[Android 6.0 动态权限申请](http://www.cnblogs.com/ldq2016/p/7090872.html)

http://www.cnblogs.com/ldq2016/p/7090872.html

**1. 概述**

[**Android**](http://lib.csdn.net/base/15) 6.0 (API 23) 之前应用的权限在安装时全部授予，运行时应用不再需要询问用户。在 Android 6.0 或更高版本对权限进行了分类，对某些涉及到用户隐私的权限可在运行时根据用户的需要动态授予。这样就不需要在安装时被强迫同意某些权限。

**2. 正常权限 和 危险权限**

Android系统权限分为几个保护级别。需要了解的两个最重要保护级别是 *正常权限* 和 *危险权限*:

（1）正常权限:

涵盖应用需要访问其沙盒外部数据或资源，但对用户隐私或其他应用操作风险很小的区域。这些权限在应用安装时授予，运行时不再询问用户。例如: 网络访问、WIFI状态、音量设置等。完整的正常权限列表参考官网 [正常权限](https://developer.android.google.cn/guide/topics/security/normal-permissions.html)。

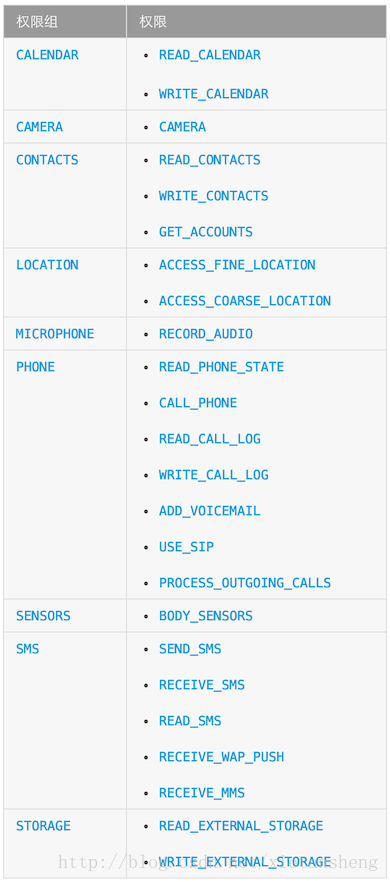
（2）危险权限:

涵盖应用需要涉及用户隐私信息的数据或资源，或者可能对用户存储的数据或其他应用的操作产生影响的区域。例如: 读取通讯录、读写存储器数据、获取用户位置等。如果应用声明需要这些危险权限，则必须在运行时明确告诉用户，让用户手动授予。

**3. 权限组**

Android系统对所有的危险权限进行了分组，称为 **权限组** 。属于同一组的危险权限将自动合并授予，用户授予应用某个权限组的权限，则应用将获得该权限组下的所有权限（前提是相关权限在 AndroidManifest.xml 中有声明）。

**危险权限** 和 **权限组** 列表如下:



PS: 在 AndroidManifest.xml 声明过的危险权限对应的权限组可以在系统 “设置” -> “应用” -> “应用信息” -> “权限” 中查看，可以手动授权和取消授权。

权限组和权限在Android代码中以 **字符串常量** 来表示，分别定义在以下两个 **静态内部类** 的字段中:

* android.Manifest.**permission\_group**（权限组）:
  + Manifest.permission\_group.CALENDAR
  + Manifest.permission\_group.STORAGE
  + ......
* android.Manifest.**permission**（权限）:
  + Manifest.permission.READ\_CALENDAR
  + Manifest.permission.READ\_EXTERNAL\_STORAGE
  + ......

