareSDK 会自行判断平台是否已经授权，若未授权，会自行执行授权操作。

获取关注列表

因为平台的差异，ShareSDK *当前仅为有限的平台提供获取关注列表的功能，这些平台包括：新浪微博、腾讯微博、Facebook 和 Twitter*（其中 Facebook 获取的是好友列表）。下面的代码演示获取授权账户在新浪微博上的 *第一页* 关注列表：

```
SinaWeibo weibo = new SinaWeibo(v.getContext());
weibo.setPlatformActionListener(new PlatformActionListener() {
    public void onError(Platform weibo, int action, Throwable t) {
        t.printStackTrace();
    }
});
```

```

        if (action == Platform.ACTION_GETTING_FRIEND_LIST) {
            // 在这里处理获取关注列表失败的代码
        }
    }

    public void onComplete(Platform weibo, int action,
        HashMap<String, Object> res) {
        if (action == Platform.ACTION_GETTING_FRIEND_LIST) {
            // 在这里处理获取关注列表成功的代码
        }
    }

    public void onCancel(Platform weibo, int action) {
        if (action == Platform.ACTION_GETTING_FRIEND_LIST) {
            // 在这里处理取消获取关注列表的代码
        }
    }
}

weibo.listFriend(50, 0, null); // 获取授权账号的列表则传递 null

```

关注列表的接口是分页的，不建议每一页请求太多项目，因为部分平台只能支持 30 个左右的分页。**分页的页码从 0 开始。**

自定义接口的使用

ShareSDK 在默认情况下只提供授权、用户获取资料、分享等功能，在部分平台会提供获取时间线、获取关注或好友列表、执行关注等等操作。但这些功能远不能满足所有的开发者。为了更灵活适应开发者的需求，ShareSDK 由提供了自定义接口的功能。利用此接口，可以比较便利地根据不同平

台 API 文档实现适合自己的接口。下面用“QQ 空间发表日志”为例子解释此接口的使用：

```
QZone qzone = new QZone(getContext());
qzone.setPlatformActionListener(this);

String url = "https://graph.qq.com/blog/add_one_blog";
String method = "POST";
short customerAction = ACTION_QZONE;
HashMap<String, Object> values = new HashMap<String, Object>();
values.put("title", getContext().getString(R.string.customer_qzone));
values.put("content", getContext().getString(R.string.blog_sample));

qzone.customerProtocol(url, method, customerAction, values, null);
```

上面的代码第一行和第二行是构造一个 QQ 空间的实例，并为之设置操作回调（PlatformActionListener）。接下来为自定义接口设置参数，包括此 API 的 url，接口调用方式（GET 或 POST），为接口定义一个回调中可以自己识别的事件代码（ACTION_QZONE），以及接口参数列表。最后一行代码是执行自定义接口，其参数列表为：

名称	类型	说明
url	String	方法所属平台目标接口的 url
method	String	接口的请求方式，可以是“GET”或者“POST”
customerAction	short	自定义的用于自己识别事件的代码，此事件代号会通过 PlatformActionListener 的回调方法返回，通过将方法的 action 参数和 CUSTOMER_ACTION_MASK 作与操作，可以得到来自自定义的事件代号

values	HashMap <String, Object>	目标接口要求的参数列表，若无须发送参数，则给予 null 或者空表即可
filePathes	HashMap <String, Object>	对于需要发送二进制文件的接口，可以通过此字段携带文件的字段名称和本地路径提交，若无须发送文件，则给予 null 或者空表即可

ShareSDK 已经为给授权方法提供自定义接口，因此*开发者可以使用新的授权方法 **platform.authorize(String[])**来自定义授权的权限*，其中的 String[] 参数是权限列表。但是请注意，并不是所有平台都接受权限设置，具体需要参考具体平台的 API 文档说明。

然后请注意上文中的“customerAction”。ShareSDK 为每一个操作都提供一个操作代码，如“分享”事件的操作代码是“ACTION_SHARE”。开发者自定义事件无法直接内置一个操作代码，因此这个接口的操作事件是开发者自己指定的。但是这有限制，开发者可以指定的事件只能是一个 short 类型的代码，因为此代码进入 ShareSDK 内部以后会被与一个内部代码 Platform.ACTION_CUSTOMER（其值为 0xa0000）作或操作，得到一个随事件流移动的操作代码，并会在最后的操作回调中给到开发者。这就是说，*开发者将来在 Platform ActionListener 中得到的操作代码，不是他一开始设置进去的，需要将得到的 action 和 Platform.CUSTOMER_ACTION_MASK 作一次与操作，才能得到最初自己传递进去的操作代码。*

