一、概述

之前项目的新特性适配工作都是同事在做，一直没有怎么太关注，不过类似这些适配的工作还是有必要做一些记录的。

对于[**Android**](http://lib.csdn.net/base/android) 7.0，提供了非常多的变化，详细的可以阅读官方文档[Android 7.0 行为变更](https://developer.android.com/about/versions/nougat/android-7.0-changes.html)，记得当时做了多窗口支持、FileProvider以及7.1的3D Touch的支持，不过和我们开发者关联最大的，或者说必须要适配的就是去除项目中传递file://类似格式的uri了。

在官方7.0的以上的系统中，尝试传递 file://URI可能会触发FileUriExposedException。

所以本文主要描述如何适配该问题，没什么难度，仅做记录。

注：本文targetSdkVersion 25 ，compileSdkVersion 25

二、拍照案例

大家应该对于手机拍照一定都不陌生，在希望得到一张高清拍照图的时候，我们通过Intent会传递一个File的Uri给相机应用。

大致代码如下:

private static final int REQUEST\_CODE\_TAKE\_PHOTO = 0x110;

private String mCurrentPhotoPath;

public void takePhotoNoCompress(View view) {

Intent takePictureIntent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);

if (takePictureIntent.resolveActivity(getPackageManager()) != null) {

String filename = new SimpleDateFormat("yyyyMMdd-HHmmss", Locale.CHINA)

.format(new Date()) + ".png";

File file = new File(Environment.getExternalStorageDirectory(), filename);

mCurrentPhotoPath = file.getAbsolutePath();

takePictureIntent.putExtra(MediaStore.EXTRA\_OUTPUT, Uri.fromFile(file));

startActivityForResult(takePictureIntent, REQUEST\_CODE\_TAKE\_PHOTO);

}

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (resultCode == RESULT\_OK && requestCode == REQUEST\_CODE\_TAKE\_PHOTO) {

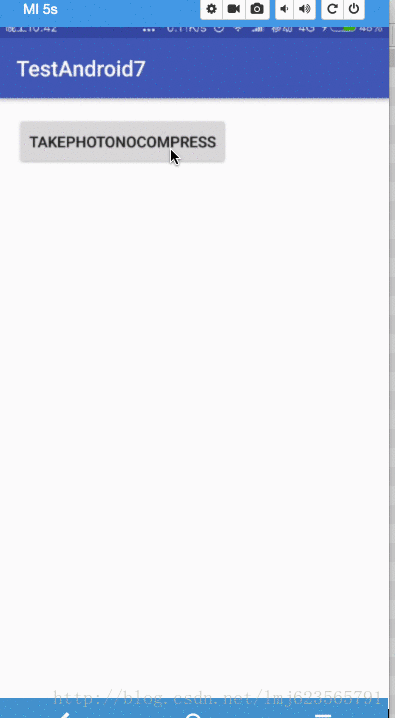
mIvPhoto.setImageBitmap(BitmapFactory.decodeFile(mCurrentPhotoPath));

}

// else tip?

}

贴个效果图吧~



未处理6.0权限，有需要的自行处理下，nexus系列如果未处理，需要手动在设置页开启存储权限。

此时如果我们使用[**android**](http://lib.csdn.net/base/android) 7.0或者以上的原生系统，再次运行一下，你会发现应用直接停止运行，抛出了android.os.FileUriExposedException：

Caused by: android.os.FileUriExposedException:

file:///storage/emulated/0/20170601-030254.png

exposed beyond app through ClipData.Item.getUri()

at android.os.StrictMode.onFileUriExposed(StrictMode.java:1932)

at android.net.Uri.checkFileUriExposed(Uri.java:2348)

所以如果你意识到自己写的代码，在7.0的原生系统的手机上直接就crash是不是很方~

原因在官网已经给了解释：

对于面向 Android 7.0 的应用，Android 框架执行的 StrictMode API 政策禁止在您的应用外部公开 file:// URI。如果一项包含文件 URI 的 intent 离开您的应用，则应用出现故障，并出现 FileUriExposedException 异常。

