

[Android \(/tags/#Android\)](#) [frameworks \(/tags/#frameworks\)](#) [AlarmManager \(/tags/#AlarmManager\)](#)

Android中AlarmManager使用指南

用AlarmManager设置一个Alarm

Posted by Cheson on February 14, 2017

AlarmManager简介

AlarmManager，顾名思义，就是“提醒”，是Android中常用的一种系统级别的提示服务，在特定的时刻为我们广播一个指定的Intent。简单的说就是我们设定一个时间，然后在该时间到来时，AlarmManager为我们广播一个我们设定的Intent,通常我们使用 PendingIntent，PendingIntent可以理解为Intent的封装包，简单的说就是在Intent上在加个指定的动作。在使用Intent的时候，我们还需要在执行startActivity、startService或sendBroadcast才能使Intent有用。而PendingIntent的话就是将这个动作包含在内了。定义一个PendingIntent对象：

```
PendingIntent pi = PendingIntent.getBroadcast(this,0,intent,0);
```

AlarmManager使用

AlarmManager的常用方法有三个：

1. set接口

```
set(int type, long startTime, PendingIntent pi);
```

该方法用于设置一次性闹钟，第一个参数表示闹钟类型，第二个参数表示闹钟执行时间，第三个参数表示闹钟响应动作。

1. setRepeating接口

```
setRepeating(int type, long startTime, long intervalTime, PendingIntent pi);
```

该方法用于设置重复闹钟，第一个参数表示闹钟类型，第二个参数表示闹钟首次执行时间，第三个参数表示闹钟两次执行的间隔时间，第三个参数表示闹钟响应动作。

1. setInexactRepeating接口

```
setInexactRepeating(int type, long startTime, long intervalTime, PendingIntent pi);
```

该方法也用于设置重复闹钟，与第二个方法相似，不过其两个闹钟执行的间隔时间不是固定的而已。

参数详解

三个方法各个参数详悉：

1. int type：闹钟的类型，常用的有5个值：AlarmManager.ELAPSED_REALTIME、AlarmManager.ELAPSED_REALTIME_WAKEUP、AlarmManager.RTC、AlarmManager.RTC_WAKEUP、AlarmManager.POWER_OFF_WAKEUP。AlarmManager.ELAPSED_REALTIME表示闹钟在手机睡眠状态下不可用，该状态下闹钟使用相对时间（相对于系统启动开始），状态值为3；AlarmManager.ELAPSED_REALTIME_WAKEUP表示闹钟在睡眠状态下会唤醒系统并执行提示功能，该状态下闹钟也使用相对时间，状态值为2；AlarmManager.RTC表示闹钟在睡眠状态下不可用，该状态下闹钟使用绝对时间，即当前系统时间，状态值为1；

AlarmManager.RTC_WAKEUP表示闹钟在睡眠状态下会唤醒系统并执行提示功能，该状态下闹钟使用绝对时间，状态值为0；

AlarmManager.POWER_OFF_WAKEUP表示闹钟在手机关机状态下也能正常进行提示功能，所以是5个状态中用的最多的状态之一，该状态下闹钟也是用绝对时间，状态值为4；不过本状态受SDK版本影响，某些版本并不支持；

