

微博SDK-Android接入文档

SDK名称:微博SDK

开发者:北京微梦创科网络技术有限公司

主要功能:实现微博登录、分享功能

SDK版本:13.10.6

[更新日志](#)

[工程配置](#)

[使用微博授权](#)

[分享内容到微博](#)

[微博SDK隐私政策说明](#)

接入须知

工程要求

| 属性 | 限制内容 |
|------------------|--|
| targetSdkVersion | <= 34 |
| minSdk | >= 22 (低于22时, 调用API前检查, Build.VERSION.SDK_INT>=22) |
| ABI | 无so依赖 |
| FileProvider | 默认authority:“\${你的包名}.wbsdk.fileprovider”;同时支持开发者自定义FileProvider及authority |

业务限制

| 属性 | 分享对象 | 限制内容 |
|-------|-------------------|---|
| Web分享 | WebpageObject | Web分享只支持分享文字 |
| 单图分享 | ImageObject | 目前支持图片格式:jpg、png、webp 图片大小限制:图片转码后需小于200k, 可通过ImageObject.checkArgs检查 说明:单图分享存在转码操作, 有单图分享需求建议同样使用多图分享API |
| 多图分享 | MultilImageObject | 目前支持图片格式:jpg、png、gif 图片大小限制:多图分享时, 每张图片建议小于10MB 图片个数限制:最多18张 |
| 视频分享 | VideoSourceObject | 目前支持视频格式:mp4 |
| 超话分享 | SuperGroupObject | 非完全开放类型, 请联系微博商务洽谈, 取得白名单权限后再进行该功能开发; 超话分享开发文档 |

工程配置

集成SDK (请选择其中一种方式集成)

方式一 (使用远程依赖)

- 1. 在project根目录的build.gradle文件中添加依赖配置

```
allprojects {
    repositories {
        mavenCentral()
        .....
    }
}
```

- 2. 在module的build.gradle文件中添加远程依赖

```
dependencies {  
    implementation 'io.github.sinaweibosdk:core:13.10.6@aar'  
}
```

方式二(使用aar本地依赖)

- 1. 打开最新文档下载需要依赖的SDK版本的aar,将AAR文件作为工程依赖
- 2. 在module的build.gradle文件中添加本地依赖

```
dependencies {  
    implementation(name:'core-13.10.6',ext:'aar')  
}
```

Proguard(混淆配置)

```
-keep public class com.sina.weibo.sdk.**{*};
```

接入代码

初始化SDK

```
public class DemoApplication extends Application {  
    private static final String APP_KEY = "在微博开发平台为应用申请的App Key";  
    private static final String REDIRECT_URL = "在微博开放平台设置的授权回调地址";  
    private static final String SCOPE = "在微博开放平台为应用申请的高级权限";  
  
    @Override  
    protected void onCreate() {  
        super.onCreate();  
        initSdk();  
    }  
  
    /**  
     * 初始化sdk。  
     * 理论上使用前只要初始化一次即可,具体分享及授权登录时将不需要再次初始化。  
     */  
    private void initSdk() {  
        AuthInfo authInfo = new AuthInfo(this, APP_KEY, REDIRECT_URL,  
            mWBAPI = WBAPIFactory.createWBAPI(this); // 传Context即可,不再  
            mWBAPI.registerApp(this, authInfo, new SdkListener() {  
                @Override  
                public void onInitSuccess() {  
                    // SDK初始化成功回调,成功一次后再次初始化将不再有任何回调  
                }  
  
                @Override  
                public void onInitFailure(Exception e) {  
                    // SDK初始化失败回调  
                }  
            }));  
    }  
}
```

使用微博授权

1. 在Activity中调用授权API(请选择其中一种方式)

