



# 锐动 Android 视频编辑 SDK



作者：RD Android Team

版本：Beta 版

日期：2017-1-14

# 1 编写目的

预期读者：

有视频开发经验或者无经验的，打算或者正在使用“锐动 Android 视频编辑 SDK”的相关工程师。

- Android 软件工程师。
- 产品经理。
- QA

# 2 名词解释

- 分辨率：用于计算机视频处理的图像，以水平和垂直方向上所能显示的像素数来表示分辨率。常见视频分辨率的有 1080P 即 1920x1080，720P 即 1080x720，640x480 等。
- 宽高比：视频分辨率的宽高比，常见的有 16:9，4:3，1:1。锐动视频编辑 SDK 对各宽高比的视频都支持编辑，导出的默认分辨率是 640x360，宽高比是 16:9。
- 帧率：每秒的帧数(fps)或者说帧率表示图形处理器处理场时每秒钟能够更新的次数。
- 码率：数据传输时单位时间传送的数据位数,一般我们用的单位是 kbps 即千位每秒。
- 素材：来自系统相机，其他 app，电脑，网络的照片，音乐，视频等。
- 视频分割：把视频分割为若干个片段，可用于从视频中截取一个或者多个精彩瞬间，或者删除不喜欢的片段，分割后的多个片段也可以用于重新调整顺序，视频合并等。
- 视频裁切：从视频画面中裁切出来仅需要的部分，锐动视频编辑 SDK 针对手机竖

屏拍摄的画面,支持动态调整裁切区域,帮助用户不同时间点选择自己需要的画面。

- 视频旋转: 对视频进行 90°, 180°, 270° 旋转, 一般用于矫正用户手机上录制的头像横置的视频。
- 视频镜像: 对视频左右镜像。
- 视频变速: 对视频播放速度调整, 实现慢镜头或者快镜头效果。
- 视频配乐: 选取本地或者网络音乐作为视频的背景音乐。
- 视频配音: 通过 mic phone 对视频配音。
- 视频字幕: 使用文字标注视频。
- 视频滤镜: 调整视频的画面颜色效果。
- 视频特效: 在视频里面增加特效动画或声音
- 合并转场: 两个视频片段之间的衔接效果。
- 主题(模版): 应用视频的整体效果方案。

## 3 集成步骤

### 3.1 运行环境

- Android 4.1 (api 16) 以上;
- 处理器: 双核 1GHz 以上 CPU(目前只支持 ARM CPU, X86、MIPS 暂不支持);  
推荐四核 1.2GHz 以上 CPU
- 内存: 1 GB 以上;

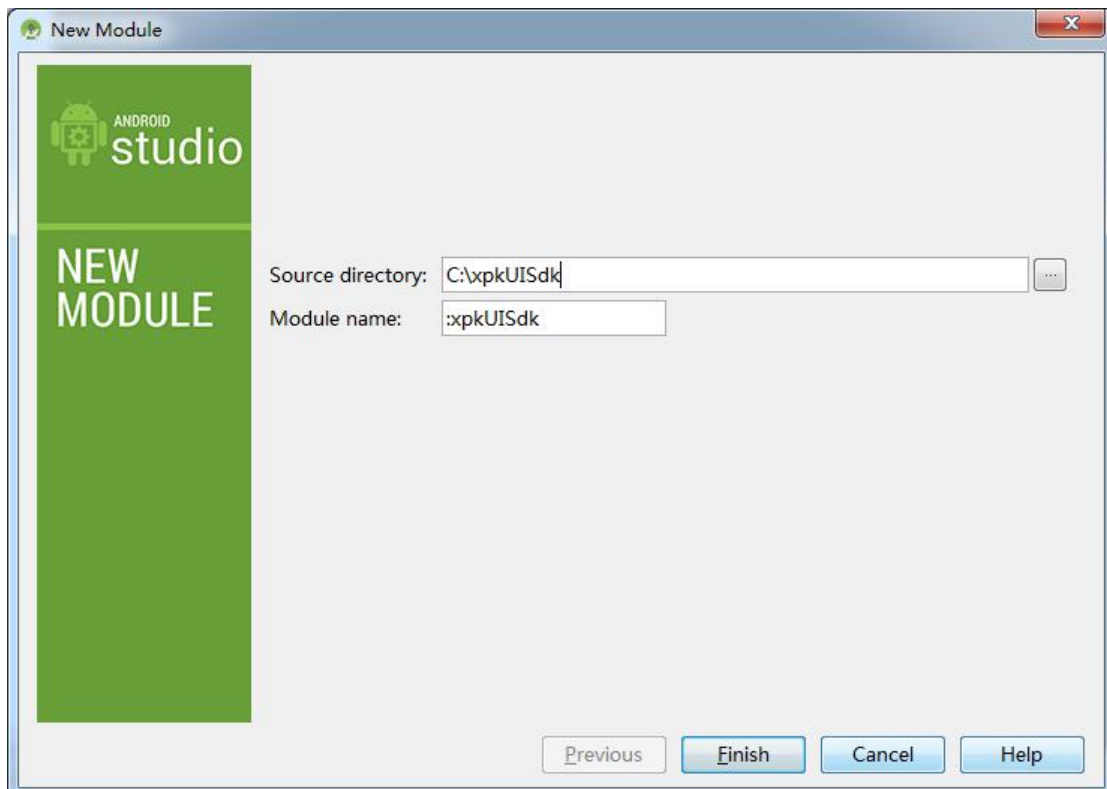
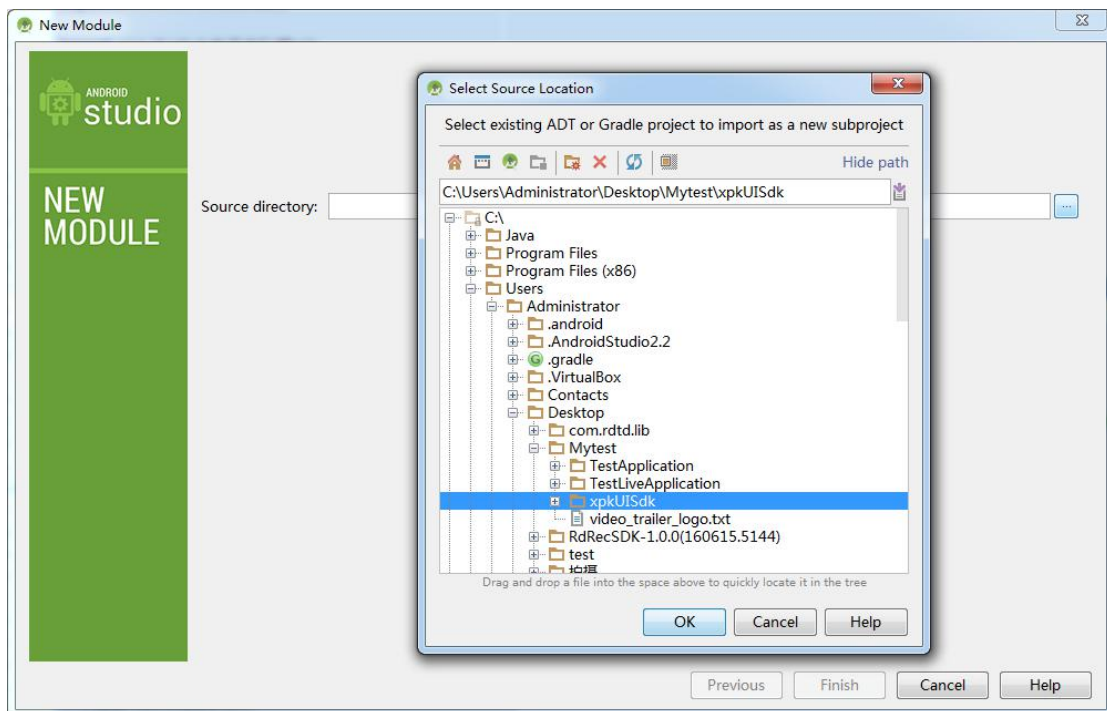
### 3.2 注册申请 AppKey 和 APPSECRET