最后说一下此接口的支持情况。考虑到部分平台无所谓“自定义接口”，而另外一些平台自定义接口实现起来会十分复杂，所以 **ShareSDK 并没有为所有平台提供此接口**。现在已实现此接口的平台包括：**豆瓣、Facebook、开心网、网**

易微博、人人网、新浪微博、搜狐微博、QQ 空间、腾讯微博、Twitter 和有道云笔记。

平台数据库的操作

ShareSDK 将每一个平台的授权数据都缓存在 *SharedPreferences* 中，并且相互独立。如果您希望获取任何一个平台的授权数据，可以像下面这样子做：

```
Platform qzone = ShareSDK.getPlatform(this, QZone.NAME);
String accessToken = qzone.getDb().getToken(); // 获取授权 token
String openId = qzone.getDb().getUserId(); // 获取用户在此平台的 ID
String nickname = qzone.getDb().get("nickname"); // 获取用户昵称
// 接下来执行您要的操作
```

不同平台都会缓存 `accessToken`、`expiresIn`、`UserId`、`nickname` 等信息，分别表示授权的 Token、授权 Token 的有效时间、用户在此平台上的 ID、用户的昵称等信息。可以通过 `getToken()` 或者 `setToken()` 进行操作。此外，开发者还可以通过直接 `get()` 某个字段的办法，获取其数据，如上面通过“`nickname`”字段获取到用户的昵称。

ShareSDK 同时还允许开发者使用“*exportData*”和“*importData*”两个方法，批量导出和导入 *PlatformDb* 中的数据。开放这两个方法的目的是：部分应用具备多用户系统，如果同一设备上不同时期要登录多个账户，那么他们需要备份上一个用户的资料。ShareSDK 并不设置多用户系统，但是用户可以通过登录不同用户的时候，批量导出旧用户的资料，然后再登录新用户，直到新用户重新登录的时候，重新导入其数据的方式，实现其多用户系统功能。

由于 ShareSDK 所有的授权数据都存在 PlatformDb 中，所以“用户是否”授权，和“取消授权（清除授权信息）”的操作依据其实 PlatformDb。*在 ShareSDK 中，判断此平台是否授权的方法是 isValid，而取消授权的方法是 removeAccount*，下面的代码演示客户端判断是否已经授权，如果授权就删除授权资料，否则就执行授权：

```
Platform qzone = ShareSDK.getPlatform(this, QZone.NAME);
If (qzone.isValid ()) {
    qzone.removeAccount ();
}
qzone.setPlatformActionListener(paListener);
qzone.authorize();
```

isValid 和 removeAccount 不开启线程，会直接返回。

快捷分享

本章内容包括：

- 什么是快捷分享
- 代码配置
- 自定义外部回调
- 为不同平台定义差别化分享内容
- 在九宫格中添加自定义图标
- 其他自定义方式

什么是快捷分享

快捷分享是 ShareSDK 提供的一套基于其接口的 GUI。通过简单的配置，可以在不考虑 ShareSDK 具体分享操作的情况下，调用很少的代码，就完成分享操作。

