# **Android ShortCuts注意事项**

## 概述

最近在做有关ShortCuts的相关需求，本来以为是个很简单的事情，中途却碰到了一些坑，于是研究了下ShortCuts的生成和删除流程，在这里总结一下分享给大家。

## 安全问题

你也许会问，ShortCuts还会涉及到安全问题？  
先不急，这里的安全问题是指的特殊情况，比如点击ShortCuts后跳转到一个Activity，Activity里面有一个Webview控件用于显示指定的Url（假设点击一个叫Test的ShortCuts后，跳转到一个叫webActivity的界面，并且要打开www.test.com这个网址），**先想一想 不往后看**，你会怎么写

你也许会这么写代码

首先，生成快捷方式

Intent shortcut = new Intent("com.android.launcher.action.INSTALL\_SHORTCUT");

shortcut.putExtra(Intent.EXTRA\_SHORTCUT\_NAME, "Test");

Intent.ShortcutIconResource iconRes

= Intent.ShortcutIconResource.fromContext(this, R.mipmap.ic\_launcher);

shortcut.putExtra(Intent.EXTRA\_SHORTCUT\_ICON\_RESOURCE, iconRes);

Intent action = new Intent(this, WebActivity.class);

action.putExtra("url", "www.test.com");

shortcut.putExtra(Intent.EXTRA\_SHORTCUT\_INTENT, action);

sendBroadcast(shortcut);

在webActivity里响应请求

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

//控件实例化

doViewsInit();

Intent in = getIntent()；

String url = in.getStringExtra("url");

webView.loadUrl(URL);

}

如果你没有这么写，那么恭喜你，你避过了安全问题，如果你这么写了，那么继续往下面看吧  
因为响应ShortCuts的Activity必须是android:exported="true"，也就是Activity的默认值，不需要显示的配置出来，所以可能很多同学没有注意到。  
exported="true"是个什么概念呢？就是说**任何第三方**的程序都可以访问你这个界面，那么问题就来了.

###### load文件

既然刚才说到任何三方都可以调用，那么另外写一个app并且写如下代码也是可以运行的

Intent i = new Intent();

i.setClassName("xxx.xxx.xxx", "xxx.xxx.xxx.webActivity");

i.putExtra("url", "http://www.baidu.com/");

startActivity(i);

这样，你就可以在自己的app里，隐式调用别人写好的界面，传入自己的参数。  
你也许又要问了，这打开一个百度有什么好安全不安全的，那么我们换成一个其他的路径，比如：**文件路径**

i.putExtra("url",

""file:///data/data/xxx.xxx.xxx/shared\_prefs/a.xml")");

你会发现，webview**加载出**了这个sp文件，这样就能获取到别人的一些信息了。

关于如何获取别人包名和是否使用了webview，手段多种多样，不在这里累述。  
至于除了加载文件还能有什么操作，欢迎各位补充.

###### 规避

主要原因是出在`android:exported="true"上面，因为这个参数是将自己暴露出来，又不能改为false，因为ShortCuts必须得是true。  
有同学会想，如果ShortCuts跳转的不是一个Activity，而是一个service，在service里面在启动Activity可以不呢？ 答案当然是：不可以。 因为跳转对象只有是Activity才会生成ShortCuts。  
因为webActivity没有办法判断是谁启动了自己，所以唯一的办法就是webActivity就不能直接接受url这个参数，而是接受一个类型参数，比如type=1，当拿到这个type=1的情况下，再在webActivity里去app中取对应的url地址，这样就只会认自己app的地址。

## 无法删除

在网上搜索各种资料，你会发现，让你删除shortcut时，传递的intent必须是和创建时一致的。  
这当然没有错，但是也**没有**全对。  
但是当你以如下方式创建shortcut时候

Intent shortcut = new Intent("com.android.launcher.action.INSTALL\_SHORTCUT");

shortcut.putExtra(Intent.EXTRA\_SHORTCUT\_NAME, "123");

Intent.ShortcutIconResource iconRes = Intent.ShortcutIconResource.fromContext(this, R.mipmap.ic\_launcher);

shortcut.putExtra(Intent.EXTRA\_SHORTCUT\_ICON\_RESOURCE, iconRes);

//点击后响应的intent没有action参数

Intent action = new Intent(this, MainActivity.class);

shortcut.putExtra(Intent.EXTRA\_SHORTCUT\_INTENT, action);

sendBroadcast(shortcut);

即使删除时使用同样的intent也没办办法删除

Intent remove = new Intent("com.android.launcher.action.UNINSTALL\_SHORTCUT");

