MENU

Android6.0权限适配，比你想的还要简单（实践篇）

18 AUGUST 2016

前言

自从升级到Android M以来，最大的改变就是增加了运行时权限RuntimePermission，6.0以上的系统如果没有做适配，运行了targetSDK=23的App时就会报权限错误；当然如果你还没准备好适配权限，把targetSDK设置成小于23就ok了，不过适配是迟早的。

运行时权限

谷歌官方将权限分为了两类，一个是正常权限（Normal Permissions），这类权限不涉及用户隐私，是不需要用户进行授权的，比如访问网络，手机震动等。还有一类是危险权限（Dangerous Permissions），一般是涉及到用户隐私的，需要用户进行授权，比如操作SD卡的写入，相机，录音等。

Dangerous Permissions	
Permission Group	Permission
android.permission-group.CALENDAR	• android.permission.READ_CALENDAR • android.permission.WRITE_CALENDAR
android.permission-group.CAMERA	• android.permission.CAMERA
android.permission-group.CONTACTS	• android.permission.READ_CONTACTS • android.permission.WRITE_CONTACTS • android.permission.READ_PROFILE • android.permission.WRITE_PROFILE
android.permission-group.LOCATION	• android.permission.ACCESS_FINE_LOCATION • android.permission.ACCESS_COARSE_LOCATION
android.permission-group.MICROPHONE	• android.permission.RECORD_AUDIO
android.permission-group.PHONE	• android.permission.READ_PHONE_STATE • android.permission.CALL_PHONE • android.permission.READ_CALL_LOG • android.permission.WRITE_CALL_LOG • com.android.voicemail.permission.ADD_VOICEMAIL • android.permission.USE_SIP • android.permission.PROCESS_OUTGOING_CALLS
android.permission-group.SENSORS	• android.permission.BODY_SENSORS • android.permission.USE_FINGERPRINT
android.permission-group.SMS	• android.permission.SEND_SMS • android.permission.RECEIVE_SMS • android.permission.READ_SMS • android.permission.RECEIVE_WAP_PUSH • android.permission.RECEIVE_MMS • android.permission.READ_CELL_BROADCASTS
android.permission-group.STORAGE	• android.permission.READ_EXTERNAL_STORAGE • android.permission.WRITE_EXTERNAL_STORAGE

我们所要关注的就是危险权限，由上图可以看到这些权限被分为不同的权限组(PermissionGroup)，这里需要说明一下，当一个权限组里的任一权限被授权，这个组里的其他权限也都会被授权，比如：READ_EXTERNAL_STORAGE这个读SD卡的权限被授权了，这时候WRITE_EXTERNAL_STORAGE也同时被授权。

预览

我们要在保证权限适配的同时，保证代码的整洁和可读，最终我们实现的效果是如下

```
requestCameraPermission(new PermissionHandler() {
            @Override
            public void onGranted() {
                Intent intent = new Intent(); //调用照相机
                intent.setAction(MediaStore.ACTION_IMAGE_CAPTURE);
                startActivity(intent);
            }

            @Override
            public void onDenied() {

            }
        });
```

实践

关于运行时权限的理论知识就不多说了，网上一搜也是一大把，我们这里着重讲如何实践。当你准备做6.0权限适配的时候，你的第一反应会是:"卧槽，项目中要修改的地方太多了，心中无数个草泥马。"这个时候你要淡定，其实一切没有那么复杂

1.打开应用程序设置-权限,比如微信,这里看到的权限就是你将要进行适配的权限，也不会太多


应用访问授权
微信
位置信息
存储空间
电话
相机
短信
身体传感器
通讯录
麦克风

2. 分析哪些权限是基础权限

所谓基础权限就是你的App普遍都需要用的，比如位置、存储权限，如果要在项目中适配这两个权限的话，代码肯定会被改得面目全非，所以我们把这两个权限的获取放在启动页去判断。如果基础权限没有授权通过，我们就不让进入App，基础权限都不给还用个毛，这么一来适配的工作就简单多了。