同样的，官网也给出了解决方案：

要在应用间共享文件，您应发送一项 content:// URI，并授予 URI 临时访问权限。进行此授权的最简单方式是使用 FileProvider 类。如需了解有关权限和共享文件的详细信息，请参阅共享文件。   
<https://developer.android.com/about/versions/nougat/android-7.0-changes.html#accessibility>

那么下面就看看如何通过FileProvider解决此问题吧。

三、使用FileProvider兼容拍照

其实对于如何使用FileProvider，其实在FileProvider的API页面也有详细的步骤，有兴趣的可以看下。

<https://developer.android.com/reference/android/support/v4/content/FileProvider.html>

FileProvider实际上是ContentProvider的一个子类，它的作用也比较明显了，file:///Uri不给用，那么换个Uri为content://来替代。

下面我们看下整体的实现步骤，并考虑为什么需要怎么做？

（1）声明provider

<provider

android:name="android.support.v4.content.FileProvider"

android:authorities="com.zhy.android7.fileprovider"

android:exported="false"

android:grantUriPermissions="true">

<meta-data

android:name="android.support.FILE\_PROVIDER\_PATHS"

android:resource="@xml/file\_paths" />

</provider>

为什么要声明呢？因为FileProvider是ContentProvider子类哇~~

注意一点，他需要设置一个meta-data，里面指向一个xml文件。

（2）编写resource xml file

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

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

<root-path name="root" path="" />

<files-path name="files" path="" />

<cache-path name="cache" path="" />

<external-path name="external" path="" />

<external-files-path name="name" path="path" />

<external-cache-path name="name" path="path" />

</paths>

在paths节点内部支持以下几个子节点，分别为：

* <root-path/> 代表设备的根目录new File("/");
* <files-path/> 代表context.getFilesDir()
* <cache-path/> 代表context.getCacheDir()
* <external-path/> 代表Environment.getExternalStorageDirectory()
* <external-files-path>代表context.getExternalFilesDirs()
* <external-cache-path>代表getExternalCacheDirs()

每个节点都支持两个属性：

* name
* path

path即为代表目录下的子目录，比如：

<external-path

name="external"

path="pics" />

代表的目录即为：Environment.getExternalStorageDirectory()/pics，其他同理。

当这么声明以后，代码可以使用你所声明的当前文件夹以及其子文件夹。

本例使用的是SDCard所以这么写即可：

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

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

<external-path name="external" path="" />

</paths>

为了简单，我们直接使用SDCard根目录，所以path里面就不填写子目录了~

这里你可能会有疑问，为什么要写这么个xml文件，有啥用呀？

刚才我们说了，现在要使用content://uri替代file://uri，那么，content://的uri如何定义呢？总不能使用文件路径吧，那不是骗自己么~

所以，需要一个虚拟的路径对文件路径进行映射，所以需要编写个xml文件，通过path以及xml节点确定可访问的目录，通过name属性来映射真实的文件路径。

（3）使用FileProvider API

好了，接下来就可以通过FileProvider把我们的file转化为content://uri了~

public void takePhotoNoCompress(View view) {

Intent takePictureIntent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);

if (takePictureIntent.resolveActivity(getPackageManager()) != null) {

String filename = new SimpleDateFormat("yyyyMMdd-HHmmss", Locale.CHINA)

.format(new Date()) + ".png";

File file = new File(Environment.getExternalStorageDirectory(), filename);

mCurrentPhotoPath = file.getAbsolutePath();

Uri fileUri = FileProvider.getUriForFile(this, "com.zhy.android7.fileprovider", file);

takePictureIntent.putExtra(MediaStore.EXTRA\_OUTPUT, fileUri);

startActivityForResult(takePictureIntent, REQUEST\_CODE\_TAKE\_PHOTO);

}

}

核心代码就这一行了~

FileProvider.getUriForFile(this, "com.zhy.android7.fileprovider", file);