- 1、 登录 <http://www.rdsdk.com> 注册用户
- 2、 登录注册好的用户
- 3、 进入视频云管理 点击(新增)获取应用的 appkey 、appsecret

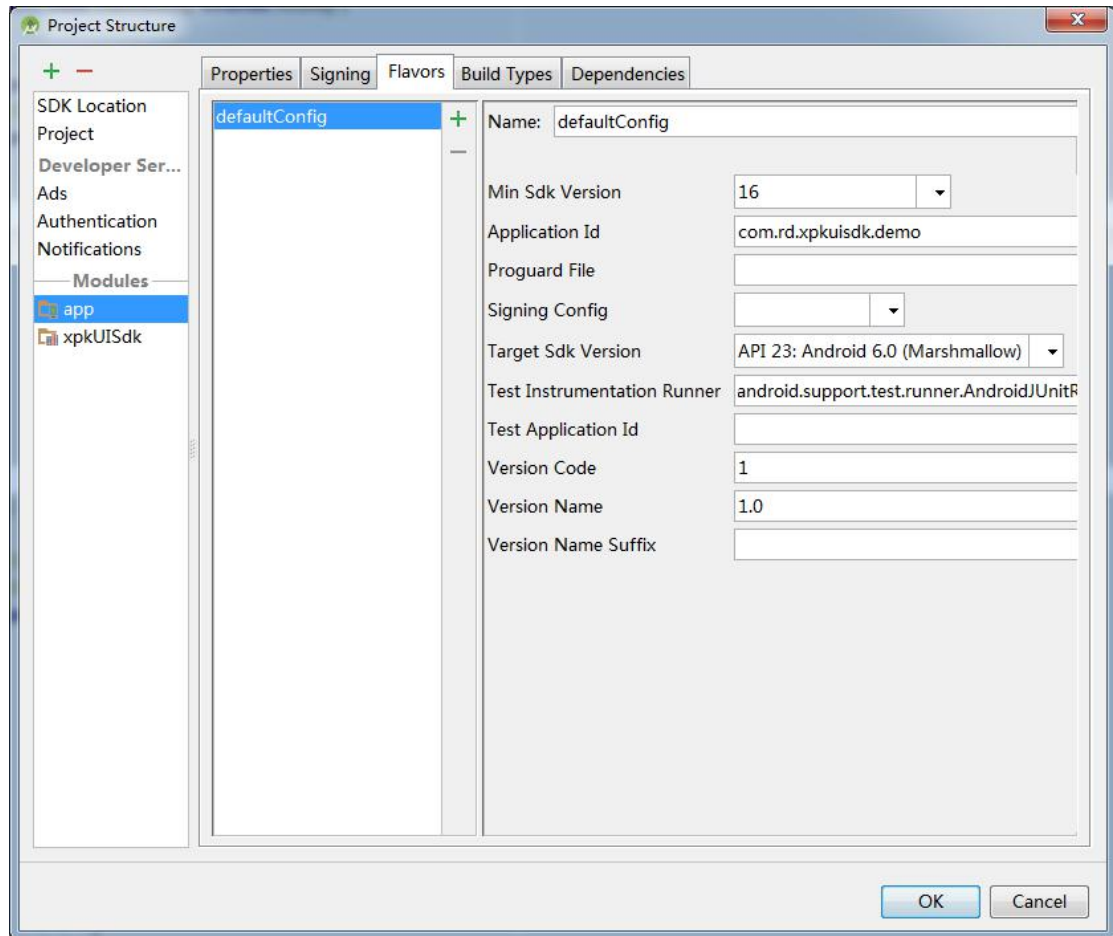
## 3.3 下载并导入 SDK

### 3.3.1 Android Studio 导入 xpkUISdk

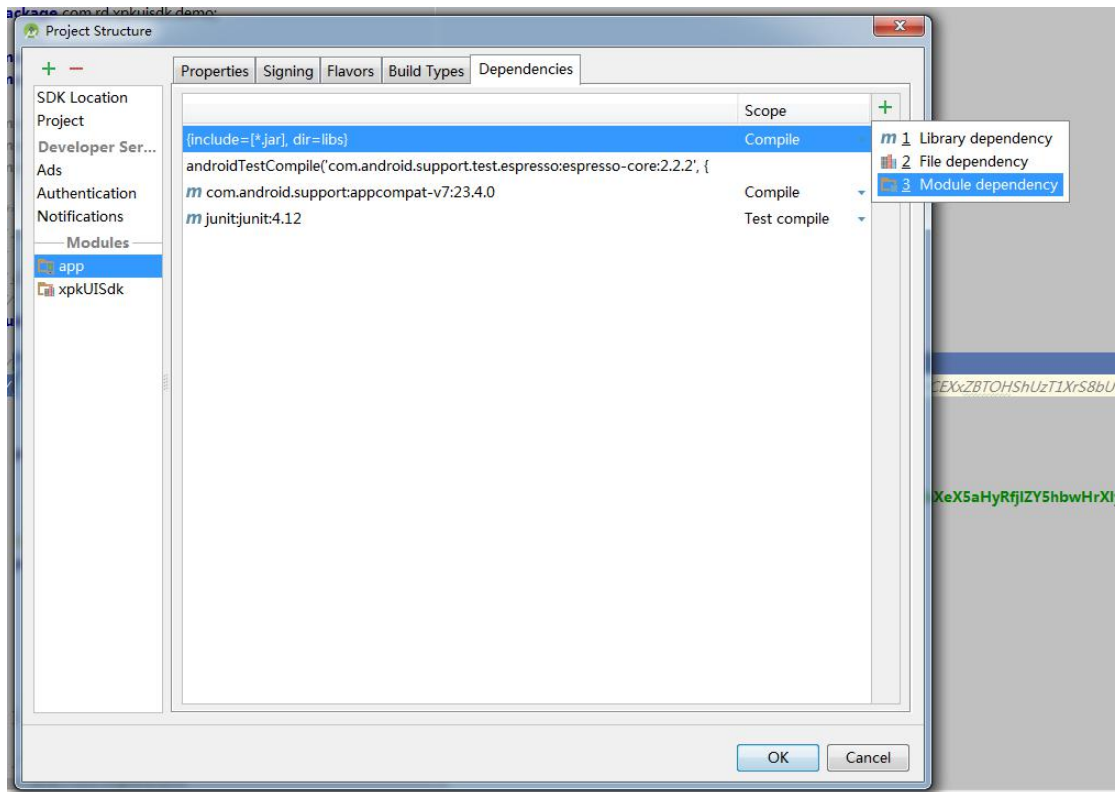