3. 上一个使用原生API获取权限的小栗子

```
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.btn_camera:
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                if (checkSelfPermission(Manifest.permission.CAMERA) == PackageManager.PERMISSION_DENIED) {
                    requestPermissions(new String[]{Manifest.permission.CAMERA}, PERMISSION_REQUEST_CODE_CAMERA);
                } else {
                    Intent intent = new Intent(); //调用照相机
                    intent.setAction(MediaStore.ACTION_IMAGE_CAPTURE);
                    startActivity(intent);
                }
            }
            break;
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);

    if (requestCode == PERMISSION_REQUEST_CODE_CAMERA) {
        int grantResult = grantResults[0];
        if (grantResult == PackageManager.PERMISSION_GRANTED) {

            Intent intent = new Intent(); //调用照相机
            intent.setAction(MediaStore.ACTION_IMAGE_CAPTURE);
            startActivity(intent);
        }
    }
}
```

获取权限操作

获取权限结果

这是最原生的使用方法，可以看到权限的获取操作和获取结果不是在同一个地方的，这样的话对原有代码的改动还是比较大的，而且操作过程繁琐，标题不是说"比你想的还要简单"吗？

4. 这个时候PermissionsDispatcher就要登场了github

> PermissionsDispatcher是一个通过注解在编译期间生成权限检查代码的工具，以最少的改动来让你的App对权限进行适配。

主要有下面5个注解

> @RuntimePermissions 标记需要运行时判断的类
> @NeedsPermission 标记需要检查权限的方法
> @OnShowRationale 授权提示回调
> @OnPermissionDenied 授权被拒绝回调
> @OnNeverAskAgain 授权不再拒绝不再显示回调

配置

跟ButterKnife和Dagger2一样，配置方法很简单
在项目的build.gradle文件中加上:

```
buildscript {
  dependencies {
    classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
  }
}
```

在module中的build.gradle加上:

```
apply plugin: 'android-apt'

dependencies {
  compile 'com.github.hotchemi:permissionsdispatcher:2.1.3'
  apt 'com.github.hotchemi:permissionsdispatcher-processor:2.1.3'
}
```

使用方法

```
@RuntimePermissions
public class PermissionsDispatcherActivity extends
AppCompatActivity    implements View.OnClickListener {

        @Override
        protected void onCreate(@Nullable Bundle
savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_normal);

            setTitle("PermissionsDispatcher");

            findViewById(R.id.btn_camera).setOnClickListener(this);
            findViewById(R.id.btn_call).setOnClickListener(this);
        }

        @Override
        public void onClick(View v) {
            switch (v.getId()) {
                case R.id.btn_call:

PermissionsDispatcherActivityPermissionsDispatcher.startCallWithChe
ck(this);
                    break;
```

```
                case R.id.btn_camera:

PermissionsDispatcherActivityPermissionsDispatcher.startCameraWithCheck(this);
                    break;
            }
        }

        @NeedsPermission(Manifest.permission.CAMERA)
        void startCamera() {
            Intent intent = new Intent(); //调用照相机
            intent.setAction(MediaStore.ACTION_IMAGE_CAPTURE);
            startActivity(intent);
        }

        @NeedsPermission(Manifest.permission.CALL_PHONE)
        void startCall(){
            Intent intent = new Intent(Intent.ACTION_CALL);
            Uri data = Uri.parse("tel:10086");
            intent.setData(data);
            startActivity(intent);
        }
    }
```

1.首先@RuntimePermissions注解需要进行权限判断的类
2.将需要权限的操作定义在一个方法里，并用@NeedsPermission(Manifest.permission.CAMERA)表明需要的权限(可以是多个)
3.Make编译一下，就会生成【当前类名+PermissionsDispatcher】的类，在原本调用的地方调用@NeedsPermission标记的方法，这时候你会发现会对应生成【方法名+WithCheck】的方法
4.如果你需要监听拒绝后的操作，则使用@OnPermissionDenied，使用方法一样。