**4. 在运行时请求权限**

设备系统是 Android 6.0 (API 23) 或更高版本，并且应用的 targetSdkVersion 是 23 或更高版本，则针对在 AndroidManifest.xml 中声明的危险权限，在运行时还需要动态请求用户授权。

动态权限请求相关操作的API封装在在android.support.v4包中，发起请求权限的Activity需要直接或间接继承android.support.v4.app.FragmentActivity。

PS: 也可以在直接或间接继承 android.support.v4.app.Fragment 的 Fragment 中发起权限请求。

代码步骤中主要包含以下几个方法:

（1）**检查权限**

// 检查权限

ContextCompat.checkSelfPermission(Context context, String permission)

返回值（android.content.pm.PackageManager中的常量）:

* 有权限: PackageManager.PERMISSION\_GRANTED
* 无权限: PackageManager.PERMISSION\_DENIED

当应用需要用到危险权限时，在执行权限相关代码前，使用该方法判断是否拥有指定的权限。有权限，则继续执行设计需要权限的代码；无权限，则向用户请求授予权限。

（2）**解释权限**

// 解释权限

ActivityCompat.shouldShowRequestPermissionRationale(Activity activity, String permission)

* 1

判断是否有必要向用户解释为什么要这项权限。如果应用第一次请求过此权限，但是被用户拒绝了，则之后调用该方法将返回 true，此时就有必要向用户详细说明需要此权限的原因（个人认为此方法是可选的）。

PS: 如果应用第一次请求此权限时被用户拒绝，第二次再请求此权限时，用户勾选了权限请求对话框的“不再询问”，则此方法返回 false。如果设备规范禁止应用拥有该权限，此方法也返回 false。

（3）**请求权限**

// 请求权限

ActivityCompat.requestPermissions(Activity activity, String[] permissions, int requestCode)

当检测到应用没有指定的权限时，调用此方法向用户请求权限。调用此方法将弹出权限请求对话框询问用户 “允许” 或 “拒绝” 指定的权限。

**PS**:

* 权限参数传入的是数组，可以调用该方法一次请求多个权限；
* 传入的权限数组参数以单个具体权限为单位，但弹框询问用户授权时，**属于同一权限组的权限将自动合并询问授权一次**；
* 请求的权限必须事先在 AndroidManifest.xml 中有声明，否则调用此方法请求时，将不弹框，而是直接返回“拒绝”的结果；
* 第一次请求权限时，用户点击了“拒绝”，第二次再请求该权限时，对话框将出现“不再询问”复选框，如果用户勾选了“不再询问”并点击了“拒绝”，则之后再请求此权限组时将不弹框，而是直接返回“拒绝”的结果。

（4）**处理结果**

请求权限的结果返回和接收一个Activity的返回类似，重写 FragmentActivity 或 (v4) Fragment 中的 onRequestPermissionsResult(...) 方法。

/\*\*

\* 处理权限请求结果

\*

\* @param requestCode

\* 请求权限时传入的请求码，用于区别是哪一次请求的

\*

\* @param permissions

\* 所请求的所有权限的数组

\*

\* @param grantResults

\* 权限授予结果，和 permissions 数组参数中的权限一一对应，元素值为两种情况，如下:

\* 授予: PackageManager.PERMISSION\_GRANTED

\* 拒绝: PackageManager.PERMISSION\_DENIED

\*/

@Override

public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {

// ...

}

**5. 代码演示**

功能很简单，如下:

1. 点击一个按钮，如果有相应权限就进行备份通讯录操作；
2. 如果没有相应的权限，则向用户申请权限；
3. 如果用户授权通过，则继续进行备份通讯录操作；
4. 如果用户拒绝授权，则弹出对话框引导用户跳转到应用权限管理界面手动授权。

部分文件代码:

1、首先在 AndroidManifest.xml 中声明权限

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.xiets.demoapp">

<!-- 声明所有需要的权限（包括普通权限和危险权限） -->

<uses-permission android:name="android.permission.READ\_CONTACTS"/>

<uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE"/>

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

<application

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme">

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>

2、布局文件 activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:padding="10dp">

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:onClick="click"

android:textSize="20sp"

android:text="备份通讯录" />

</RelativeLayout>

3、MainActivity

package com.xiets.demoapp;

import android.Manifest;

import android.content.DialogInterface;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.net.Uri;

import android.os.Bundle;

import android.provider.Settings;

import android.support.annotation.NonNull;

import android.support.v4.app.ActivityCompat;

import android.support.v4.content.ContextCompat;

import android.support.v7.app.AlertDialog;

import android.support.v7.app.AppCompatActivity;

import android.view.View;

import android.widget.Toast;

