

# Android接入步骤-1.6

## 第一步：获取必要的接入信息

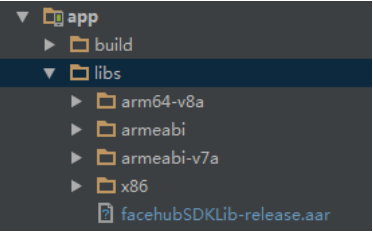
- 申请app\_id 以上信息可以在[面馆云开发者注册](#)页面获取
- 在面馆云开发者后台绑定应用ID和应用签名

## 第二步：下载与配置

您可以[点击这里](#)下载面馆云SDK。

1.在gradle中加入facehubSDKLib1.0.0-release.aar

1.1 请根据您的应用的编译版本下载相应的.aar文件，并将相应.aar文件 复制到app/libs下；



1.2 修改project的build.gradle

```
allprojects {
    repositories {
        jcenter()
        flatDir {
            dirs 'libs'
        }
    }
}
```

1.3 修改Module:app的build.gradle

```
dependencies {
    compile 'com.android.support:support-v4:22+'
    compile 'com.android.support:appcompat-v7:22+'
    compile 'com.android.support:recyclerview-v7:22+'

    compile(name:'facehubSDKLib-release', ext:'aar')
    compile 'com.loopj.android:android-async-http:1.4.9'
    compile 'de.greenrobot:eventbus:2.4.0'
    compile 'com.nostra13.universalimageloader:universal-image-loader:1.9.5'
    compile 'in.srain.cube:grid-view-with-header-footer:1.0.12'
}
```

注意！如果您使用的\*\*编译版本\*\*是23，请在添加依赖时使用相应的版本 如:

```
compile 'com.android.support:support-v4:23+'
compile 'com.android.support:appcompat-v7:23+'
compile 'com.android.support:recyclerview-v7:23+'
```

## 第三步：开始集成

### 一、初始化SDK

在你的application onCreate时调用初始化：

1.目前商店页有两种风格可供开发者挑选：

默认风格(0)：



风格1:



2.有五种配色方案可供开发者挑选:

- 配色0：默认配色



- 配色1：自定义主题色(示例：荧光绿)