2. long startTime：闹钟的第一次执行时间，以毫秒为单位，可以自定义时间，不过一般使用当前时间。需要注意的是，本属性与第一个属性（type）密切相关，如果第一个参数对应的闹钟使用的是相对时间（ELAPSED_REALTIME和ELAPSED_REALTIME_WAKEUP），那么本属性就得使用相对时间（相对于系统启动时间来说），比如当前时间就表示为：SystemClock.elapsedRealtime()；如果第一个参数对应的闹钟使用的是绝对时间（RTC、RTC_WAKEUP、POWER_OFF_WAKEUP），那么本属性就得使用绝对时间，比如当前时间就表示为：System.currentTimeMillis()。
3. long intervalTime：对于后两个方法来说，存在本属性，表示两次闹钟执行的间隔时间，也是以毫秒为单位。
4. PendingIntent pi：绑定了闹钟的执行动作，比如发送一个广播、给出提示等等。PendingIntent是Intent的封装类。需要注意的是，如果是通过启动服务来实现闹钟提示的话，PendingIntent对象的获取就应该采用PendingIntent.getService(Context c,int i,Intent intent,int j)方法；如果是通过广播来实现闹钟提示的话，PendingIntent对象的获取就应该采用PendingIntent.getBroadcast(Context c,int i,Intent intent,int j)方法；如果是采用Activity的方式来实现闹钟提示的话，PendingIntent对象的获取就应该采用PendingIntent.getActivity(Context c,int i,Intent intent,int j)方法。如果这三种方法错用了的话，虽然不会报错，但是看不到闹钟提示效果。

举例说明

定义一个闹钟，5秒钟重复响应

1. MainActivity，在onCreate中完成：

```
//创建Intent对象，action为ELITOR_CLOCK，附加信息为字符串“你该打酱油了”
Intent intent = new Intent("ELITOR_CLOCK");
intent.putExtra("msg", "你该打酱油了");

//定义一个PendingIntent对象，PendingIntent.getBroadcast包含了sendBroadcast的动作。
//也就是发送了action 为"ELITOR_CLOCK"的intent
PendingIntent pi = PendingIntent.getBroadcast(this, 0, intent, 0);

//AlarmManager对象，注意这里并不是new一个对象，Alarmmanager为系统级服务
AlarmManager am = (AlarmManager) getSystemService(ALARM_SERVICE);

//设置闹钟从当前时间开始，每隔5s执行一次PendingIntent对象pi，注意第一个参数与第二个参数的关系
// 5秒后通过PendingIntent pi对象发送广播
am.setRepeating(AlarmManager.RTC_WAKEUP, System.currentTimeMillis(), 5*1000, pi);
```

那么启动MainActivity之后，由于定义了AlarmManager am，并且调用了am.setRepeating(...)函数，则系统每隔5s将会通过pi启动intent发送广播，其action为ELITOR_CLOCK。所以我们需要在Manifest.xml中注册一个receiver，同时自己定义一个广播接收器类。

1. 定义一个广播接收器类MyReceiver，重写onReceive()函数

```
public class MyReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        // TODO Auto-generated method stub
        Log.d("MyTag", "onclock.....");
        String msg = intent.getStringExtra("msg");
        Toast.makeText(context, msg, Toast.LENGTH_SHORT).show();
    }
}
```

1. 在Manifest.xml中注册广播接收器

```
<receiver android:name=".MyReceiver">
    <intent-filter>
        <action android:name="ELITOR_CLOCK" />
    </intent-filter>
</receiver>
```

PREVIOUS

第一次搭建个人主页 (/2017/02/13/BLOG-INIT/)

NEXT

ALARMMANAGERSERVICE之设置ALARM流程 (/2017/02/14/SETALARMFLOW/)

FEATURED TAGS (/tags/)

- 前端 (/tags/#前端)

Android (/tags/#Android)

frameworks (/tags/#frameworks)

AlarmManager (/tags/#AlarmManager)

Performance (/tags/#Performance)

systrace (/tags/#systrace)

PowerManager (/tags/#PowerManager)

Wakelock (/tags/#Wakelock)

Guitar (/tags/#Guitar)

民谣 (/tags/#民谣)


赵雷 (/tags/#赵雷)

Doze (/tags/#Doze)


Android Performance Patterns (/tags/#Android Performance Patterns)


FRIENDS


待遇见志同道合的你 (<https://github.com>) 小明 (<http://www.betterming.cn>)


 (<https://twitter.com/chendongqi>)

 (<https://www.zhihu.com/people/chendongqi>)

 (<http://weibo.com/chendongqi>)

 (<https://www.facebook.com/chendongqi>)

 (<https://github.com/chendongqi>)

 (<https://www.linkedin.com/in/firstname-lastname-idxxxx>)