代码配置

集成快捷分享的步骤可以参考**导入 SDK** 章节的说明。下面主要介绍如何调用快捷分享：

1) 首先也是最重要的，和直接调用接口一样，需要在启动快捷分享前，添加 `initSDK` 的代码。

2) 在您准备触发分享的地方（如 `OnClick` 方法里面），添加如下代码：

```
OnekeyShare oks = new OnekeyShare();

// 分享时 Notification 的图标和文字
oks.setNotification(R.drawable.ic_launcher,
    getContext().getString(R.string.app_name));
// address 是接收人地址，仅在信息和邮件使用
oks.setAddress("12345678901");
// title 标题，印象笔记、邮箱、信息、微信、人人网和 QQ 空间使用
oks.setTitle(getContext().getString(R.string.share));
// titleUrl 是标题的网络链接，仅在人人网和 QQ 空间使用
oks.setTitleUrl("http://sharesdk.cn");
// text 是分享文本，所有平台都需要这个字段
oks.setText(getContext().getString(R.string.share_content));
// imagePath 是图片的本地路径，Linked-In 以外的平台都支持此参数
```

```
oks.setImagePath(MainActivity.TEST_IMAGE);
// imageUrl 是图片的网络路径，新浪微博、人人网、QQ 空间、
// 微信、易信、Linked-In 支持此字段
oks.setImageUrl("http://sharesdk.cn/ rest.png");
//微信、易信中使用，表示视屏地址或网页地址
oks.setUrl("http://sharesdk.cn");
// appPath 是待分享应用程序的本地路劲，仅在微信中使用
oks.setAppPath(MainActivity.TEST_IMAGE);
// comment 是我对这条分享的评论，仅在人人网和 QQ 空间使用
oks.setComment(getContext().getString(R.string.share));
// site 是分享此内容的网站名称，仅在 QQ 空间使用
oks.setSite(context.getString(R.string.app_name));
// siteUrl 是分享此内容的网站地址，仅在 QQ 空间使用
oks.setSiteUrl("http://sharesdk.cn");
// venueName 是分享社区名称，仅在 Foursquare 使用
oks.setVenueName("Southeast in China");
// venueDescription 是分享社区描述，仅在 Foursquare 使用
oks.setVenueDescription("This is a beautiful place!");
// latitude 是维度数据，仅在新浪微博、腾讯微博和 Foursquare 使用
oks.setLatitude(23.122619f);
// longitude 是经度数据，仅在新浪微博、腾讯微博和 Foursquare 使用
oks.setLongitude(113.372338f);
// 是否直接分享（true 则直接分享）
oks.setSilent(silent);
// 指定分享平台，和 slient 一起使用可以直接分享到指定的平台
if (platform != null) {
    oks.setPlatform(platform);
}
// 去除注释可通过 OneKeyShareCallback 来捕获快捷分享的处理结果
// oks.setCallback(new OneKeyShareCallback());
//通过 OneKeyShareCallback 来修改不同平台分享的内容
```

```
oks.setShareContentCustomizeCallback(  
    new ShareContentCustomizeDemo());  
  
oks.show(context);
```

请注意，上面代码中的参数是快捷分享 *可以接受的所有参数* 列表，如果您的项目并不集成 ShareSDK 的所有平台，并 *没有必要* 像上面的代码那样传递所有参数进快捷分享。所以请根据您的项目特点，去除不必要的参数。*具体平台参数说明，可以参考 [API 文档](#)*。

3) 如果您的项目中集成微信平台，请将 SDK 解压目录中的 “*Src/wxapi*” 目录复制到您的项目主包下。如果您的项目中集成易信平台，请将 SDK 解压目录中的 “*Src/yxapi*” 目录复制到您的项目主包下。

完成上述的步骤，快捷分享就已经集成了。

自定义外部回调

新版本的快捷分享添加了一个 callback 的 Extra，*可以在启动快捷分享的时候，设置一个 callback 给它，于是快捷分享操作结束以后，不再调用自己的 callback，而是调用从外部设置进来的 callback*。设置方法和其他的 extra 类似：

```
OnekeyShare oks = new OnekeyShare();  
// 参考代码配置章节，设置分享参数  
// 设置自定义的外部回调  
oks.setCallback(new OneKeyShareCallback());  
oks.show(context);
```

上述代码中的 *OneKeyShareCallback* 是一个实现了 Platform

ActionListener 的类，因此它有 `onComplete`、`onError` 和 `onCancel` 等方法。启动快捷分享的时候将 `OneKeyShareCallback` 的类名传递进去，快捷分享自己会尝试创建其实例，如果创建失败或者没有传递 `callback` 字段，则使用默认的 `callback`，如果成功，则以后会将分享结果给这个类处理。

为不同平台定义差别化分享内容

快捷分享允许开发者设置了统一的分享模板以后，为不同平台设置差别化的分享内容。为了实现这个功能，需要为快捷分享提供一个 `ShareContentCustomizeCallback` 的实例，并参考下面的伪代码设置到快捷分享中：

```
OnekeyShare oks = new OnekeyShare();
// 参考代码配置章节，设置分享参数
//通过 OneKeyShareCallback 来修改不同平台分享的内容
oks.setShareContentCustomizeCallback(
    new ShareContentCustomizeDemo());
oks.show(context);
```

`ShareContentCustomizeCallback` 是一个接口，使用时需要实现其 `onShare(Platform, ShareParams)` 方法。此方法被回调的时候会传递一个即将执行分享的平台实例和即将被分享的分享内容（`ShareParams`）。开发者可以通过修改这个 `ShareParams` 参数完成分享内容根据平台的差异化。