- 配色2：黑色配色



推荐表情

- 

抖抖村

宅萌的极点，快乐的村子

下载
- 

瓜籽儿夏日版

“瓜籽儿”陪你过酷暑~~

下载
- 

寿司绵绵蛋

软萌暖寿司绵绵蛋

下载

最近热门



- 配色3：白色配色



推荐表情

- 

抖抖村

宅萌的极点，快乐的村子

下载
- 

瓜籽儿夏日版

“瓜籽儿”陪你过酷暑~~

下载
- 

寿司绵绵蛋

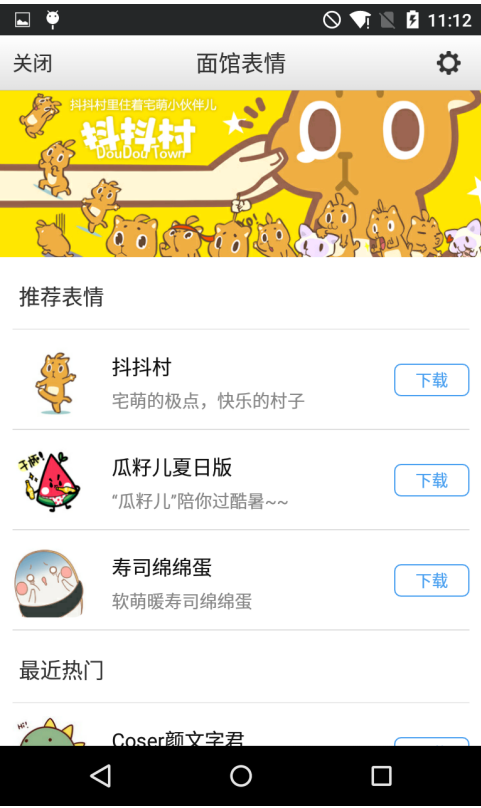
软萌暖寿司绵绵蛋

下载

最近热门



- 配色4：灰色配色

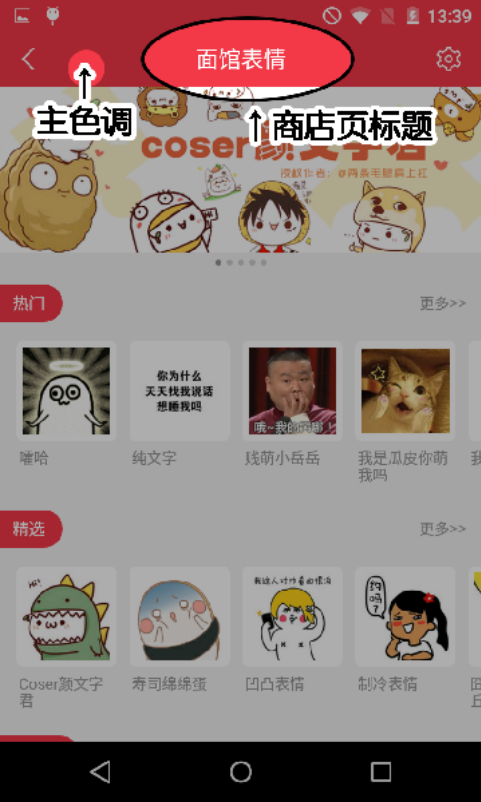


3. 在你的application onCreate时调用初始化:

```
java FacehubApi.init(getApplicationContext(),APP_ID); //初始化 FacehubApi.initViews(getApplicationContext()); //初始化api中的views FacehubApi.initViews(getApplicationContext());
```

注意: - ~~ FacehubApi.setAppId() ~~ 方法在v1.5之后与 init() 合并;

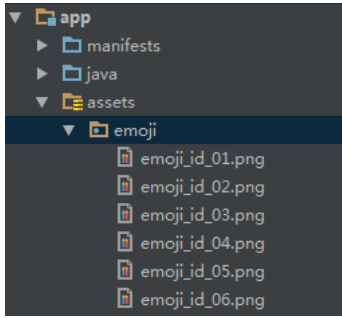
- 如果不使用表情同步功能, 可调用 init(Context context, String appId ,boolean offlineMode) , 其中 offlineMode 参数为 true 时, 则启用离线模式, 即接入方无需手动创建/登录用户账号, 但也无法在多台设备上同步同一账号的表情数据。



## 二、配置本地预置表情

(如果没有本地预置表情, 请跳过本步骤)

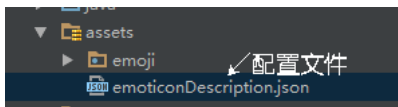
1. 请将表情资源文件放置在 `assets://emoji/` 目录下;



1. 请编写配置Json文件, 放在assets目录下, 文件格式请参考附件 `testDescription.json` 。

- 注意: "id"字段请保证与资源文件名相同,即 文件名=id.format
- 特别说明: sdk-v1.5.1版本之后, 关于自定义表情的混排配置, 改为在json中配置, 即 `needMixLayout` 字段。
- `row` 和 `column` 字段用来控制自定义列表在键盘中显示的行列数, 请根据实际效果进行调整。

2. 在API初始化之后调用 `loadEmoticonFromLocal(int, String)`



```
//同一个version的jsonConfigFile只会解析一次,不用担心性能,如需更新表情内容,请增加version的值
/**
 * 参数说明 : {@link FacehubApi#loadEmoticonFromLocal(int, String)}
 * 1.配置文件版本号;
 * 2.配置文件,在assets文件夹内的具体路径;
 * 3.抛出异常 : {@link LocalEmoPackageParseException} ,配置JSON解析出错时抛出异常;
 */
try {
    FacehubApi.getApi().loadEmoticonFromLocal(2,"emoticonDescription.json");
} catch (LocalEmoPackageParseException e) {
    Log.e(Constants.TAG,"解析预置表情 配置Json出错 : " + e);
    e.printStackTrace();
}
```

特别说明: sdk-v1.5.1版本之后, 关于自定义表情的混排配置, 改为在json中配置, 即 `needMixLayout` 字段。

### 三、注册用户

```
try {
    FacehubApi.getApi().registerUser(bindingUserId, new ResultHandlerInterface() {
        @Override
        public void onResponse(Object response) {
            User user = (User) response;
            String content = "注册用户成功!\nId : " + user.getUserId() + "\nToken : " + user.getToken();
            LogX.d(content);
            textView.setText(content);
        }
        @Override
        public void onError(Exception e) {
            String s = "注册用户出错 : " + e;
            LogX.e(s);
            textView.setText(s);
        }
    });
} catch (FacehubSDKException e) {
    String sss = "注册用户出错 : " + e;
    LogX.e(sss);
    textView.setText(sss);
}
```

参数说明: - `bindingUserId` 用户在您应用内的id; - `ResultHandlerInterface` 注册结果回调, 如果注册成功, 则会在 `onResponse` 中返回一个 `User` 对象, `User.getUserId()` 获取到与 `bindingUserId` 对应SDK内的用户id; - `FacehubSDKException` 异常: 如果在初始化中设置了 `offlineMode=true`, 则不可进行注册操作, 因为此时为离线模式, 不允许手动注册。

