安卓6.0到9.0适配

雾里看花六月天

作者：雾里看花六月天

链接：

https://www.jianshu.com/p/23b8ae9d5a95

1

前言

大家还记得Android 6.0权限适配的泪水吗？而现在谷歌已经出了Android P的稳定版，而且谷歌粑粑，为了大家能给辛苦熬夜加班，特地的和个大市场合作，要强制推出9.0的适配，而近期在下不才，为了报着多踩坑的心态，做了一下7.0～9.0的适配，脸颊也是老泪两行。

2

安卓6.0的适配

2.1 怎么适配

● 在6.0所有权限都需要申请？

曰：当然不是。只有属于危险权限的才需要申请。危险权限看下表1-2

● 那危险权限也很多啊，也要一个个申请？

曰：当然不是。你看看下面的表，都分好组了（9组），对于同一组内的权限，只要有一个被同意，其他的都会被同意。

2.2 列举权限的分组

表1-2危险权限分组

分组	名字	分割线
PHONE	android.permission.READ_PHONE_STATE	
	android.permission	

	.CALL_PHONE	
	android.permission.READ_CALL_LOG	
	android.permission.ADD_VOICEMAIL	
	android.permission.WRITE_CALL_LOG	
	android.permission.USE_SIP	
	android.permission.PROCESS_OUTGOING_CALLS	
CALENDAR	android.permission.READ_CALENDAR	
	android.permission.WRITE_CALENDAR	
CAMERA	android.permission.CAMERA	
CONTACTS	android.permission.READ_CONTACTS	
	android.permission.WRITE_CONTACTS	
	android.permission.GET_ACCOUNTS	
LOCATION	android.permission.ACCESS_FINE_LOCATION	
	android.permission.ACCESS_COARSE_LOCATION	
MICROPHONE	android.permission.RECORD_AUDIO	
SENSORS	android.permission.BODY_SENSORS	
SMS	android.permission.SEND_SMS	
	android.permission.RECEIVE_SMS	
	android.permission	

	.READ_SMS	
	android.permission.RECEIVE_WAP_PUSH	
	android.permission.RECEIVE_MMS	
STORAGE	android.permission.READ_EXTERNAL_STORAGE	
	android.permission.WRITE_EXTERNAL_STORAGE	

```
<!--  危险权限  start  -->

<!--PHONE-->

<uses-permission  android:name="android.permission.READ_PHONE_STATE"/>

<uses-permission  android:name="android.permission.CALL_PHONE"/>

<uses-permission  android:name="android.permission.READ_CALL_LOG"/>

<uses-permission  android:name="android.permission.ADD_VOICEMAIL"/>

<uses-permission  android:name="android.permission.WRITE_CALL_LOG"/>

<uses-permission  android:name="android.permission.USE_SIP"/>

<uses-permission  android:name="android.permission.PROCESS_OUTGOING_CALLS"/>

<!--CALENDAR-->

<uses-permission  android:name="android.permission.READ_CALENDAR"/>

<uses-permission  android:name="android.permission.WRITE_CALENDAR"/>

<!--CAMERA-->

<uses-permission  android:name="android.permission.CAMERA"/>

<!--CONTACTS-->

<uses-permission  android:name="android.permission.READ_CONTACTS"/>

<uses-permission  android:name="android.permission.WRITE_CONTACTS"/>

<uses-permission  android:name="android.permission.GET_ACCOUNTS"/>

<!--LOCATION-->

<uses-permission  android:name="android.permission.ACCESS_FINE_LOCATION"/>

<uses-permission  android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