在九宫格中添加自定义图标

快捷分享的九宫格列表提供添加自定义图标的功能，方

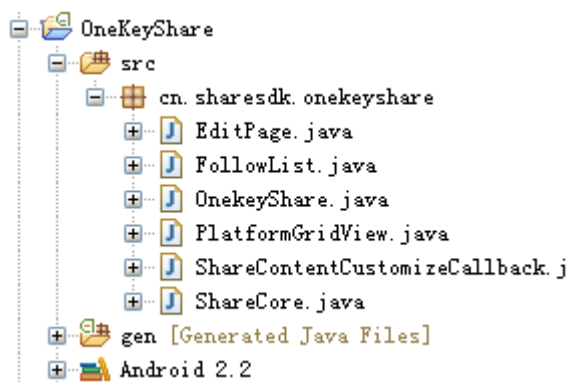
法如下：

```
OnekeyShare oks = new OnekeyShare();
// 参考代码配置章节，设置分享参数
// 构造一个图标
Bitmap logo = BitmapFactory.decodeResource(getResources(), R.drawable.logo);
// 定义图标的标签
String label = getResources().getString(R.string.app_name);
// 图标点击后会通过 Toast 提示消息
OnClickListener listener = new OnClickListener() {
    public void onClick(View v) {
        Toast.makeText(getContext(), "Hello", Toast.LENGTH_SHORT).show();
        oks.finish();
    }
};
oks.setCustomerLogo(logo, label, listener);
oks.show(context);
```

这样的代码可以多次使用，以添加任意数量的自定义图标。快捷分享默认会将自定义图标**放在九宫格的最后位置**，如果您需要放在它在其他地方，可以到 PlatformGridView.PlatformAdapter 中修改代码。

其他自定义方式

因为通用性，所以快捷分享不会提供太多涉及平台差异的高级功能。不过快捷分享是开源的，**其源码放在 OneKeyShare 项目里面**，开发者可以通过修改这些源码强化快捷分享的功能。



随着 ShareSDK 的不断优化和壮大,新平台加入的同时我们还在强化已有平台的功能。比方说微信及微信朋友圈分享时支持分享 **Bitmap** 对象、印象笔记分享时支持多图文同时分享等等。这些功能只是部分平台特有的,不会集成到快捷分享中。如果您的项目需要这些功能,需要修改快捷分享的代码,或者直接参考**接口调用**章节的相关描述,调用接口来实现。

插件服务

本章内容包括：

- 插件服务的定义
- 创建插件服务
- 评论与赞服务

插件服务的定义

当下 ShareSDK 已经具备了一定的技术、数据和服务功能，为了让集成开发者可以更好地在 ShareSDK 上进行后续的开发，ShareSDK 开放了一个名为“插件服务”的框架。开发者可以通过继承 ShareSDK 内部的一个“服务”基类后，利用 ShareSDK 的资源，结合您自己的代码，对 ShareSDK 进行二次开发，创建更强大的功能。本章将介绍如何创建一个插件服务并，结合 ShareSDK 的“评论与赞”服务，描述其使用方法。

创建插件服务

ShareSDK 的插件服务都集成自 `cn.sharesdk.framework.Service`，至于服务的名字和用途并没有特别的限制，只是在混淆的时候，这个类需要被 `keep` 下来，因为 ShareSDK 内部需要用到这个类的方法。下面是一个空的服务：

```
public class MyService extends Service {
    private static final int VERSION_INT = 1;
    private static final String VERSION_NAME = "1.0.0";

    public MyService(Context context) {
        super(context);
    }

    public void onBind() {
        // 当 Service 被注册到 ShareSDK 后，此方法会被调用
    }
}
```

```
public void onUnbind() {  
    // 当 Service 被反注册到 ShareSDK 前，此方法会被调用  
}  
  
protected int getServiceVersionInt() {  
    return VERSION_INT;  
}  
  
public String getServiceVersionName() {  
    return VERSION_NAME;  
}  
  
}
```

从上面的实例代码可以看出，ShareSDK 对插件服务的约束很少，只是要求子类实现 `getServiceVersionInt` 和 `getServiceVersionName` 两个方法而已，因为 ShareSDK 内部通信协议中需要用到这两个字段。其他的方法，如 `onBind` 会在插件服务被实例化并注册到 ShareSDK 以后调用一次，而 `onUnbind` 相反，在从 ShareSDK 反注册之前被调用。**ShareSDK 建议开发者不要修改插件服务的构造方法**，虽然这个方法现在和 `onBind` 在同一个线程中被执行，但是后续版本的 ShareSDK，考虑到效率问题，可能会将之移动到另外一个子线程中，因此原来的初始化代码就可能因此失效。如果您需要在服务被注册到 ShareSDK 以后执行一些初始化操作，请在 `onBind` 方法中完成。

虽然插件服务提供了公开的构造方法，但是不应该使用这个方法来实例化一个插件服务，而应该调用 `ShareSDK.registerService` 方法，传递插件服务的类来完成注册。如：

```
ShareSDK.registerService(MyService.class);
```

ShareSDK 会在 `registerService` 方法内部实例化插件服务，并以单例模式保存这个实例，直到 `ShareSDK.unregisterService` 方法被调用后，此实例会被反注册并在随后被回收。