### 四、登录/恢复数据

直接使用您应用的用户id登录，调用：

```
FacehubApi.getApi().login( bindingUserId , new ResultHandlerInterface() {
    @Override
    public void onResponse(Object response) {
        User user = (User) response;
        String content = "登录用户成功!\nId : " + user.getUserId() + "\nToken : " + user.getToken();
        LogX.d(content);
        textView.setText(content);
    }

    @Override
    public void onError(Exception e) {
        String s = "登录用户出错 : " + e;
        LogX.e(s);
        textView.setText(s);
    }
}, new ProgressInterface() {
    @Override
    public void onProgress(double process) {
        //登陆进度
    }
});
```

参数说明： - `bindingUserId` 用户在您应用内的id； - `ResultHandlerInterface` 登录结果回调，如果登录成功，则会在 `onResponse` 中返回一个 `User` 对象，`User.getUserId()` 获取到与 `bindingUserId` 对应SDK内的用户id；

- 备注：如果因为网络等原因导致登录出错，SDK会在键盘显示时自动进行登录重试，您也可以重新手动调用 `login()` 函数重新登录；
- 如果您不使用表情同步功能，则无需手动调用\*\*登录\*\*与\*\*退出\*\*函数，相应的需要在调用 `FacehubApi.init()` 函数时设置参数 `offlineMode` 为 `true`

## 五、设置表情键盘

1.在你的聊天界面中加入：

```
<!--通常放置在你的聊天输入框下方 -->
<com.azusasoftware.facehubcloudsdk.views.EmoticonKeyboardView
    android:id="@+id/chatting_keyboard"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
</com.azusasoftware.facehubcloudsdk.views.EmoticonKeyboardView>
```



2.初始化View：

2.1 找到对应View

```
EmoticonKeyboardView mEmoticonKeyboard = findViewById(R.id.chatting_keyboard);
```

2.2 调用初始化

```

/**
 * 键盘初始化参数说明：{@link EmoticonKeyboardView#initKeyboard(boolean, String, View.OnClickListener)}
 * 1.是否有本地预置表情
 * 2.键盘右下角发送按钮的配色(RGB值)，可设置为空
 * 3.键盘右下角发送按钮的点击回调，可设置为空
 */
emoticonKeyboardView.initKeyboard(true, "#467fff", new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String string = "点击发送按钮,发送消息.";
        textView.setText(string);
        showToast(string, false);
    }
});
//如果没有本地预置表情，则调用
emoticonKeyboardView.initKeyboard(false, null, null);

```

## 六、增加监听器

### 6.1 点击回调

```

/**
 * 五、
 * 点击表情后的回调
 * 可根据 {@link Emoticon#isLocal()} 来区分是否是预存的表情
 * 可根据{@link Emoticon#getFullPath()} 来拿取表情文件的路径
 * 根据{@link Emoticon#getThumbPath()} 来拿取表情缩略图路径
 * 注意！：如果是本地表情，则返回其在assets的路径,如"emoji/emoji_id_1.png",因此其实际路径应为 "assets://emoji/emoji_id_1.png"
 */
emoticonKeyboardView.setEmoticonSendListener(new EmoticonSendListener() {
    @Override
    public void onSend(Emoticon emoticon) {
        currentEmoId = emoticon.getId();
        String s = "输入表情：[" + emoticon.getDescription() + "]";
        String content="";

        if(emoticon.isEmoji()){ //系统emoji与颜文字
            s = "发送表情：[" + emoticon.getDescription() + "]";
            textView.setText(s);
            showToast(s, false);
        }else if (emoticon.isLocal()) { //自定义表情
            switch (emoticon.getLocalType()){
                case "custom_list":
                    content = s + "\n自定义本地表情资源路径： " + "assets://" + emoticon.getFullPath();
                    boolean isNeedMixLayout = emoticon.isNeedMixLayout(); //用来判断是否图文混排
                    content += "\n需要图文混排： " + isNeedMixLayout;
                    break;
                case "voice":
                    content = s + "\n语音表情，描述： " + emoticon.getDescription();
                    break;
            }
            textView.setText(content);
            showToast(s, false);
        } else { //sdk提供的表情
            s = "发送表情：[" + emoticon.getId() + "]";
            content = s + "\n表情文件路径： " + emoticon.getFullPath();
            textView.setText(content);
            showToast(s, false);
        }
    }
});

```

### 6.2 如果在Json中设置了图文混排为true，则需要为键盘绑定删除按钮的监听

```

emoticonKeyboardView.setOnDeleteListener(new OnDeleteListener() {
    @Override
    public void onDelete() {
        //删除输入框中的内容
        textView.setText("");
        showToast("点击删除", false);
    }
});

```



6.3 如果\*\*【允许图文混排】\*\*，请根据输入框的内容设置\*\*【发送按钮】\*\*的状态，调用

```
EmoticonKeyboardView.setSendButtonEnabled(boolean) ；
```