原理

PermissionsDispatcher在编译期间，对需要权限判断的方法前后进行修饰，增加权限检查、获取逻辑，我们打开生成的代码看看便知道了,这里边并没有什么高深的东西。

```
final class PermissionsDispatcherActivityPermissionsDispatcher {
  private static final int REQUEST_STARTCAMERA = 2;

  private static final String[] PERMISSION_STARTCAMERA = new String[] {"android.permission.CAMERA"};

  static void startCameraWithCheck(PermissionsDispatcherActivity target) {
    if (PermissionUtils.hasSelfPermissions(target, PERMISSION_STARTCAMERA)) {
      target.startCamera();
    } else {
      ActivityCompat.requestPermissions(target, PERMISSION_STARTCAMERA, REQUEST_STARTCAMERA);
    }
  }

  static void onRequestPermissionsResult(PermissionsDispatcherActivity target, int requestCode, int[] grantResults) {
    switch (requestCode) {
      case REQUEST_STARTCAMERA:
      if (PermissionUtils.getTargetSdkVersion(target) < 23 && !PermissionUtils.hasSelfPermissions(target, PERMISSION_STARTCAMERA)) {
        return;
      }
      if (PermissionUtils.verifyPermissions(grantResults)) {
        target.startCamera();
      }
      break;
      default:
      break;
    }
  }
}
```

封装

从上面的使用方法来看，增加一个权限判断需要定义一个方法，如果需要监听拒绝，则还要定义对应的方法，当需要获取不同权限的时候代码就多了。这时，我们就可以把权限代码抽取到Activity父类中，这里叫BasePermissionActivity，代码如下.

```
/**
 * 权限管理
 * Created by Laiyimin on 2016/8/16.
 */
```

```
@RuntimePermissions
public abstract class BasePermissionActivity extends
AppCompatActivity {

    /**
     * 权限回调接口
     */
    public abstract class PermissionHandler {
        /**
         * 权限通过
         */
        public abstract void onGranted();

        /**
         * 权限拒绝
         */
        public void onDenied() {
        }
    }

    private PermissionHandler mHandler;


    @Override
    public void onRequestPermissionsResult(int requestCode,
String[] permissions, int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);

BasePermissionActivityPermissionsDispatcher.onRequestPermissionsRes
ult(this, requestCode, grantResults);
    }
```

```
    //----------------------------------------------------------
    /**
     * 请求相机权限
     *
     * @param permissionHandler
     */
    protected void requestCameraPermission(PermissionHandler
permissionHandler) {
        this.mHandler = permissionHandler;

BasePermissionActivityPermissionsDispatcher.handleCameraPermissionW
ithCheck(this);
    }


    @NeedsPermission(Manifest.permission.CAMERA)
    void handleCameraPermission() {
        if (mHandler != null)
            mHandler.onGranted();
    }

    @OnPermissionDenied(Manifest.permission.CAMERA)
    void deniedCameraPermission() {
        if (mHandler != null)
            mHandler.onDenied();
    }

    @OnNeverAskAgain(Manifest.permission.CAMERA)
    void OnCameraNeverAskAgain() {
        showDialog("[相机]");
    }

    //----------------------------------------------------------
```

```
    /**
     * 请求电话权限
     *
     * @param permissionHandler
     */
    protected void requestCallPermission(PermissionHandler
permissionHandler) {
        this.mHandler = permissionHandler;

BasePermissionActivityPermissionsDispatcher.handleCallPermissionWit
hCheck(this);
    }


    @NeedsPermission(Manifest.permission.CALL_PHONE)
    void handleCallPermission() {
        if (mHandler != null)
            mHandler.onGranted();
    }

    @OnPermissionDenied(Manifest.permission.CALL_PHONE)
    void deniedCallPermission() {
        if (mHandler != null)
            mHandler.onDenied();
    }

    @OnNeverAskAgain(Manifest.permission.CALL_PHONE)
    void OnCallNeverAskAgain() {
        showDialog("[电话]");
    }


    public void showDialog(String permission) {
```

```
        new AlertDialog.Builder(this)
                .setTitle("权限申请")
                .setMessage("在设置-应用-荟医医生-权限中开启" +
permission + "权限，以正常使用荟医功能")
                .setPositiveButton("去开启", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int
which) {
                        Intent intent = new
Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
                        Uri uri = Uri.fromParts("package",
getPackageName(), null);
                        intent.setData(uri);
                        startActivity(intent);

                        dialog.dismiss();
                    }
                })
                .setNegativeButton("取消", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int
which) {
                        if (mHandler != null) mHandler.onDenied();
                        dialog.dismiss();
                    }
                })
                .setCancelable(false)
                .show();
    }

}
```