使用某个插件服务之前，ShareSDK 要求此服务必须已经被注册，但是插件服务的反注册操作并不是必须的，即使一个服务从来没有被显式反注册，当 `ShareSDK.stopSDK` 方法被调用的时候，所有注册到 ShareSDK 的服务都会被反注册。

ShareSDK 允许开发者多次注册同一个插件服务，但 ShareSDK 内部只认第一次注册的服务，后续的注册操作都会失效。须要注意的时候，***registerService 和 unregisterService 两个方法都应该在 ShareSDK.initSDK 和 ShareSDK.stopSDK 两个方法之间调用***，离开 ShareSDK 的生命周期，插件服务是没有意义的。

使用服务的时候，代码和实例化一个 Platform 类似，但是更加方便：

```
MyService service = ShareSDK.getService(MyService.class);
```

也就是说，获取插件服务以后，不需要执行强制转换（感谢 Java 的泛型功能，让 ShareSDK 能够内部完成了强制转换）。

更多插件服务的详细接口说明，请参考我们的 [Wiki](#) 和 [Api 文档](#)。

评论与赞服务

评论与赞是 ShareSDK 提供的一个插件服务，它提供了一

些简单的插件，开发者可以通过代码或者 xml 的方式将其集成到自己的应用中，使用的时候调用控件的方法设置一些简单参数，就可以完成一个类似于论坛的主题评论功能。同时，如果开发者不喜欢 ShareSDK 提供的控件和 UI 效果，还可以自己编写 UI，然后调用插件服务类 Socialization 里面的接口，直接与 ShareSDK 的服务器沟通。

代码配置

首先，请参考**导入 Share SDK**的说明导入 Socialization 项目，并将您的项目依赖此项目。

其次，和使用其他插件服务类似，评论与赞功能之前，也需要注册此服务，可以在 initSDK 后，调用下面的代码：

```
// 注册社会化组件服务，此服务用于“评论与赞”功能的演示
ShareSDK.registerService(Socialization.class);
```

请注意，此方法必须在包含评论与赞控件的 UI 初始化之前调用，比方说如果您的 activity 的 layout 里面集成了 QuickCommentBar（快速评论工具栏），那么就需要在 activity 的 setContentView 方法之前调用上面的注册代码，因为 QuickCommentBar 初始化以后就会立刻通过 Socialization 来请求数据了。

完成注册以后，您有两个方法使用评论与赞功能：现成的控件和直接调用网络接口。下面用两个小节分别介绍这两种方法。

通过 GUI 控件集成服务

使用 GUI 控件的好处在于可以不理睬插件服务与

ShareSDK 后台复杂的通信协议、甚至很多和评论与赞功能相关的逻辑都可以不需要理会，只需要添加少量的代码完成配置，其他的事情就由 GUI 控件自己完成了。但它的坏处也很明显，就是自定义能力不强。

ShareSDK 为开发者提供了两个 GUI 控件，分别是快速评论工具栏和跟帖列表页面。其中的工具栏提供了对一个主题的评论、（微博等的）转发和称赞功能。跟帖列表是一个 FakeActivity，即一个页面，完整展示一个主题的评论列表，可以发帖、跟帖、称赞和评论别人的帖子等等。



(a)



(b)



(c)



(d)

上面的图片，图(a)展示的是快速评论工具栏，图(b)是跟帖列表，图(c)是跟帖编辑页面，图(d)是调用快捷分享来完成对跟帖列表中一个评论进行转发。如果您选择此方案，则您的项目需要同时集成快捷分享的 GUI 功能，因为评论与赞服务会在转发时使用到它。

使用这套 GUI 控件的时候，您需要先有一个如图(a)那样显示主题的页面，ShareSDK 并没有为主题显示提供解决方案（但是图(a)的显示效果，可以到 ShareSDK 的 Sample 中挖出来用），所以当您设计好主题显示效果以后，可以通过 xml 或者代码的方式，将快速评论工具栏添加到页面底部。下面是一个图(a)页面的 xml 布局：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#fff5f5f5" >

    <cn.sharesdk.framework.TitleLayout
        android:id="@+id/llTitle"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@drawable/auth_title_back" />

    <ScrollView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/llTitle"
        android:layout_marginBottom="45dp" >

        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical" >

            <cn.sharesdk.socialization.component.TopicTitle
                android:id="@+id/llTopicTitle"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content" />

            <ImageView
                android:id="@+id/ivIllustration"
```

```
        android:layout_width="fill_parent"
        android:layout_height="240dp"
        android:scaleType="centerInside"
        android:src="@drawable/pic" />
```

```
    <TextView
```

```
        android:id="@+id/tvText"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:autoLink="all"
        android:padding="10dp"
        android:text="@string/share_content"
        android:textColor="#ff383838"
        android:textSize="17dp" />
```