点击 File--->Import Module，选择路径，填写“Module name”，默认为 xpkUISdk，点击“finish”。



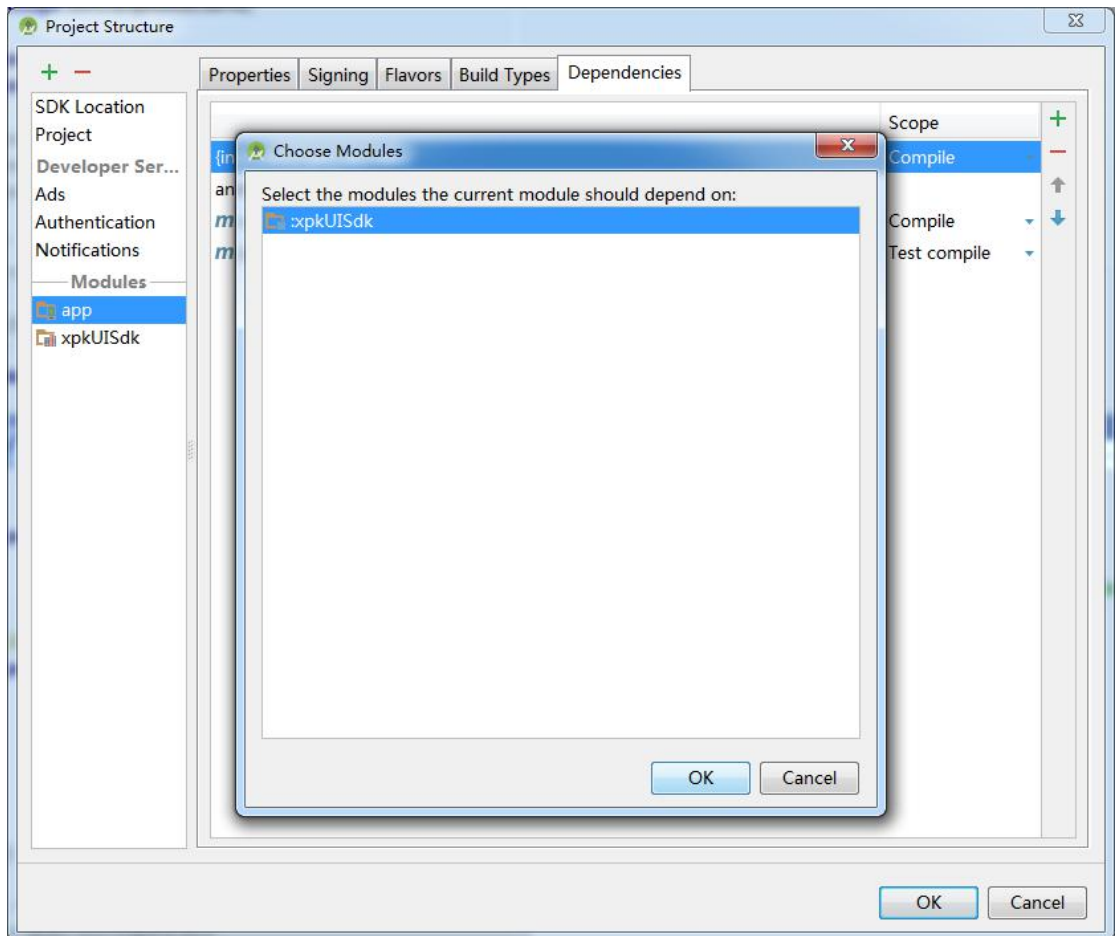
之后点击 File--->Project Structure, 选择你自己工程的 Module, 在 Flavors 选项卡中设置 Min Sdk Version 不低于 16