/\*\*

\* 一键备份通讯录

\*

\* @author xietansheng

\*/

public class MainActivity extends AppCompatActivity {

private static final int MY\_PERMISSION\_REQUEST\_CODE = 10000;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

/\*\*

\* 点击按钮，将通讯录备份保存到外部存储器备。

\*

\* 需要3个权限(都是危险权限):

\* 1. 读取通讯录权限;

\* 2. 读取外部存储器权限;

\* 3. 写入外部存储器权限.

\*/

public void click(View view) {

/\*\*

\* 第 1 步: 检查是否有相应的权限

\*/

boolean isAllGranted = checkPermissionAllGranted(

new String[] {

Manifest.permission.READ\_CONTACTS,

Manifest.permission.READ\_EXTERNAL\_STORAGE,

Manifest.permission.WRITE\_EXTERNAL\_STORAGE

}

);

// 如果这3个权限全都拥有, 则直接执行备份代码

if (isAllGranted) {

doBackup();

return;

}

/\*\*

\* 第 2 步: 请求权限

\*/

// 一次请求多个权限, 如果其他有权限是已经授予的将会自动忽略掉

ActivityCompat.requestPermissions(

this,

new String[] {

Manifest.permission.READ\_CONTACTS,

Manifest.permission.READ\_EXTERNAL\_STORAGE,

Manifest.permission.WRITE\_EXTERNAL\_STORAGE

},

MY\_PERMISSION\_REQUEST\_CODE

);

}

/\*\*

\* 检查是否拥有指定的所有权限

\*/

private boolean checkPermissionAllGranted(String[] permissions) {

for (String permission : permissions) {

if (ContextCompat.checkSelfPermission(this, permission) != PackageManager.PERMISSION\_GRANTED) {

// 只要有一个权限没有被授予, 则直接返回 false

return false;

}

}

return true;

}

/\*\*

\* 第 3 步: 申请权限结果返回处理

\*/

@Override

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

super.onRequestPermissionsResult(requestCode, permissions, grantResults);

if (requestCode == MY\_PERMISSION\_REQUEST\_CODE) {

boolean isAllGranted = true;

// 判断是否所有的权限都已经授予了

for (int grant : grantResults) {

if (grant != PackageManager.PERMISSION\_GRANTED) {

isAllGranted = false;

break;

}

}

if (isAllGranted) {

// 如果所有的权限都授予了, 则执行备份代码

doBackup();

} else {

// 弹出对话框告诉用户需要权限的原因, 并引导用户去应用权限管理中手动打开权限按钮

openAppDetails();

}

}

}

/\*\*

\* 第 4 步: 备份通讯录操作

\*/

private void doBackup() {

// 本文主旨是讲解如果动态申请权限, 具体备份代码不再展示, 就假装备份一下

Toast.makeText(this, "正在备份通讯录...", Toast.LENGTH\_SHORT).show();

}

/\*\*

\* 打开 APP 的详情设置

\*/

private void openAppDetails() {

AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setMessage("备份通讯录需要访问 “通讯录” 和 “外部存储器”，请到 “应用信息 -> 权限” 中授予！");

builder.setPositiveButton("去手动授权", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

Intent intent = new Intent();

intent.setAction(Settings.ACTION\_APPLICATION\_DETAILS\_SETTINGS);

intent.addCategory(Intent.CATEGORY\_DEFAULT);

intent.setData(Uri.parse("package:" + getPackageName()));

intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

intent.addFlags(Intent.FLAG\_ACTIVITY\_NO\_HISTORY);

intent.addFlags(Intent.FLAG\_ACTIVITY\_EXCLUDE\_FROM\_RECENTS);

startActivity(intent);

}

});

builder.setNegativeButton("取消", null);

builder.show();

}

}