```
<!--MICROPHONE-->
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<!--SENSORS-->
<uses-permission android:name="android.permission.BODY_SENSORS"/>
<!--SMS-->
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_WAP_PUSH"/>
<uses-permission android:name="android.permission.RECEIVE_MMS"/>
<!--STORAGE-->
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<!-- 危险权限 Permissions end -->
```

以上是列出9组需要动态申请的权限，建议自己统一封装成一个工具类，这里就不细说了。

3

安卓7.0的适配

3.1 应用间共享文件

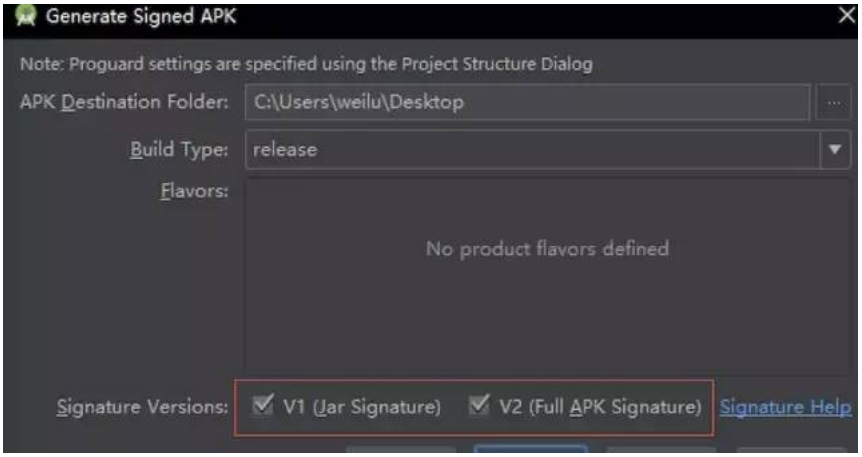
在targetSdkVersion大于等于的24的App中，但是我们没有去适配7.0。那么在调用安装页面，或修改用户头像操作时，就会失败。那么就需要你去适配7.0或是将targetSdkVersion改为24以下（不推荐）。适配的方法这里就不细讲，大家可以看鸿洋的

Android 7.0 行为变更 通过FileProvider在应用间共享文件这篇文章

https://blog.csdn.net/lmj623565791/article/details/72859156

3.2 APK signature scheme v2

> Android 7.0 引入一项新的应用签名方案 APK Signature Scheme v2，它能提供更快的应用安装时间和更多针对未授权 APK 文件更改的保护。在默认情况下，Android Studio 2.2 和 Android Plugin for Gradle 2.2 会使用 APK Signature Scheme v2 和传统签名方案来签署您的应用。



1）只勾选v1签名就是传统方案签署，但是在7.0上不会使用V2安全的验证方式。

2）只勾选V2签名7.0以下会显示未安装，7.0上则会使用了V2安全的验证方式。

3）同时勾选V1和V2则所有版本都没问题。

3.3 org.apache不支持问题

```
//  build.gradle里面加上这句话

defaultConfig  {

        useLibrary  'org.apache.http.legacy'

}
```

4

安卓8.0的适配

4.1　安卓8.0中PHONE权限组新增两个权限

```
ANSWER_PHONE_CALLS：允许您的应用通过编程方式接听呼入电话。要在您的应用中处理呼入电话，您可以使用  acceptRingingCall()  函数。

READ_PHONE_NUMBERS  ：权限允许您的应用读取设备中存储的电话号码。
```

4.2 通知适配

安卓8.0中，为了更好的管制通知的提醒，不想一些不重要的通知打扰用户，新增了通知渠道，用户可以根据渠道来屏蔽一些不想要的通知.

兼容的代码

```
/**

 *  安卓8。0通知的兼容类哦，

 *  NotifyCompatYc       yc  :  是雨辰的简写，谢谢哦，嘿嘿  ----高贵的子信

 */
```

```
public class NotifyCompatYc {

    public static final String QFMD_CHANNEL_ID = "com.oms.mingdeng";

    public static final String QFMD_CHANNEL_NAME = "祈福明燈";

    public static final String LJMS_DEFAULT_CHANNEL_NAME = "靈機妙算";

    public static final String LJMS_CHANNEL_ID = "com.oms.mmcnotity";

    public static final String XYS_CHANNEL_ID = "com.oms.xuyuanshu";

    public static final String XYS_CHANNEL_NAME = "許願樹";

    public static void setONotifyChannel(NotificationManager manager, NotificationCompat.Builder builder, String channeId, String channelName){

        if (TextUtils.isEmpty(channeId)||TextUtils.isEmpty(channelName)){

            L.e("NotifyCompatYc:  ".concat("安卓8.0的通知兼容库中 channeId 与 channelName 不能为empty"));

        }

        if (Build.VERSION.SDK_INT >= 26) {

            //第三个参数设置通知的优先级别

            NotificationChannel channel =

                    new NotificationChannel(channeId, channelName, NotificationManager.IMPORTANCE_DEFAULT);

            channel.canBypassDnd();//是否可以绕过请勿打扰模式

            channel.canShowBadge();//是否可以显示icon角标

            channel.enableLights(true);//是否显示通知闪灯

            channel.enableVibration(true);//收到小时时震动提示

            channel.setBypassDnd(true);//设置绕过免打扰

            channel.setLockscreenVisibility(NotificationCompat.VISIBILITY_SECRET);

            channel.setLightColor(Color.RED);//设置闪光灯颜色

            channel.getAudioAttributes();//获取设置铃声设置

            channel.setVibrationPattern(new long[]{100, 200, 100});//设置震动模式

            channel.shouldShowLights();//是否会闪光

            if (manager != null) {

                manager.createNotificationChannel(channel);