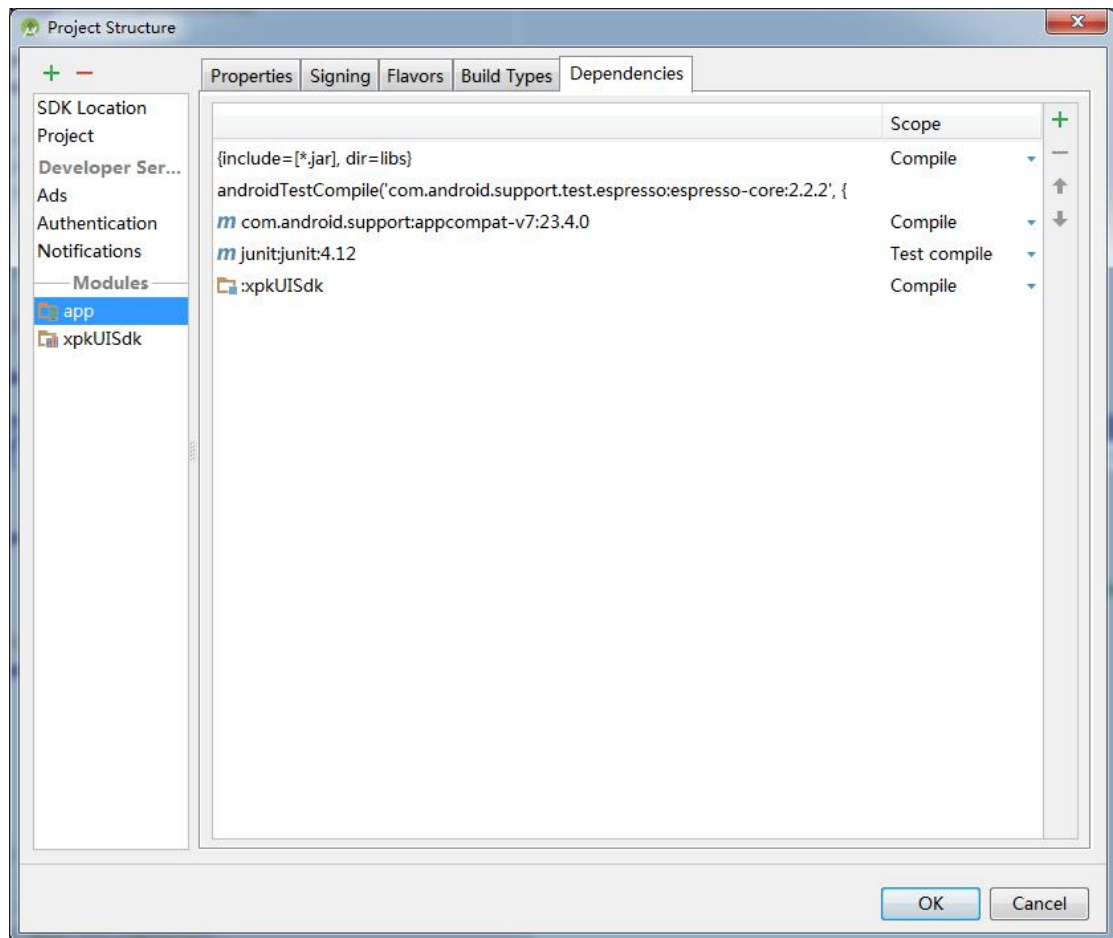
然后切换到 Dependencies 选项卡并点击右侧 “+” 号在弹出的下拉菜单选择 Module Dependency 点击。



弹出的小窗口中选择 SDK 相关的 Modules, “OK”



添加成功后点击 Project Structure 设置界面的“OK”完成导入工作。



## 关于集成 NDK SO 库引发冲突解决办法如下:

### 步骤一: sdk library 模块过滤

```
android {  
    compileSdkVersion 19  
    buildToolsVersion "23.0.2"  
    packagingOptions {  
        //过滤掉本地libs/armeabi和libs/armeabi-v7a/中的部分so，以jcenter  
        exclude 'lib/armeabi/libRdBase.so'  
        exclude 'lib/armeabi/libLiveRecorder.so'  
        exclude 'lib/armeabi/libRecorderKernel.so'  
        exclude 'lib/armeabi-v7a/libRdBase.so'  
        exclude 'lib/armeabi-v7a/libLiveRecorder.so'  
        exclude 'lib/armeabi-v7a/libRecorderKernel.so'  
    }  
}
```

### 步骤二: application 模块中过滤

编辑 SDK NDK 包含以下架构的 SO 库:

- armeabi-v7a

建议在 Module 的 *build.gradle* 文件中使用 NDK 的“*abiFilter*”配置, 设置支持的 SO 库架构。

如果在添加“*abiFilter*”之后 *Android Studio* 出现以下提示:

NDK integration is deprecated in the current plugin. Consider trying the new experimental plugin.