```
    </LinearLayout>
```

```
</ScrollView>
```

```
<ImageView
```

```
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/llTitle"
    android:background="@drawable/title_shadow" />
```

```
<cn.sharesdk.socialization.QuickCommentBar
```

```
    android:id="@+id/qcBar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
```



```
android:background="@drawable/ssdk_social_toolbar_bg"/>
```

```
</RelativeLayout>
```

从布局文件可以简单看到，通过 XML 的方式添加 QuickCommentBar 的方式和添加其他的控件并没有什么区别，这是因为 QuickCommentBar 其实只是个简单的 LinearLayout，考虑到部分开发者并不是在 Eclipse 上使用标准的 ADT 来开发，因此 ShareSDK 并没有为 QuickCommentBar 提供基于 xml 的自定义属性，所以完成上面配置以后，回到 java 方面，需要添加一些代码：

```
qcBar = (QuickCommentBar) pageView.findViewById(R.id.qcBar);  
qcBar.setTopic(topicId, topicTitle, topicPublishTime, topicAuthor);  
qcBar.getBackButton().setOnClickListener(this);  
qcBar.setOnekeyShare(oks);
```

这几行代码包含很多信息，须要逐一讲解。

首先从页面的 View 树中找到 QuickCommentBar 实例，其次，由于 QuickCommentBar 是面向主题的，它的初始化至少需要传递主题的 ID，主题的标题，此外主题的发布时间和作者可以直接为 null（不过，如果设置了，会好看很多）。这里的主题 ID 是开发者自己定义的、可以在开发者自己应用内唯一识别一个主题的字符串。由于 ShareSDK 内部识别一个主题依靠的不仅仅这个字段，因此不同的应用拥有相同的 ID 没有问题，只要这两个应用在 ShareSDK 注册到的 Appkey 不一样就行。

再者，因为 QuickCommentBar 默认会在左边添加一个返回按钮，如果开发者不需要这个按钮，可以通过 QuickCommentBar 的 getBackButton 方法获取到这个按钮，之后修改其可

视性就行了。不过上面的代码没有隐藏这个按钮，而是为它设置一个点击回调。

最后，由于 *评论与赞服务依赖于快捷分享*，因此在第四行代码中设置了一个快捷分享的实例给它。

事实上，评论与赞服务还提供了 *垃圾评论过滤机制*，可以通过类似于下面例子的方式，自定义并添加您的过滤条件：

```
qcBar = (QuickCommentBar) pageView.findViewById(R.id.qcBar);
// 各种初始化和设置
CommentFilter.Builder builder = new CommentFilter.Builder();
// 非空过滤器
builder.append(new FilterItem() {
    // 返回 true 表示是垃圾评论
    public boolean onFilter(String comment) {
        if (TextUtils.isEmpty(comment)) {
            return true;
        } else if (comment.trim().length() <= 0) {
            return true;
        } else if (comment.trim().toLowerCase().equals("null")) {
            return true;
        }
        return false;
    }
});
// append 其他的过滤条件
// 构造过滤器
filter = builder.build();
qcBar.setCommentFilter(filter);
```

`CommentFilter.Builder` 是垃圾过滤器 `CommentFilter` 的构造器，

`CommentFilter` 不能自己实例化，因此只能使用 `Builder` 来构造。`Builder` 通过 `append` 方法每次添加一个过滤条件 `FilterItem` 进来，将来过滤的时候，会按照这个顺序来执行。`FilterItem` 是一个接口，需要实现 `onFilter` 方法，这个方法会得到一个名为 `comment` 的 `String` 对象，它就是即将发表的评论。`comment` 会在 `onFilter` 里面判断，如果结果认为是垃圾评论，返回 `true`，否则返回 `false`。添加完所有条件以后，调用 `Builder` 的 `build` 方法，产生一个 `CommentFilter` 出来。将 `CommentFilter` 设置给 `QuickCommentBar` 就行了，当发表评论的时候，`QuickCommentBar` 会自行调用 `CommentFilter` 的 `filter` 方法过滤评论，如果判断结果是垃圾评论，会通过 `Toast` 告知用户。

评论发表出去以后，`QuickCommentBar` 默认会自己打开跟帖页面 `CommentListPage`。`CommentListPage` 是一个 `FakeActivity`，代码在评论与赞的 `jar` 内部，使用的时候就像调用快捷分享一样构造一个实例出来，然后 `show` 就行，不影响现有的其他页面。调用这个页面出来的代码如下：