//方式一,优先客户端授权,未安装时使用Web授权

```
private void startAuth() {
    mWBAPI.authorize(this, new WbAuthListener() {
        @Override
        public void onComplete(Oauth2AccessToken token) {
            Toast.makeText(MainActivity.this, "微博授权成功", Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onError(UiError error) {
            Toast.makeText(MainActivity.this, "微博授权出错", Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onCancel() {
            Toast.makeText(MainActivity.this, "微博授权取消", Toast.LENGTH_SHORT).show();
        }
    });
}
```

//方式二,指定通过客户端授权操作

```
private void startClientAuth() {
    mWBAPI.authorizeClient(this, new WbAuthListener() {
        @Override
        public void onComplete(Oauth2AccessToken token) {
            Toast.makeText(MainActivity.this, "微博授权成功", Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onError(UiError error) {
            Toast.makeText(MainActivity.this, "微博授权出错", Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onCancel() {
            Toast.makeText(MainActivity.this, "微博授权取消", Toast.LENGTH_SHORT).show();
        }
    });
}
```

//方式三,指定通过网页(Web)授权操作

```
private void startWebAuth() {
    mWBAPI.authorizeWeb(this, new WbAuthListener() {
        @Override
        public void onComplete(Oauth2AccessToken token) {
            Toast.makeText(MainActivity.this, "微博授权成功", Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onError(UiError error) {
            Toast.makeText(MainActivity.this, "微博授权出错:" + error.getMessage(), Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onCancel() {
            Toast.makeText(MainActivity.this, "微博授权取消", Toast.LENGTH_SHORT).show();
        }
    });
}
```

2. 重写Activity的onActivityResult()方法,示例如下

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

注:这里是设置授权的回调,如果不重写,将收不到授权结果。

3. 错误码定位:OAuth2.0 错误码

分享内容到微博

1. 实现回调接口,代码示例如下

```

private static class ShareCallback implements WbShareCallback {
    @Override
    public void onComplete() {
        Toast.makeText(ShareActivity.this, "分享成功", Toast.LENGTH_SHORT);
    }

    @Override
    public void onError(UiError error) {
        Toast.makeText(ShareActivity.this, "分享失败:" + error.errorMessage);
    }

    @Override
    public void onCancel() {
        Toast.makeText(ShareActivity.this, "分享取消", Toast.LENGTH_SHORT);
    }
}

```

2. 重写Activity的onActivityResult()方法,示例如下

```

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (mWBAPI != null) {
        mWBAPI.doResultIntent(data, this);
    }
}