第二个参数就是我们配置的authorities，这个很正常了，总得映射到确定的ContentProvider吧~所以需要这个参数。

然后再看一眼我们生成的uri：

content://com.zhy.android7.fileprovider/external/20170601-041411.png

可以看到格式为：content://authorities/定义的name属性/文件的相对路径，即name隐藏了可存储的文件夹路径。

现在拿7.0的原生手机运行就正常啦~

不过事情到此并没有结束~~

打开一个4.4的模拟器，运行上述代码，你会发现又Crash啦,抛出了：Permission Denial~

Caused by: java.lang.SecurityException: Permission Denial: opening provider android.support.v4.content.FileProvider from ProcessRecord{52b029b8 1670:com.android.camera/u0a36} (pid=1670, uid=10036) that is not exported from uid 10052

at android.os.Parcel.readException(Parcel.java:1465)

at android.os.Parcel.readException(Parcel.java:1419)

at android.app.ActivityManagerProxy.getContentProvider(ActivityManagerNative.java:2848)

at android.app.ActivityThread.acquireProvider(ActivityThread.java:4399)

因为低版本的系统，仅仅是把这个当成一个普通的Provider在使用，而我们没有授权，contentprovider的export设置的也是false；导致Permission Denial。

那么，我们是否可以将export设置为true呢？

很遗憾是不能的。

在FileProvider的内部：

@Override

public void attachInfo(Context context, ProviderInfo info) {

super.attachInfo(context, info);

// Sanity check our security

if (info.exported) {

throw new SecurityException("Provider must not be exported");

}

if (!info.grantUriPermissions) {

throw new SecurityException("Provider must grant uri permissions");

}

mStrategy = getPathStrategy(context, info.authority);

}

确定了exported必须是false，grantUriPermissions必须是true ~~

所以唯一的办法就是授权了~

context提供了两个方法：

* grantUriPermission(String toPackage, Uri uri,   
  int modeFlags)
* revokeUriPermission(Uri uri, int modeFlags);

可以看到grantUriPermission需要传递一个包名，就是你给哪个应用授权，但是很多时候，比如分享，我们并不知道最终用户会选择哪个app，所以我们可以这样：

List<ResolveInfo> resInfoList = context.getPackageManager()

.queryIntentActivities(intent, PackageManager.MATCH\_DEFAULT\_ONLY);

for (ResolveInfo resolveInfo : resInfoList) {

String packageName = resolveInfo.activityInfo.packageName;

context.grantUriPermission(packageName, uri, flag);

}

根据Intent查询出的所以符合的应用，都给他们授权~~

恩，你可以在不需要的时候通过revokeUriPermission移除权限~

那么增加了授权后的代码是这样的：

public void takePhotoNoCompress(View view) {

Intent takePictureIntent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);

if (takePictureIntent.resolveActivity(getPackageManager()) != null) {

String filename = new SimpleDateFormat("yyyyMMdd-HHmmss", Locale.CHINA)

.format(new Date()) + ".png";

File file = new File(Environment.getExternalStorageDirectory(), filename);

mCurrentPhotoPath = file.getAbsolutePath();

Uri fileUri = FileProvider.getUriForFile(this, "com.zhy.android7.fileprovider", file);

List<ResolveInfo> resInfoList = getPackageManager()

.queryIntentActivities(takePictureIntent, PackageManager.MATCH\_DEFAULT\_ONLY);

for (ResolveInfo resolveInfo : resInfoList) {

String packageName = resolveInfo.activityInfo.packageName;

grantUriPermission(packageName, fileUri, Intent.FLAG\_GRANT\_READ\_URI\_PERMISSION

| Intent.FLAG\_GRANT\_WRITE\_URI\_PERMISSION);

}

takePictureIntent.putExtra(MediaStore.EXTRA\_OUTPUT, fileUri);

startActivityForResult(takePictureIntent, REQUEST\_CODE\_TAKE\_PHOTO);

}

}

这样就搞定了，不过还是挺麻烦的，如果你仅仅是对旧系统做兼容，还是建议做一下版本校验即可，也就是说不要管什么授权了，直接这样获取uri

Uri fileUri = null;

if (Build.VERSION.SDK\_INT >= 24) {

fileUri = FileProvider.getUriForFile(this, "com.zhy.android7.fileprovider", file);

} else {

fileUri = Uri.fromFile(file);