这里定义了一个PermissionHandler回调接口，同时mHandler保存了当前权限的回调操作(因为同一时间只能有一次权限请求)，

```
protected void requestCameraPermission(PermissionHandler
permissionHandler) {
    this.mHandler = permissionHandler;

BasePermissionActivityPermissionsDispatcher.handleCameraPermissionW
ithCheck(this);
}



@NeedsPermission(Manifest.permission.CAMERA)
void handleCameraPermission() {
    if (mHandler != null)
        mHandler.onGranted();
}
```

这里定义好各种权限请求的方法供子类调用，例如requestCameraPermission，在它里边调用了权限判断方法，逻辑很简单，大家看看代码就明白了。最后在子类中只要调用这个方法就行了。

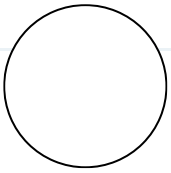
```
requestCameraPermission(new PermissionHandler() {
        @Override
        public void onGranted() {
            Intent intent = new Intent(); //调用照相机
            intent.setAction(MediaStore.ACTION_IMAGE_CAPTURE);
            startActivity(intent);
        }

        @Override
        public void onDenied() {
```

```
        }
    });
```

后续

在App运行过程中，用户可能手动去关闭权限，如果这个时候正在使用着权限，应用就会挂掉，我们的处理就是让App自动重启，让其在引导页重新获取权限，保证软件后续正常运行。附带一个CrashHandler类实现了异常重启，在项目代码中可以找到。

GitHub地址 https://github.com/a5533348/XPermission



EM.Lai

Read more posts by this author.

Share this post

9条评论 最新 最早 最热

谭冉冉✓

赞一个。比easypermissions好用多了。

8月19日 回复 顶 转发

成诺

什么叫基本权限，位置能叫基本权限？

8月19日 回复 顶 转发

成诺

存储权限为什么要用到，默认的存储空间给你干嘛的？非得存内存卡里吗？我手机没内存卡是不是功能都不让用？

8月19日 回复 顶 转发

CE3

位置权限也是基础权限 are you sure?

8月20日 回复 顶 转发

赖亦敏

回复 CE3: 这里的基础权限是相对App来说的，比如打车软件

8月21日 回复 顶 转发

赖亦敏

回复 成诺：确实是不给用，你试试微信，支付宝这些主流App第一次启动的时候，你拒绝任何一个权限你都进不去

8月21日 回复 顶 转发

...

回复 谭冉冉✓：你是不是瞎，哪里看好用了

8月22日 回复 顶 转发

吕中宜

回复 成诺：.....存储权限不是针对内存卡的

8月22日 回复 顶 转发

32

不实用,你在项目中敢这么玩试试.
标题更是标题--并没有比想象的简单,只是绕过了问题而没有解决问题.意味着已经默认了app会崩溃很多次.

24小时前 回复 顶 转发

社交帐号登录： 微信 微博 QQ 人人 更多»

说点什么吧...

发布

再见理想正在使用多说

YOU MIGHT ENJOY

Fragment中startActivityForResult不回调

onActivityResult问题

问题 一次开发中，突然发现Fragment中的onActivityResult方法不回调了？使用的fragment是v4版本的，调用的是fragment的startActivityForResult()。 排查 首先检查fragment所在Activity，发现Activity中存在onActivityResult，fragment发起startActivityforResult后这里是会被调用的，但是fragment中的不调用。这时如果把Activity中的onActivityResult去掉，fragment中又可以回调了，那么问题很明确是出在Activity中了. 解决 原因是Activity的onActivityResult，如果想把result继续传到子Fragment中，必须调用 super.onActivityResult(…