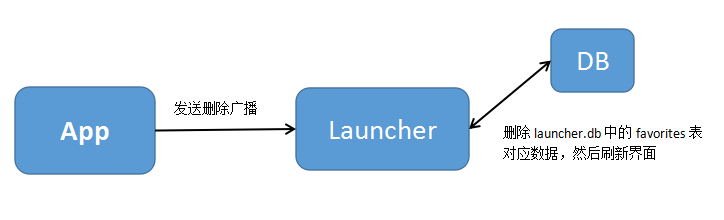
remove.putExtra(Intent.EXTRA\_SHORTCUT\_NAME, "123");

Intent action2 = new Intent(this, MainActivity.class);

remove.putExtra(Intent.EXTRA\_SHORTCUT\_INTENT, action2);

sendBroadcast(remove);

这是为什么呢？！  
还得来看看ShortCuts的删除实现，大致流程如下



shortcuts\_flow.png

我们来看看launcher中，接受到广播后是如何处理的，如下给出主要函数  
packages/apps/Launcher3/src/com/android/launcher3/UninstallShortcutReceiver.java

private static void removeShortcut(Context context, Intent data) {

Intent intent = data.getParcelableExtra(Intent.EXTRA\_SHORTCUT\_INTENT);

String name = data.getStringExtra(Intent.EXTRA\_SHORTCUT\_NAME);

...

if (intent != null && name != null) {

final ContentResolver cr = context.getContentResolver();

Cursor c = cr.query(LauncherSettings.Favorites.CONTENT\_URI,

new String[] { LauncherSettings.Favorites.\_ID, LauncherSettings.Favorites.INTENT },

LauncherSettings.Favorites.TITLE + "=?", new String[] { name }, null);

final int intentIndex = c.getColumnIndexOrThrow(LauncherSettings.Favorites.INTENT);

...

while (c.moveToNext()) {

...

if (intent.filterEquals(Intent.parseUri(c.getString(intentIndex), 0))) {

...

cr.delete(uri, null, null);

...

}

...

}

}

其中这个intent 就是App中传入的EXTRA\_SHORTCUT\_INTENT，name就是ShortCuts的名字，如果都不为空，则在数据库中查找匹配的数据，这里只是对**名字**进行了匹配，匹配不到则无法删除，所以我们刚才的例子，是可以匹配到的.  
所以问题的关键是，如下条件是否可以通过，如果通过则删除ShortCut

if (intent.filterEquals(Intent.parseUri(c.getString(intentIndex), 0))) {...}

#### filterEquals

这个方法的作用是，判断2个intent是否完全一致，包括action, type, package等信息

public boolean filterEquals(Intent other) {

if (other == null) {

return false;

}

if (!Objects.equals(this.mAction, other.mAction)) return false;

if (!Objects.equals(this.mData, other.mData)) return false;

if (!Objects.equals(this.mType, other.mType)) return false;

if (!Objects.equals(this.mPackage, other.mPackage)) return false;

if (!Objects.equals(this.mComponent, other.mComponent)) return false;

if (!Objects.equals(this.mCategories, other.mCategories)) return false;

return true;

}

所以，网上说的，删除与创建的intent需要完全一致 是正确的.  
但是上面的例子，2个intent确实是完全一致的，**为什么**还是会无法删除呢？

#### parseUri

说明parseUri方法，在我们传递过来的intent中，添加了一点料，这个料是什么呢?

以上面的例子来说，在数据库中LauncherSettings.Favorites.INTENT字段下面的值，是这样的

#Intent;component=com.example.hly.demo/.MainActivity;end

然后通过parseUri方法转换成一个Intent

public static Intent parseUri(String uri, int flags) throws URISyntaxException {

...

// new format

Intent intent = new Intent(ACTION\_VIEW);

Intent baseIntent = intent;

...

// action

if (uri.startsWith("action=", i)) {

intent.setAction(value);

}

...

return intent;

}

**重点来了!!!**  
parseUri返回的是一个intent，而这个intent在实例化的时候却带得有一个**ACTION\_VIEW**的action  
这是什么意思？  
就是说，如果你创建shortcut时的intent中是没有带action信息，launcher不会存入action信息，但是在删除的时候取出来进行匹配的时候，系统会自动给你加上ACTION\_VIEW的action，从而导致了**匹配失败！！**  
但是如果，你创建shortcut的intent是带得有action信息的，在匹配的时候，这个action信息会把系统的ACTION\_VIEW这个action**覆盖**，这样就能和删除时的intent进行匹配了

所以刚才的例子如果想正确删除的话，需要加入ACTION\_VIEW的action

Intent remove = new Intent("com.android.launcher.action.UNINSTALL\_SHORTCUT");

remove.putExtra(Intent.EXTRA\_SHORTCUT\_NAME, "123");

Intent action2 = new Intent(this, MainActivity.class);//重点

action2.setAction(Intent .ACTION\_VIEW);.

remove.putExtra(Intent.EXTRA\_SHORTCUT\_INTENT, action2);

sendBroadcast(remove);