            }

            if (builder != null) {

                builder.setChannelId(channeId);//这个id参数要与上面channel构建的第一个参数对应

            }
```

```
            }
        }
        public static void setONotifyChannel(NotificationManager manager,  String channeId, String channelName) {
            setONotifyChannel(manager,null,channeId,channelName);
        }
        public static Notification getNotification(Context context, String channelId) {
            NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(context, channelId);
            Notification notification = notificationBuilder.setOngoing(true)
                    .setSmallIcon(R.drawable.ic_launcher)
                    .setPriority(NotificationManager.IMPORTANCE_MIN)
                    .setCategory(Notification.CATEGORY_SERVICE)
                    .build();
            return notification;
        }
}
/**
 * 作者：子川
 * 时间：2017/2/9:18:03
 * 邮箱：lantianhua@linghit.com
 * 说明：TODO
 */
public class NotifyManager {
    //  单例开始
    private volatile static NotifyManager INSTANCE;
    private NotifyManager(Context context) {
        initNotifyManager(context);
    }
    public static NotifyManager getInstance(Context context) {
        if (INSTANCE == null) {
            synchronized (NotifyManager.class) {
                if (INSTANCE == null) {
```

```
                                INSTANCE = new NotifyManager(context);
                            }
                        }
                    }
                    return INSTANCE;
                }
                // 单例结束
                private NotificationManager manager;
            // NotificationManagerCompat
                private NotificationCompat.Builder builder;
                //初始化通知栏配置
                private void initNotifyManager(Context context) {
                    context = context.getApplicationContext();
                    manager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
                    // 如果存在则清除上一个消息
    //                manager.cancel(news_flag);
                    builder = new NotificationCompat.Builder(context,NotifyCompatYc.QFMD_CHANNEL_ID);
                    NotifyCompatYc.setONotifyChannel(manager,builder,NotifyCompatYc.QFMD_CHANNEL_ID,NotifyCompatYc.QFMD_CHANNEL_NAME);
                    // 设置标题
                    builder.setContentTitle(context.getResources().getString(R.string.qfmd_notify_title1));
                    // 状态栏的动画提醒语句
                    builder.setTicker(context.getResources().getString(R.string.qfmd_notify_ticker));
                    // 什么时候提醒的
                    builder.setWhen(System.currentTimeMillis());
                    // 设置通知栏的优先级
                    builder.setPriority(Notification.PRIORITY_DEFAULT);
                    // 设置点击可消失
                    builder.setAutoCancel(true);
                    // 设置是否震动等
                    builder.setDefaults(Notification.DEFAULT_VIBRATE);
                    // 设置icon
```

```
            builder.setSmallIcon(R.drawable.lingji_icon);
            //  设置点击意图
            Intent intent = new Intent(context, GongdenggeActivity.class);
            Bundle bundle = new Bundle();
            bundle.putBoolean(Contants.INTENT_GOTO_MYLMAP, true);
            intent.putExtras(bundle);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            PendingIntent pendingIntent = PendingIntent.getActivity(context, 230, intent, PendingIntent.FLAG_UPDATE_CURRENT);
            builder.setContentIntent(pendingIntent);
        }
        /**
         * 显示祈福明灯过期通知
         */
        public void showQiFuLampOutOfDateNotify(Context context) {
            //  设置内容
            builder.setContentText(context.getResources().getString(R.string.qfmd_notify_content1));
            manager.notify(13251, builder.build());
        }
        public void showQiFuLampBlessNotify(Context context) {
            builder.setContentText(context.getResources().getString(R.string.qfmd_notify_content2));
            manager.notify(13255, builder.build());
        }
}
```

4.3 安装APK

首先在AndroidManifest文件中添加安装未知来源应用的权限：

```
<uses-permission android:name="android.permission.REQUEST_INSTALL_PACKAGES"/>
```

这样系统会自动询问用户完成授权。当然你也可以先使用 canRequestPackageInstalls()查询是否有此权限，如果没有的话使用Settings.ACTION_MANAGE_UNKNOWN_APP_SOURCES这个action将用户引导至安装未知应用权限界面去授权。

```
private static final int REQUEST_CODE_UNKNOWN_APP = 100;
private void installAPK(){
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
```