}

这样会比较方便~也避免导致一些问题。当然了，完全使用uri也有一些好处，比如你可以使用私有目录去存储拍摄的照片~

文章最后会给出快速适配的方案~~不需要这么麻烦~

好像，还有什么知识点没有提到，再看一个例子吧~

四、使用FileProvider兼容安装apk

正常我们在编写安装apk的时候，是这样的：

public void installApk(View view) {

File file = new File(Environment.getExternalStorageDirectory(), "testandroid7-debug.apk");

Intent intent = new Intent(Intent.ACTION\_VIEW);

intent.setDataAndType(Uri.fromFile(file),

"application/vnd.android.package-archive");

startActivity(intent);

}

拿个7.0的原生手机跑一下，android.os.FileUriExposedException又来了~~

android.os.FileUriExposedException: file:///storage/emulated/0/testandroid7-debug.apk exposed beyond app through Intent.getData()

好在有经验了，简单修改下uri的获取方式。

if (Build.VERSION.SDK\_INT >= 24) {

fileUri = FileProvider.getUriForFile(this, "com.zhy.android7.fileprovider", file);

} else {

fileUri = Uri.fromFile(file);

}

再跑一次，没想到还是抛出了异常（警告，没有Crash）:

java.lang.SecurityException: Permission Denial:

opening provider android.support.v4.content.FileProvider

from ProcessRecord{18570a 27107:com.google.android.packageinstaller/u0a26} (pid=27107, uid=10026) that is not exported from UID 10004

可以看到是权限问题，对于权限我们刚说了一种方式为grantUriPermission，这种方式当然是没问题的啦~

加上后运行即可。

其实对于权限，还提供了一种方式，即：

intent.addFlags(Intent.FLAG\_GRANT\_READ\_URI\_PERMISSION | Intent.FLAG\_GRANT\_WRITE\_URI\_PERMISSION);

我们可以在安装包之前加上上述代码，再次运行正常啦~

现在我有两个非常疑惑的问题：

* 问题1：为什么刚才拍照的时候，Android 7的设备并没有遇到Permission Denial的问题？

恩，之所以不需要权限，主要是因为Intent的action为ACTION\_IMAGE\_CAPTURE，当我们startActivity后，会辗转调用Instrumentation的execStartActivity方法，在该方法内部，会调用intent.migrateExtraStreamToClipData();方法。

该方法中包含：

if (MediaStore.ACTION\_IMAGE\_CAPTURE.equals(action)

|| MediaStore.ACTION\_IMAGE\_CAPTURE\_SECURE.equals(action)

|| MediaStore.ACTION\_VIDEO\_CAPTURE.equals(action)) {

final Uri output;

try {

output = getParcelableExtra(MediaStore.EXTRA\_OUTPUT);

} catch (ClassCastException e) {

return false;

}

if (output != null) {

setClipData(ClipData.newRawUri("", output));

addFlags(FLAG\_GRANT\_WRITE\_URI\_PERMISSION|FLAG\_GRANT\_READ\_URI\_PERMISSION);

return true;

}

}

可以看到将我们的EXTRA\_OUTPUT，转为了setClipData，并直接给我们添加了WRITE和READ权限。

注：该部分逻辑应该是21之后添加的。

* 问题2：为什么刚才拍照案例的时候，Android 4.4设备遇到权限问题，不通过addFlags这种方式解决？

因为addFlags主要用于setData，setDataAndType以及setClipData（注意：4.4时，并没有将ACTION\_IMAGE\_CAPTURE转为setClipData实现）这种方式。

所以addFlags方式对于ACTION\_IMAGE\_CAPTURE在5.0以下是无效的，所以需要使用grantUriPermission，如果是正常的通过setData分享的uri，使用addFlags是没有问题的（可以写个简单的例子[**测试**](http://lib.csdn.net/base/softwaretest)下，两个app交互，通过content://）。

五、总结下

终于将知识点都涵盖到了~

总结下，使用content://替代file://，主要需要FileProvider的支持，而因为FileProvider是ContentProvider的子类，所以需要在AndroidManifest.xml中注册；而又因为需要对真实的filepath进行映射，所以需要编写一个xml文档，用于描述可使用的文件夹目录，以及通过name去映射该文件夹目录。

对于权限，有两种方式：

* 方式一为Intent.addFlags，该方式主要用于针对intent.setData，setDataAndType以及setClipData相关方式传递uri的。
* 方式二为grantUriPermission来进行授权