```

注:和授权登录同理,这里是设置分享的回调,如果不重写,将收不到分享结果。

3. 微博分享示例代码如下

```

private void doWeiboShare() {
    //相册-照片
    if (mShareMediaImg.isChecked()) {
        openGallery(REQ_PICK_IMAGE_COMPOSER, false);
        return;
    }
    //相册-视频
    if (mShareMediaVideo.isChecked()) {
        openGallery(REQ_PICK_VIDEO_COMPOSER, true);
        return;
    }
    WeiboMultiMessage message = new WeiboMultiMessage();

    TextObject textObject = new TextObject();
    String text = "我正在使用微博客户端发博器分享文字。";

    // 分享文字
    if (mShareText.isChecked()) {
        text = "这里设置您要分享的内容!";
        textObject.text = text;
        message.textObject = textObject;
    }

    // 分享图片
    if (mShareImage.isChecked()) {
        ImageObject imageObject = new ImageObject();
        Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.image);
        imageObject.setImageData(bitmap);
        message.imageObject = imageObject;
    }

    // 分享网页
    if (mShareUrl.isChecked()) {
        WebpageObject webObject = new WebpageObject();
        webObject.identify = UUID.randomUUID().toString();
        webObject.title = "标题";
        webObject.description = "描述";
        Bitmap bitmap = BitmapFactory.decodeResource(getResources(), R.drawable.image);
        ByteArrayOutputStream os = null;
        try {
            os = new ByteArrayOutputStream();
            bitmap.compress(Bitmap.CompressFormat.JPEG, 85, os);
            webObject.thumbData = os.toByteArray();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                if (os != null) {
                    os.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```

webObject.actionUrl = "https://weibo.com";
webObject.defaultText = "分享网页";
message.mediaObject = webObject;
}

if (mShareMultiImage.isChecked()) {
    // 分享多图
    MultiImageObject multiImageObject = new MultiImageObject();
    ArrayList<Uri> list = new ArrayList<>();
    File externalFilesDir = getExternalFilesDir(null);
    File externalCacheDir = getExternalCacheDir();
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
        String authority = this.getPackageName() + ".wbsdk.fileprovider";
        list.add(FileProvider.getUriForFile(this, authority, new File(externalFilesDir, "aaa.png")));
        list.add(FileProvider.getUriForFile(this, authority, new File(externalCacheDir, "bbb.jpg")));
        list.add(FileProvider.getUriForFile(this, authority, new File(externalFilesDir, "file_image.png")));
        list.add(FileProvider.getUriForFile(this, authority, new File(externalCacheDir, "file_image.jpg")));
        list.add(FileProvider.getUriForFile(this, authority, new File(externalFilesDir, "file_image.png")));
        list.add(FileProvider.getUriForFile(this, authority, new File(externalCacheDir, "file_image.jpg")));
        list.add(FileProvider.getUriForFile(this, authority, new File(externalFilesDir, "file_image.png")));
        list.add(FileProvider.getUriForFile(this, authority, new File(externalCacheDir, "file_image.jpg")));
    } else {
        list.add(Uri.fromFile(new File(externalFilesDir, "aaa.png")));
        list.add(Uri.fromFile(new File(externalCacheDir, "bbb.jpg")));
        list.add(Uri.fromFile(new File(externalFilesDir, "file_image.png")));
        list.add(Uri.fromFile(new File(externalCacheDir, "file_image.jpg")));
        list.add(Uri.fromFile(new File(externalFilesDir, "file_image.png")));
        list.add(Uri.fromFile(new File(externalCacheDir, "file_image.jpg")));
        list.add(Uri.fromFile(new File(externalFilesDir, "file_image.png")));
        list.add(Uri.fromFile(new File(externalCacheDir, "file_image.jpg")));
    }
    multiImageObject.imageList = list;
    message.multiImageObject = multiImageObject;
}

if (mShareVideo.isChecked()) {
    // 分享视频
    VideoSourceObject videoObject = new VideoSourceObject();
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
        String filePath = getExternalFilesDir(null) + "/file_video/eee";
        File videoFile = new File(filePath);
        if (!videoFile.getParentFile().exists()) {
            videoFile.getParentFile().mkdir();
        }
        videoObject.videoPath = FileProvider.getUriForFile(this, this.getPackageName(), videoFile);
    } else {
        videoObject.videoPath = Uri.fromFile(new File(getExternalFilesDir(null), "file_video/eee"));
    }
    message.videoSourceObject = videoObject;
}

if (mShareSuperGroup.isChecked()) {
    // 如果想要拉起超话类型发布者需要传superGroupobject对象
    SuperGroupObject superGroupObject = new SuperGroupObject();
    /**
     * 超话名, 可以传可以不传
     * 1、如果不传该参数则拉起选择超话弹层
     * 2、如果传该参数, 不出选择超话弹层, 直接选中超话
     */
    String sgName = ((EditText) findViewById(R.id.edit_sg_name)).getText().toString();
    if (TextUtils.isEmpty(sgName)) {
        sgName = this.getString(R.string.demo_sg_name);
    }
    superGroupObject.sgName = sgName;
    // 超话板块, 可以传可以不传
    String sgSection = ((EditText) findViewById(R.id.edit_sg_section)).getText().toString();
    superGroupObject.secName = sgSection;
    // 扩展参数, 需要透传的信息加在这个参数里
    String sgExt = ((EditText) findViewById(R.id.edit_sg_ext)).getText().toString();
    superGroupObject.sgExtParam = sgExt;
    message.superGroupObject = superGroupObject;
}

mWBAPI.shareMessage(this, message, mShareClientOnly.isChecked());
}

```