```
                boolean hasInstallPermission = getPackageManager().canRequestPackageInstalls();
                if (hasInstallPermission) {
                        //安装应用
                } else {
                        //跳转至“安装未知应用”权限界面，引导用户开启权限
                        Uri selfPackageUri = Uri.parse("package:" + this.getPackageName());
                        Intent intent = new Intent(Settings.ACTION_MANAGE_UNKNOWN_APP_SOURCES, selfPackageUri);
                        startActivityForResult(intent, REQUEST_CODE_UNKNOWN_APP);
                }
        }else {
                //安装应用
        }
}
//接收“安装未知应用”权限的开启结果
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST_CODE_UNKNOWN_APP) {
                installAPK();
        }
}
```

4.4 SharedPreferences闪退

```
SharedPreferences read = getSharedPreferences(RELEASE_POOL_DATA, MODE_WORLD_READABLE);
//MODE_WORLD_READABLE ：8.0以后不能使用这个获取，会闪退，修改成MODE_PRIVATE
```

4.5 SecurityException的闪退

问题原因：项目使用了ActiveAndroid，在 8.0 或 8.1 系统上使用 26 或以上的版本的 SDK 时，调用 ContentResolver 的 notifyChange 方法通知数据更新，或者调用 ContentResolver 的 registerContentObserver 方法监听数据变化时，会出现上述异常。

解决方案：

（1）在清单文件配置

```
<provider
```

```
            android:name="com.activeandroid.content.ContentProvider"
            android:authorities="com.ylmf.androidclient"
            android:enabled="true"
            android:exported="false">
</provider>
```

（2）去掉这个监听刷新的方法，改为广播刷新

4.6 静态广播无法正常接收

问题原因： Android 8.0 引入了新的广播接收器限制，因此您应该移除所有为隐式广播 Intent 注册的广播接收器

```
//setNotificationUri(cursor);
//给注销掉
/**  通知数据库发生变化*/
private  void  notifyChange  ()  {
        //mContext.getContentResolver().notifyChange(ORDER_URI,  null,  false);
        BroadcastController.sendChangeDataBroadCast(mContext);
}
```

解决方案：

使用动态广播代替静态广播

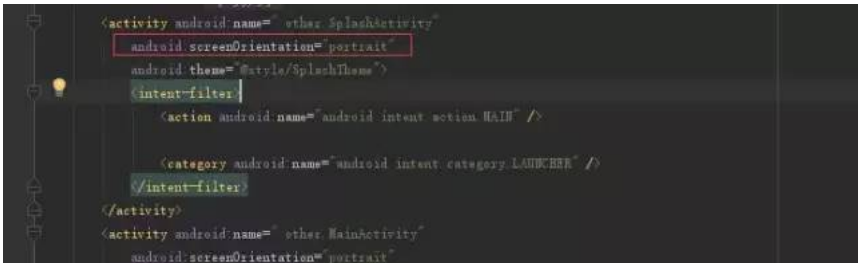
4.7 Caused by: java.lang.IllegalStateException: Only fullscreen opaque activities can request orientation

解决方案(1)：

问题原因： Android 8.0 非全屏透明页面不允许设置方向(8.1以上谷歌就修复去掉了这个限制)

解决方案：android:windowIsTranslucent设置为false

解决方案(2)：<去掉方向的设置>



解决方案（3）：

使用对话框、fragmentDialog、Popwindow的代替透明的Activity。

5

安卓9.0的适配

5.1 CLEARTEXT communication to life.115.com not permitted by network security policy

问题原因： Android P 限制了明文流量的网络请求，非加密的流量请求都会被系统禁止掉

解决方案：

在资源文件新建xml目录，新建文件

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true" />
</network-security-config>
```

清单文件配置：

```
<application
    android:networkSecurityConfig="@xml/network_security_config">
    <!--9.0加的，哦哦-->
    <uses-library
        android:name="org.apache.http.legacy"
        android:required="false" />
</application>
```

但还是建议都使用https进行传输

5.2 其他Api的修改

java.lang.IllegalArgumentException: Invalid Region.Op - only INTERSECT and DIFFERENCE are allowed

```
if (Build.VERSION.SDK_INT >= 26) {
    canvas.clipPath(mPath);
} else {
    canvas.clipPath(mPath, Region.Op.REPLACE);
}
```

总结：经过几天的踩坑，终于把targetSdkVersion升级到28，对于以上的经验，也许还存在某些疏漏的，也希望大家可以指正，补充，告诉，希望对你有一定的帮助，鄙人也很开心

历史适配文章：

Android O 适配详细指南

Android 刘海屏适配方案

快来了解下 Android P 兼容与适配

Android 8.0适配指北

推荐阅读：

非 UI 线程能调用 View.invalidate()？

Android关于Color你所知道的和不知道的一切

一些提高Android开发效率的经验和技巧

扫一扫　关注我的公众号

如果你想要跟大家分享你的文章，欢迎投稿~

┌(ˆ0ˆ)┘明天见！

阅读原文