相比来说方式二较为麻烦，因为需要指定目标应用包名，很多时候并不清楚，所以需要通过PackageManager进行查找到所有匹配的应用，全部进行授权。不过更为稳妥~

方式一较为简单，对于intent.setData，setDataAndType正常使用即可，但是对于setClipData，由于5.0前后Intent#migrateExtraStreamToClipData，代码发生变化，需要注意~

好了，看到现在是不是觉得适配7.0挺麻烦的，其实一点都不麻烦，下面给大家总结一种快速适配的方式。

六、快速完成适配

（1）新建一个module

创建一个library的module，在其AndroidManifest.xml中完成FileProvider的注册，代码编写为：

<application>

<provider

android:name="android.support.v4.content.FileProvider"

android:authorities="${applicationId}.android7.fileprovider"

android:exported="false"

android:grantUriPermissions="true">

<meta-data

android:name="android.support.FILE\_PROVIDER\_PATHS"

android:resource="@xml/file\_paths" />

</provider>

</application>

注意一点，android:authorities不要写死，因为该library最终可能会让多个项目引用，而android:authorities是不可以重复的，如果两个app中定义了相同的，则后者无法安装到手机中（authority conflict）。

同样的的编写file\_paths~

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

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

<root-path

name="root"

path="" />

<files-path

name="files"

path="" />

<cache-path

name="cache"

path="" />

<external-path

name="external"

path="" />

<external-files-path

name="external\_file\_path"

path="" />

<external-cache-path

name="external\_cache\_path"

path="" />

</paths>

最后再编写一个辅助类，例如：

public class FileProvider7 {

public static Uri getUriForFile(Context context, File file) {

Uri fileUri = null;

if (Build.VERSION.SDK\_INT >= 24) {

fileUri = getUriForFile24(context, file);

} else {

fileUri = Uri.fromFile(file);

}

return fileUri;

}

public static Uri getUriForFile24(Context context, File file) {

Uri fileUri = android.support.v4.content.FileProvider.getUriForFile(context,

context.getPackageName() + ".android7.fileprovider",

file);

return fileUri;

}

public static void setIntentDataAndType(Context context,

Intent intent,

String type,

File file,

boolean writeAble) {

if (Build.VERSION.SDK\_INT >= 24) {

intent.setDataAndType(getUriForFile(context, file), type);

intent.addFlags(Intent.FLAG\_GRANT\_READ\_URI\_PERMISSION);

if (writeAble) {

intent.addFlags(Intent.FLAG\_GRANT\_WRITE\_URI\_PERMISSION);

}

} else {

intent.setDataAndType(Uri.fromFile(file), type);

}

}

}

可以根据自己的需求添加方法。

好了，这样我们的一个小库就写好了~~

（2）使用

如果哪个项目需要适配7.0，那么只需要这样引用这个库，然后只需要改动一行代码即可完成适配啦，例如：

**拍照**

public void takePhotoNoCompress(View view) {

Intent takePictureIntent = new Intent(MediaStore.ACTION\_IMAGE\_CAPTURE);

if (takePictureIntent.resolveActivity(getPackageManager()) != null) {

String filename = new SimpleDateFormat("yyyyMMdd-HHmmss", Locale.CHINA)

.format(new Date()) + ".png";

File file = new File(Environment.getExternalStorageDirectory(), filename);

mCurrentPhotoPath = file.getAbsolutePath();

Uri fileUri = FileProvider7.getUriForFile(this, file);

takePictureIntent.putExtra(MediaStore.EXTRA\_OUTPUT, fileUri);

startActivityForResult(takePictureIntent, REQUEST\_CODE\_TAKE\_PHOTO);

}

}

只需要改动

Uri fileUri = FileProvider7.getUriForFile(this, file);

即可。

**安装apk**

同样的修改setDataAndType为：

FileProvider7.setIntentDataAndType(this,

intent, "application/vnd.android.package-archive", file, true);

即可。

ok，繁琐的重复性操作终于简化为一行代码啦~

源码地址：

<https://github.com/hongyangAndroid/FitAndroid7>