则在项目根目录的 *gradle.properties* 文件中添加:

```
android.useDeprecatedNdk=true
```

在集成 NDK SO 库时, 请注意只保留支持的架构 SO 库, 参考截图配置

```
defaultConfig {  
    minSdkVersion 16  
    targetSdkVersion 23  
    ndk {  
        abiFilters "armeabi", "armeabi-v7a"  
    }  
}
```



## Android studio 上绑定 SDK javadoc 文档

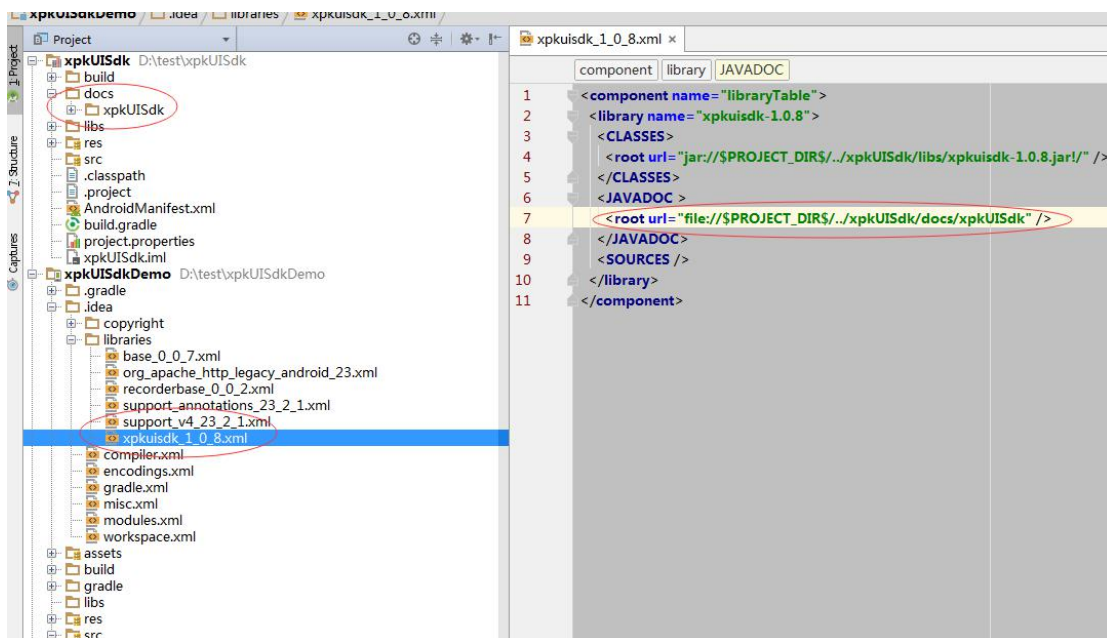
Android Studio 中使用 SDK jar 包时, 会在\$PROJECT\_DIR\$/idea/libraries 目录中生成 xpkuisdk\_\$VERSION\$.xml 文件, 添加以下配置:

```
<JAVADOC >
  <root url="file://$PROJECT_DIR$/../xpkUISdk/docs/xpkUISdk" />
</JAVADOC>
```

其中\$PROJECT\_DIR\$为项目根目录; \$VERSION\$代表 SDK 版本, 当前为 1.0.8, 则该 XML 文件则为 xpkuisdk\_1\_0\_8.xml;

当前配置演示是以 SDK demo 项目进行的, 需要根据实际集成环境进行修改。

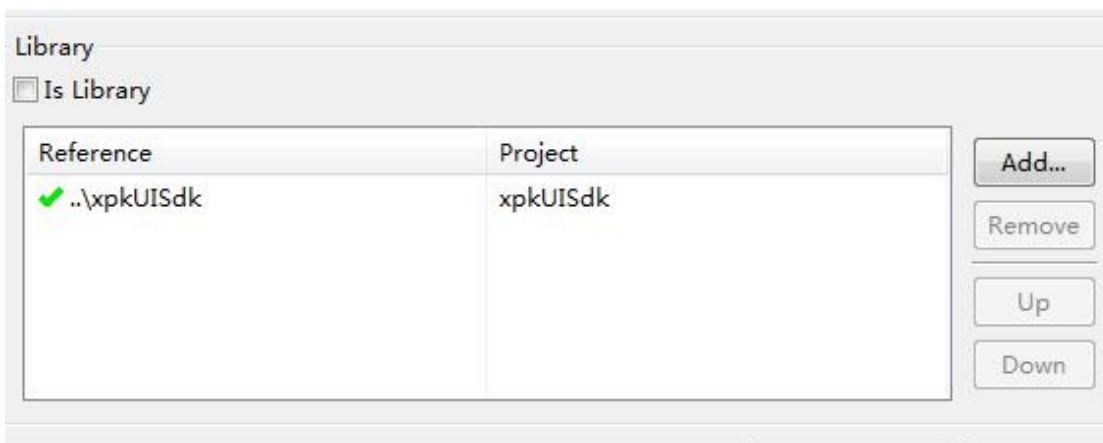
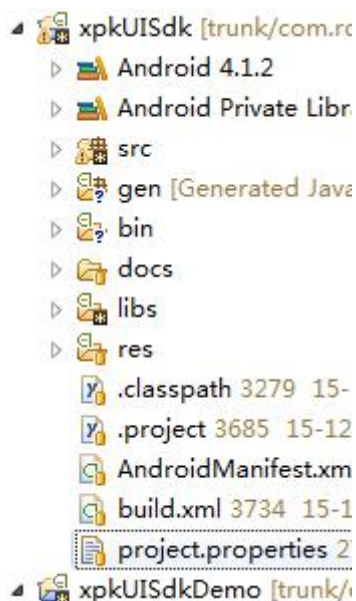
参考如下截图:



添加完 JAVADOC 后, 将是如下结果:



### 3.3.2 Eclipse 导入 xpkUISdk



### 3.3.3 准备 AndroidManifest.xml (权限, 注册 Activity)

a. 添加权限:

```
<!-- 必要权限开始 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<!-- 需要支持后台或休眠后保存输出时, 需添加的权限 -->
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

b. 注册必要的 Activity:

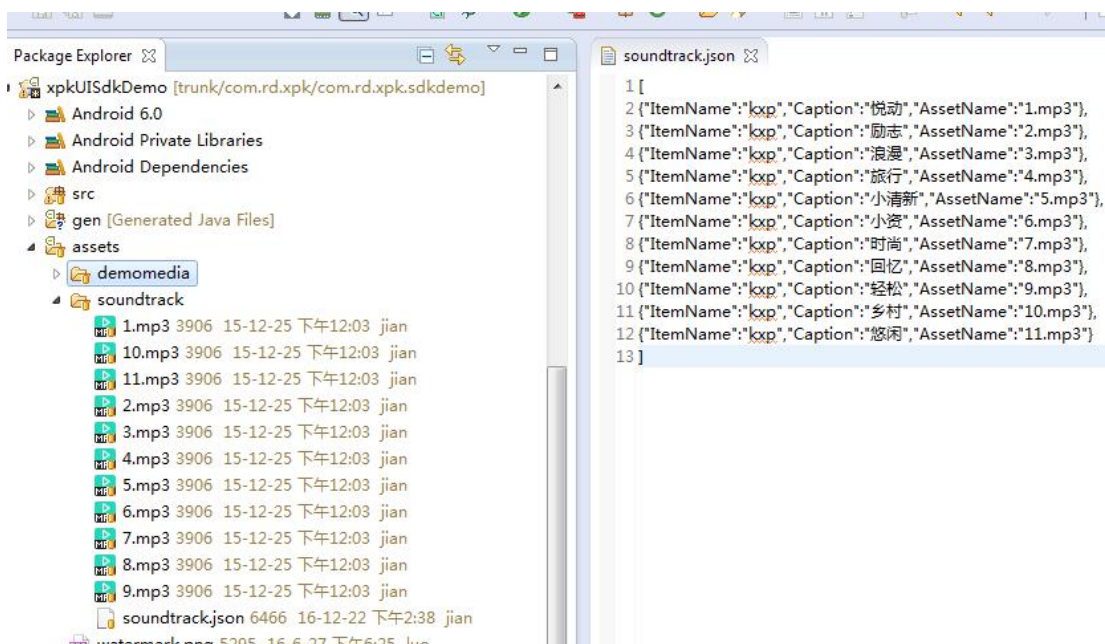
```
<activity
    android:name="com.rd.xpkuisdk.SelectMediaActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait" />
<activity
    android:name="com.rd.xpkuisdk.VideoPreviewActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait" />
<activity
    android:name="com.rd.xpkuisdk.EditPreviewActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait" />
<activity
    android:name="com.rd.xpkuisdk.CropVideoActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait" />
<activity
    android:name="com.rd.xpkuisdk.VideoEditActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateAlwaysHidden|adjustResize" />
<activity
    android:name="com.rd.xpkuisdk.CropVideoVerActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait" />
<activity
    android:name="com.rd.xpkuisdk.SpeedPreviewActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait" />
<activity
    android:name="com.rd.xpkuisdk.ImageDurationActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait" />
<activity
    android:name="com.rd.xpkuisdk.ExtPhotoActivity"
    android:configChanges="keyboardHidden"
    android:screenOrientation="portrait"
    android:windowSoftInputMode="stateAlwaysHidden|adjustResize" />
```

如果不想影响当前项目的 AndroidManifest.xml, Eclipse 下可以合并 xpkUISdk 中的配置, 修改 project.properties 文件如下:

```
android.library.reference.1=../xpkUISdk
manifestmerger.enabled=true
```



### 3.3.4 替换内置配音



替换对应音频文件和 soundtrack.json

### 3.3.5 调用 sdk 初始化

```
public class AppImpl extends Application {
    /**
     * 已获取的AppKey
     */
    public static final String APP_KEY = "27eb1f31f0f52501";
    /**
     * 已获取的AppSecret
     */
    public static final String APP_SECRET = "3ac8a8b40ce5bd5c37642978ef097731TI

    @Override
    public void onCreate() {
        super.onCreate();
        initXpkSdk();
    }

    /**
     * 初始化SDK
     */
    public void initXpkSdk() {
        // 这里确定是否启用日志,在调试阶段可开启该选项,方便定位问题。打包为发行版时,此启用无效
        XpkSdk.enableDebugLog(true);
        // 自定义根目录,如果为空则默认为/sdcard/xpk
        String strCustomPath = Environment.getExternalStorageDirectory()
            + "/xpk";
        // sdk初始
        XpkSdk.init(AppImpl.this, strCustomPath, APP_KEY, APP_SECRET,
            new SdkHandler().getCallBack());
        // 自定义Crash handler,实际项目可不加入
        MyCrashHandler.getInstance().init(this);
    }
}
```

初始化成功后才能正常使用其他功能接口,还要注意系统权限的申请,否则 SDK 在系统 api 级别  $\geq 23$  (android 6.0+) 下不能正常使用

```
// 请求code:读取外置存储
private static final int REQUEST_CODE_READ_EXTERNAL_STORAGE_PERMISSIONS = 1;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // setContentView(R.layout.activity_main);

