彻底认识 PendingIntent 原创

Jinux111 2018-05-25 10:22:47

文章标签 PendingIntent Notification Intent Android 文章分类 Android 框架 阅读数 1.6万

最近在写一个闹钟程序的时候使用到了 PendingIntent, 而且是两个地方用到，一个是 AlarmManager 定时的时候, 另一个是在点击通知进入应用的时候。其实我早就想深入研究一下 PendingIntent 了，因为我很好奇一下几个问题:

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

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

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。
> 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:

1. 将某个动作的触发时机交给其他应用
2. 让那个应用代表自己去执行那个动作(权限都给他了)

API

获取 PendingIntent

1. getActivity
2. getActivities
3. getBroadcast
4. getService
5. getForegroundService

> 为什么没有 getContentProvider?
> 我猜测，ContentProvider 作为一个数据源，太重要了，相当于是把数据直接暴露出去了

它们的参数都相同，都是四个：Context， requestCode, Intent, flags。Context 不必多说，要想让其他组件代替自己办事，当然要将自己的上下文传给它。action, requestCode 和 Intent 共同来标志一个行为的唯一性，什么意思

呢?

简单的说，我们通过相同的方法(action), 相同的 requestCode 和相同的 Intent 获取到的 PendingIntent, 虽然可能不是同一个对象，但是，却是代表同一个东西，之所以这样看 flags 参数就知道了。

FLAG_ONE_SHOT: 只执行一次， 在调用了 send 以后自动调用 cancel，不能在调用 send 了。
FLAG_NO_CREATE: 不创建新的，如果我们之前设置过，这次就能获取到，否则，返回 null。
FLAG_CANCEL_CURRENT: 如果之前设置过，就取消掉, 重新创建个新的
FLAG_UPDATE_CURRENT: 如果之前设置过，就更新它。更新什么呢，Intent 的 Extras
FLAG_IMMUTABLE: 设置 Intent 在 send 的时候不能更改

发送 PendingIntent

send 是触发 PendingIntent 包含的行为，它有很多重载形式，我们通常的开发用不到他，除非我们做桌面程序开发或者 Android Framework 开发。
这里我们只是大体说明一下，可以传给它一个 Intent 来对它原来的 Intent 做修改，但是如果目标设置了 FLAG_IMMUTABLE 则给参数忽略。可以设置 callback 当发送完成获得回调，并且可以通过设置handler决定回调发生的线程。

取消 PendingIntent

只有设置 PendingIntent 的原来的应用可以取消它，发送方只能发送，当一个 PendingIntent 被取消后，发送则不会成功。

PendingIntent 的使用场景

已知的使用场景是:

1. 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
2. 闹钟，定时执行某个操作
3. 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。