七、显示和隐藏键盘：

```
mEmoticonKeyboard.show();
mEmoticonKeyboard.hide();
```

（目前表情键盘仅支持宽度为全屏宽度，如有其它需求或问题，请联系我们。）

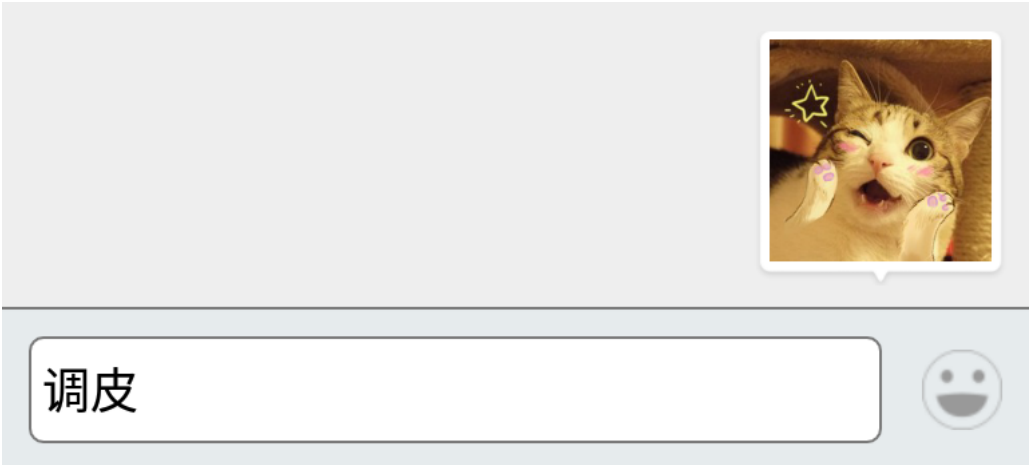
- 切换横/竖屏时请调用

```
emoticonKeyboardView.onScreenWidthChange()
```

更新键盘视图；

八、表情联想

您可通过含义来联想已收藏的表情，例如用户输入“开心”，联想出相应的表情。调用 `FacehubApi.getApi().findEmoticonByDescription(keyword)` ,会返回一个与关键



词含义对应的 `Emoticon` 对象；

- 注意，此功能只会在用户已收藏的表情中检索！

九、显示收到的表情：

在接受到其他用户发来的带有面馆云表情的聊天信息时，可以通过表情的ID来获取图片以显示表情。

```
FacehubApi.getApi().getEmoticonById( emoticonId, new ResultHandlerInterface() {
    @Override
    public void onResponse(Object response) {
        Emoticon emoticon = (Emoticon) response;
        String content = "获取到表情[" + emoticon.getId() + "]\npath : " + emoticon.getFullPath();
        textView.setText(content);
    }

    @Override
    public void onError(Exception e) {
        String content = "获取表情[" + currentEmoId + "失败! : " + e;
        textView.setText(content);
    }
});
```

十、退出登录

退出登录时，请调用 `FacehubApi.getApi().exitViews()` 关闭SDK的页面视图，然后调用退出函数 `FacehubApi.getApi().logout()` ；

- 如果您不使用表情同步功能，则无需手动调用\*\*退出\*\*函数，相应的需要在调用 `FacehubApi.init()` 函数时设置参数 `offlineMode` 为 `true` ，启用离线模式

常见问题

- 登录失败：
  - 请检查appld, userId与authToken是否有误；

- 关于混淆：
  - 面馆云SDK为开源项目，如您的应用需要混淆，请保留SDK内的所有类即可。即：`-keep class com.azusasoftware.facehubcloudsdk.** {*;}`
  - 如在混淆时出现log4j库的警告，请在 proguard-rules.pro 中添加 `-dontwarn org.apache.log4j.**`
- 商店页/列表管理页崩溃：
  - 请保证您的应用的编译版本、support库的版本与SDK的编译版本对应；
- 本地表情配置出错：
  - 请检查 json 配置文件中 id 和 format 是否与文件名对应，即 文件名=id.format；
  - 表情资源文件请勿使用中文字符！
- 本地表情不显示：
  - 调用本地表情的 `Emoticon.getFullPath()` 之后，返回其在assets的路径,如 `"emoji/emoji_id_1.png"`，因此其实际路径应为 `"assets://emoji/emoji_id_1.png"`。
- 登录/注册出错：
  - 请检查初始化时是否设置了 `offlineMode=true`，如果此项为true，则启用离线模式，您无法手动调用登录/注册。
- 注册账号后，列表错乱：
  - 调用注册后不会自动登陆，如需获取列表，需手动调用登录函数。
- 自定义列表显示效果有误：
  - 请修改配置json文件中的 `row` 和 `column` 字段，以达到最优效果。
- 发送按钮点击后无效：
  - 请检查 `EmoticonKeyboardView#initKeyboard(boolean, String, View.OnClickListener)` 中第三个参数（发送回调）不为 `null`。