```
CommentListPage page = new CommentListPage();
page.setTopic(topicId, topicTitle, publishTime, author);
page.setCommentFilter(filter);
page.setOnekeyShare(oks);
page.show(getContext(), null);
```

了解了 `QuickCommentBar` 的配置代码以后，上面的代码很容易理解：先实例化一个 `CommentListPage`，然后设置主题的信息（ID、标题、发布时间、作者），再设置垃圾过滤器、快捷分享实例，最后调用 `show` 方法，将页面显示出来。

`CommentListPage` 的显示效果就是前文图(b)。列表本身采用分页效果展示，可下拉刷新、滑至底部可以自动加载下一页。可以显示评论和“盖楼”效果，如果楼层太多，会折

叠部分中间的楼层，但是可以手动全部展开。每一个评论都在右上角有一个“赞”的图标，点击一下可以对此评论一个称赞，称赞以后不能取消。点击楼层或者评论的文字可以弹出一个 小窗，显示“回复”、“分享”和“复制”。选择“回复”，进入前文图(c)，选择“分享”，弹出快捷分享的宫格列表，效果如前文图(d)，选择“复制”，则将楼层或者评论的文本复制到系统的剪贴板中。

回复的时候允许同时分享到新浪微博和腾讯微博，其操作类似于快捷分享的编辑页面。

直接调用 Socialization 的网络接口

虽然上述的 GUI 控件也可以完成一定程度的自定义，但是如果这还不能满足您的需求，或是您不想使用这套控件，评论与赞可以为您提供完成上述功能所需要的网络接口，这些接口都放在类 `cn.sharesdk.socialization.Socialization` 中。
`Socialization` 提供的方法列表如下：

限定符和类型	方法和说明
HashMap<String, Object>	getCommentOutline(String topicId) 此方法通过 topicId 返回快速评论工具栏的喜欢数、分享数和评论数
String	likeTopic(String topicId, String topicTitle) 此方法用于向 ShareSDK 服务器发送一个对主题 的喜欢操作
String	shareTopic(int snsPlatId, String topicId, String topicTitle, ShareParams shareParams, HashMap<String, Object> res) 分享一个主题以后，需调用此接口完成分享数据的统计。其中的 snsPlatId 可以通过 ShareSDK. platformNameToId 通过平台名称获取

String	replyTopic(int snsPlatId, String topicId, String topicTitle, String reply) 此方法用来回复一个主题
ArrayList <Comment>	getCommentList(String topicId) 获取一个主题的评论列表。此列表仅包含评论的 ID 和排序需要的部分字段，评论的实际内容不包含其中
boolean	getCommentDetails(String topicId, ArrayList<Comment> comments) 填充指定的评论列表，返回 true 表示填充成功
String	likeComment(String topicId, String topicTitle, String commentId) 称赞一个主题时调用此接口
String	shareComment(int snsPlatId, String topicId, String topicTitle, String commentId, ShareParams shareParams, HashMap<String, Object> res) 分享一个评论以后，需要调用此接口完成分享数统计
String	replyComment(int snsPlatId, String topicId, String topicTitle, String commentId, String reply) 此方法用来回复一个评论

更详细的方法列表，请参考 [Api 文档](#)。

直接调用网络接口，意味着您须要自行把握统计接口的调用，评论之前也须要自行构造和调用垃圾评论过滤器来过滤评论内容，获取评论列表以后，也须要自行处理排序和盖楼等效果。但直接调用网络接口有一个好处，就是您的应用可以不依赖快捷分享。最后，由于 **这些接口大部分都须要连接网络**，因此最好在非主线程中调用，以免阻塞主线程。

常见问题

本章内容包括：

- 第三方登录步骤
- 微信、易信无法分享
- 改应用信息后不能分享
- 混淆提示错误
- 豆瓣不能分享
- 印象笔记转正式服务器
- 获取腾讯微博图片地址
- 集成 ShareSDK 到 cocos2d-x
- Google+或 QQ 无法分享
- 快捷分享九宫格图标无法排序

第三方登录步骤

问题：如何实现第三方账号登录？

回答：详细步骤，您可以参考章节“[获取用户资料及第三方登录](#)”和 ShareSDK 官方论坛相关文档“[ShareSDK 第三方登录步骤](#)”中的描述。

微信、易信无法分享

问题：使用自己或者 sample 的代码都不能完成微信（易信）的分享，提示“分享正在后台进行”，然后就没有结果了。

回答：这是*因为您的应用没有在微信（易信）开放平台上正确注册您的 android 客户端信息的原因*。请参考微信、易信的签名和注册章节，完成信息填写并将应用提交微信（易信）开放平台审核，审核通过以后您的应用就可以正确分享了。

改应用信息后不能分享

问题：在 Sample 中使用默认的 Appkey 可以分享，但是改了“ShareSDK.xml”中的信息为我的 AppKey 以后就不能分享了

回答：某一个平台授权完成以后，其授权信息会保存在应用的 SharedPreferences 中，此后的操作会读取类似于 AccessToken 之类的数据，都是在此处读取。很多平台在分享的时候需要携带 Appkey 等信息，而 Appkey 和 AccessToken 是绑定的。因此*当您授权以后，再修改 Appkey，就会导致*

Appkey 和 AccessToken 不匹配，因此被分享平台拒绝。为了不被拒绝，您修改 Appkey 的时候，需要**清除应用的缓存数据，或者重新安装应用**。

混淆提示错误

问题：集成了 ShareSDK 以后，混淆出现问题。

回答：混淆的时候需要在混淆脚本中 keep ShareSDK 的包，具体的内容可以参考 Sample，下面是主要的一些内容，如：