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M
        && !XpkSdk.isInitialized()) {
        if (checkSelfPermission(Manifest.permission.READ_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED) {
            requestPermissions(
                new String[] { Manifest.permission.READ_EXTERNAL_STORAGE },
                REQUEST_CODE_READ_EXTERNAL_STORAGE_PERMISSIONS);
        }
    }
}
```

```
@SuppressWarnings("NewApi")
@Override
public void onRequestPermissionsResult(int requestCode,
                                       String[] permissions, int[] grantResults) {
    switch (requestCode) {
        case REQUEST_CODE_READ_EXTERNAL_STORAGE_PERMISSIONS: {
            for (int i = 0; i < permissions.length; i++) {
                if (grantResults[i] == PackageManager.PERMISSION_GRANTED) {
                    if (!XpkSdk.isInitialized()) {
                        ((AppImpl) getApplication()).init();
                    }
                } else {
                    Toast.makeText(this, "未允许读写存储！", Toast.LENGTH_SHORT).show();
                    finish();
                }
            }
            break;
            default: {
                super.onRequestPermissionsResult(requestCode, permissions,
                                                  grantResults);
            }
        }
    }
}
```

### 3.3.6 指定配置参数

SDK 初始化成功之后, 需要部分自定义功能时需指定配置参数, 不指定时也不会影响使用, 只是按照默认配置进行调用。

#### a. 界面配置类示例:

```
UIConfiguration uiConfig = new UIConfiguration.Builder()
    //设置横竖屏(支持横屏, 竖屏, 自动)
    .setOrientation(orientation)
    //设置是否使用自定义相册 (传 false 将调用秀拍客相册)
    .useCustomAlbum(useCustomAlbum)
    //设置是否隐藏配音配乐
    .hideMusic(hide)
    //设置是否隐藏片段编辑
    .hidePartEdit(hide)
    .get();
```

若选择自定义相册, 将会回调以下接口:

```
com.rd.xpkuisdk.ISdkCallBack.getVideo(context) //添加视频
com.rd.xpkuisdk.ISdkCallBack.getPhoto(context) //添加图片
```

在上述方法中调用自定义相册, 选择视频或图片完成后, 将它们的路径通过调用以下方法完成选择:

```
com.rd.xpkuisdk.XpkSdk.onCustomizeAlbum(
    Context context, //应用上下文
    ArrayList<String> medialist //视频或图片路径集合
)
```

#### b. 导出视频配置类示例:

```
ExportConfiguration exportConfig = new ExportConfiguration.Builder()
    //设置保存导出视频时的码流 单位 M
    .setVideoBitRate(bitRate)
    //设置导出视频时长 单位 ms (设置为 0 或默认不设置将导出完整视频)
    .setVideoDuration(exportVideoDuration)
    //设置导出视频保存路径 (设 null 将保存到默认路径)
    .setSavePath(savePath)
    //设置视频片尾图片路径 (设 null 将没有片尾)
    .setTrailerPath(trailerPath)
    //设置视频片尾时长 单位 ms (默认为 2000ms)
    .setTrailerDuration(trailerDuration)
    .get();
```

#### c. 录制拍摄配置类示例:

```
CameraConfiguration cameraConfig = new CameraConfiguration.Builder()
    //设置限制录制的视频时长 单位为秒 0 代表没有时间限制
    .setVideoMaxTime(maxTime)
```

```
//确定是否支持拍照模式下点击拍照按钮立即返回
.setTakePhotoReturn(taskPhotoReturn)
//设置录制时启动默认页面方式
.setCameraUIType(type)
//录制时静音
.setAudioMute(mute)
.get();
```

其中 CameraUIType 可选常量定义如下:

```
//代表默认启动 16: 9 宽屏录制界面并可切换到 1: 1 界面
CameraConfiguration.WIDE_SCREEN_CAN_CHANGE
//代表默认启动 1: 1 界面并可切换到 16: 9 宽屏录制界面
CameraConfiguration.SQUARE_SCREEN_CAN_CHANGE
//代表默认启动 1: 1 界面并不可切换到 16: 9 宽屏录制界面
CameraConfiguration.ONLY_SQUARE_SCREEN
```

指定配置参数例子:

```
XpkSdk.getXpkSdkService().initConfiguration(
    exportConfig, //导出配置
    uiConfig, //界面配置
    cameraConfig //录制拍摄配置 );
```

## 3.4 录制视频

### 3.4.1 录制视频或拍照

进入该界面调用以下接口:

```
public static void com.rd.xpkuisdk.onXpkCamera(
    Context context, //应用上下文
    boolean gotoEdit //是否录制完成后, 进入高级编辑界面
)
```

录制视频或录制视频并编辑导出完成后将会回调以下接口:

```
com.rd.xpkuisdk.ISdkCallBack.getVideoPath(
    context, //应用上下文
    exportType, //编辑导出或录制完成后的回调类型
    videoPath //视频路径
)
```

其中回调类型 exportType 参数值定义如下:

```
XpkSdk.XPK_CAMERA_EDIT_EXPORT //gotoEdit 为 true 时, 代表录制完成并编辑导出后的返回
XpkSdk.XPK_CAMERA_EXPORT //gotoEdit 为 false 时, 代表只是录制完成后的返回
```

## 3.5 编辑视频

### 3.5.1 进入 sdk 的功能界面 (选择媒体资源)

```
void com.rd.xpkuisdk.XpkSdk.onXpkVideo(Context context//应用上下文)
```

### 3.5.2 进入 sdk 的编辑界面 (直接进入编辑界面)

```
void com.rd.xpkuisdk.XpkSdk.onXpkEdit(  
    Context context, //应用上下文  
    ArrayList<String> list //媒体路径集合(可用的图片或视频的路径)  
)
```

### 3.5.3 进入 sdk 的截取界面 (直接进入截取界面)

通过以下接口可以直接进入截取界面：

