引用:https://blog.51cto.com/13616847/2120149

https://www.jianshu.com/p/4a8fc0b78094

**几个问题:**

1.已经有了　Intent, 为什么还有　PendingIntent?

2.PendingIntent 也就几个场景用到过，还有其他场景吗？

3.它的内部实现。

**Intent 和 PendingIntent 的区别**

Intent 是意图的意思。Android 中的　Intent 正是取自这个意思，它是一个消息对象，通过它，Android 系统的四大组件能够方便的通信，并且保证解耦。

Intent 可以说明某种意图，携带一种行为和相应的数据，发送到目标组件。

IntentFilter 与 Intent 配套使用，它声明了一个组件接受某个　Intent。

PendingIntent 是对 Intent 的封装，关键的不同在于：

A组件 创建了一个 PendingIntent 的对象然后传给 B组件，B 在执行这个 PendingIntent 的 send 时候，它里面的 Intent 会被发送出去，而接受到这个 Intent 的 C 组件会认为是 A 发的。即B 以 A 的权限和身份发送了这个 Intent。

比如，我们的 Activity 如果设置了 exported = false，其他应用如果使用 Intent 就访问不到这个 Activity，但是使用 PendingIntent 是可以的。

综上所述，PendingIntent 有两个特点：

将某个动作的触发时机交给其他应用

让那个应用代表自己去执行那个动作(权限都给他了)

**PendingIntent 的使用场景**

已知的使用场景是:

通知，在点击通知时执行调起本应用的操作，当然也可以执行其他操作

闹钟，定时执行某个操作

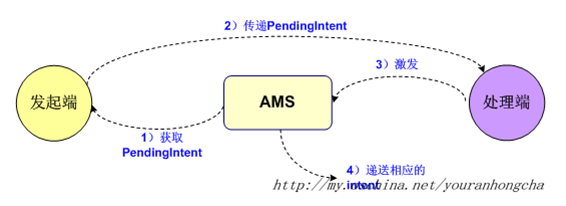
桌面小部件，点击小部件时执行某个操作

通知，闹钟，桌面小部件，都是运行在其他应用中的，但是给我们的感知就像是我们自己的应用的一部分。

**内部实现**

大体的原理是: A应用希望让B应用帮忙触发一个行为，这是跨应用的通信，需要 Android 系统作为中间人，这里的中间人就是 ActivityManager。 A应用创建建 PendingIntent，在创建 PendingIntent 的过程中，向 ActivityManager 注册了这个 PendingIntent，所以，即使A应用死了，当它再次苏醒时，只要提供相同的参数，还是可以获取到之前那个 PendingIntent 的。当 A 将 PendingIntent 调用系统 API 比如 AlarmManager.set()，实际是将权限给了B应用，这时候， B应用可以根据参数信息，来从 ActivityManager 获取到 A 设置的 PendingIntent。

在Android系统中，最适合做集中性管理的组件就是AMS（Activity Manager Service）啦，所以它义不容辞地承担起管理所有PendingIntent的职责。这样我们就可以画出如下示意图：



**PendingIntent 只能通过下列的静态方法获取**

//获取Broadcast关联的

PendingIntent PendingIntent.getBroadcast(Context context, int requestCode, Intent intent, int flags)

//获取Activity关联的

PendingIntent PendingIntent.getActivity(Context context, int requestCode, Intent intent, int flags)

PendingIntent.getActivity(Context context, int requestCode, Intent intent, int flags, Bundle options)

//获取Service关联的

PendingIntent PendingIntent.getService(Context context, int requestCode, Intent intent, int flags)

第四个参数 flags，在 PendingIntent 定义了四个比较常用的 FLAG：

//如果新请求的 PendingIntent 发现已经存在时,取消已存在的,用新的 PendingIntent 替换

int FLAG\_CANCEL\_CURRENT

//如果新请求的PendingIntent发现已经存在时,忽略新请求的,继续使用已存在的.日常开发中很少使用

int FLAG\_NO\_CREATE

//表示PendingIntent只能使用一次,如果已使用过,那么getXXX(...)将会返回NULL

//也就是说同类的通知只能使用一次，后续的通知单击后无法打开。

int FLAG\_ONE\_SHOT

//如果新请求的PendingIntent发现已经存在时,如果Intent有字段改变了,这更新已存在的 //PendingIntent

int FLAG\_UPDATE\_CURRENT

getActivity()的意思其实是，获取一个PendingIntent对象，而且该对象日后激发时所做的事情是启动一个新activity。也就是说，当它异步激发时，会执行类似Context.startActivity()那样的动作。相应地，getBroadcast()和getService()所获取的PendingIntent对象在激发时，会分别执行类似Context..sendBroadcast()和Context.startService()这样的动作。

ContextProvider之所以不能,可能是怕直接修改数据引起系统混乱.不能直接操纵数据,尤其是系统数据

**PendingIntent 的取消**

PendingIntent 是系统对于待处理数据的一个引用，称之为：token；当主程序被 Killed 时，token 还是会继续存在的，可以继续供其他进程使用。如果要取消 PendingIntent，需要调用PendingIntent 的 cancel 方法。

**正确区分不同 PendingIntent 有两种方法：**

1.PendingIntent.getXXX(…) 方法中的 requestCode 不同

2.通过 Intent.filterEquals 测试时不相等