```
-keep class cn.sharesdk.**{*;}  
-keep class com.sina.**{*;}
```

上面的脚本分别 keep 了 ShareSDK 和新浪微博 SSO 授权方式所需要的所有 jar 和源码。

考虑到部分开发者会使用附带 Sample 中的侧栏控件，那么此时还需要 keep 这个控件的代码：

```
-keep class m.framework.**{*;}
```

最后请注意，**注册到 ShareSDKUIShell 下用于自定义授权页面的类也是需要 keep 的**，否则 ShareSDK 将找不到它。

豆瓣不能分享

问题：使用 Sample 也不能利用豆瓣分享

回答：Sample 中的豆瓣应用信息是未审核通过的账号信息，这些应用只是用于演示，只有已经在应用详情住添加了测试账号，才能测试。

豆瓣添加测试账号的方法如下：



完成上述操作以后还不能测试，则需要联系豆瓣的客服人员了，因为这应该是他们的问题。

印象笔记转正式服务器

问题：印象笔记如何从沙箱模式切换到生产模式。

回答：印象笔记切换服务器的方式，可以通过修改应用信息中的 **“HostType”** 来实现。在**中国大陆**，印象笔记有两个服务器，一个是沙箱(**sandbox**)，一个是生产服务器(**china**)。一般你注册应用，它会先让你使用 **sandbox**，当你完成测试以后，可以到 <http://dev.yinxiang.com/support/> 上激活你的 **ConsumerKey**，激活成功后，修改 **HostType** 为 **china** 就好了。至于如果您申请的是**国际版的印象笔记 (Evernote)**，则其生产服务器类型为 **“product”**。

获取腾讯微博图片地址

问题：新浪微博分享后，为何拿到的图片地址打不开？

回答：新浪微博对分享后的图片做了处理，从腾讯微博的“[API 文档/API 问题 QA](#)”中有记录：其返回数据中的图片，需要在其地址后面添加“/120”、“/160”、“/460”和“/2000”，分表代表不同的尺寸。而头像，则后面添加“/20”、“/30”、“/40”、“/50”和“100”，也同样返回不同尺寸的图片。

集成 ShareSDK 到 cocos2d-x

问题：ShareSDK 是否支持 coco2d-x？

回答：ShareSDK 理论上支持所有可以引用 jar 包的 Android 第三方平台，因此 ShareSDK 支持 coco2d-x，也支持 Unity3D。关于 cocos2d-x 中集成 ShareSDK 的方法，可以参考开发者[海浪](#)的[博文](#)。在此 ShareSDK 再次衷心感谢其对我们发展的支持和帮助！更多 ShareSDK 的官方、第三方使用示例，请光临我们的 [Wiki](#)。

Google+或 QQ 客户端无法分享

问题：使用快捷分享，提示 Google+（或 QQ）未安装或版本过低。

回答：Google+（或 QQ 客户端）的分享基于其客户端，如果没有安装 Google+（或 QQ）客户端或此客户端版本太低，会收到错误提示。如果您是直接调用接口分享，可以通过 isValid 方法，判断 Google+（或 QQ）客户端是否已处于可用状态。

快捷分享九宫格图标无法排序

问题:虽然在 ShareSDK.xml 中设置了不同平台的 SortId,但是在九宫格中排列的顺序还是不正确

回答:在部分手机上可能会出现编码上的问题,或者开发者在编写配置文件的时候,误删了一些 xml 的 tag,导致 ShareSDK 解析此文件时出错。请仔细查看 Logcat 中的日志,如果提示缺少结束的 tag,应该检查是否误删。如果提示“org.xmlpull.v1.XmlPullParserException: PI must not start with xml”,很可能是手机编码有问题,请删除 ShareSDK.xml 文件的 xml 头部(即第一行的“<?xml version="1.0" encoding="utf-8"?>”)就行了。