```
void com.rd.xpkuisdk.XpkSdk.onXpkTrimVideo(  
    Context context, //应用上下文  
    String videoPath, //媒体路径  
    String savePath, //视频截取后的保存路径 设置 null 将会保存到默认路径.../xpk/xpkVideos/  
    String title, //标题栏文字内容 设置 null 标题将会设为“截取”  
    int titleBarColor, //标题栏背景色  
    String buttonCancelText, //取消按钮文字 设置 null 将会显示图标 ×  
    String buttonConfirmText, //确认按钮文字 设置 null 将会显示图标 ✓  
    int buttonColor //按钮背景色  
);
```

完成截取需分为两个步骤，第一步是先确认截取返回方式，第二步通过返回方式将最终的结果通过对应的回调接口进行返回。

#### 第一步：确认截取返回方式

确认截取返回方式时将会回调以下接口：

```
com.rd.xpkuisdk.ISdkCallBack.getVideoTrim(  
    context, //应用上下文
```



```
exportType //回调类型  
)
```

其中回调类型 `exportType` 参数值定义如下：

```
XpkSdk.XPK_TRIMVIDEO_EXPORT //普通截取  
  
XpkSdk.XPK_TRIMVIDEO_DURATION_EXPORT //定长截取
```

## 第二步：设置截取返回方式

通过以下接口完成截取返回方式设置：

```
void com.rd.xpkuisdk.XpkSdk.onVideoTrim(  
    Context context, //应用上下文  
    int returnType //需要给定的截取返回方式  
)
```

其中截取返回方式 `returnType` 参数值定义如下：

```
RETURN_TRIM_VIDEO = 0//返回截取后视频
```

完成截取并导出视频后将会回调以下接口：

```
com.rd.xpkuisdk.ISdkCallBack.getVideoPath(  
    context, //应用上下文  
    exportType, //截取并导出视频回调类型  
    videoPath//截取并导出后的视频路径  
)
```

其中回调类型 `exportType` 参数值定义如下：

```
XpkSdk.XPK_TRIMVIDEO_EXPORT //普通截取  
  
XpkSdk.XPK_TRIMVIDEO_DURATION_EXPORT //定长截取
```

```
RETURN_TRIM_TIME = 1//返回截取时间
```

截取界面将会返回截取视频的开始和结束时间 并执行以下回调：

```
com.rd.xpkuisdk.ISdkCallBack.getVideoTrimTime(  
    context, //应用上下文  
    exportType, //返回截取时间回调类型  
    startTime, //开始时间(单位 ms)  
    endTime //结束时间(单位 ms)  
)
```

其中回调类型 `exportType` 参数值定义如下：

```
XpkSdk.XPK_TRIMVIDEO_EXPORT //普通截取  
  
XpkSdk.XPK_TRIMVIDEO_DURATION_EXPORT //定长截取
```

### 3.5.4 导出视频

编辑完成后，根据导出配置生成视频，导出完成后将会回调以下接口：

```
com.rd.xpkuisdk.ISdkCallBack.getVideoPath(  
    context, //应用上下文  
    exportType, //导出视频回调类型  
    videoPath //视频路径  
)
```

其中回调类型 exportType 参数值定义如下：

XpkSdk.XPK\_EDIT\_EXPORT //普通编辑导出完成后的返回

XpkSdk.XPK\_CAMERA\_EDIT\_EXPORT //代表录制完成并编辑导出后的返回

## 3.6 响应退出

应用结束时，需调用一下 SDK 的退出接口 XpkSdk.exitApp,一般是在最后退出 Activity 中的 onDestroy 方法内，具体实现参见 demo。

## 3.7 使用自定义播放器

com.rd.xpkuisdk.ui.XpkVideoView

### 3.7.1 在布局文件中添加播放器

```
<com.rd.xpkuisdk.XpkVideoView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```

### 3.7.2 播放器初始化

```
void com.rd.xpkuisdk.XpkVideoView.setPlayerListener(XpkPlayerListener listener)
```

```
/**
 * 播放器listener
 *
 * @author scott,abreal
 */
public interface XpkPlayerListener {
    /**
     * 播放器已经就绪
     *
     * @param mediaPlayerControl
     */
    void onPlayerPrepare(MediaPlayerControl mediaPlayerControl);

    /**
     * 播放器出现错误
     *
     * @param mediaPlayerControl
     * @param what
     *         错误号
     * @param extra
     *         错误扩展信息
     * @return 返回true代表已处理错误
     */
    boolean onPlayerError(MediaPlayerControl mediaPlayerControl, int what,
        int extra);

    /**
     * 播放器播放已结束
     *
     * @param mediaPlayerControl
     */
    void onPlayerCompletion(MediaPlayerControl mediaPlayerControl);

    /**
     * 获取到当前播放位置
     *
     * @param mediaPlayerControl
     * @param nPosition
     *         当前播放位置(ms)
     */
    void onGetCurrentPosition(MediaPlayerControl mediaPlayerControl,
        int nPosition);
}
```

### 3.7.3 播放器接口

setVideoPath(String videoPath) //设置视频路径

boolean isPlaying() //判断播放器是否正在播放

void start() //开始播放

void pause() //暂停播放

int getCurrentPosition() //获取当前播放进度

```
void seekTo(int position) //跳到指定播放位置(单位 ms)
```

```
int getDuration() //获取媒体时长, 单位 ms
```

## 3.8 分享视频

暂无

## 3.9 视频水印

添加水印的方法, 参见截图, 将一个 PNG 图片放在 assets 目录下, 固定名称为 **watermark.png**



## 3.10 打包混淆

需要在 proguard.cfg 文件中添加如下配置项:

```
-dontwarn com.rd.**
```

```
-keep class com.rd.** { *; }
```