通知

参考：http://blog.csdn.net/vipzjyno1/article/details/25248021

<https://developer.android.com/guide/topics/ui/notifiers/notifications.html?hl=zh-cn#Progress>

通知是应用常规UI外部向用户展示的消息。它可以在活动，广播接收器，服务中创建。

**使用**

* **主要涉及到两个类：**

**Notification：**通知信息类，它里面对应通知的各个信息。 由NotificationCompat.Builder.build()创建，它将返回包含您的具体规范的 Notification 对象。

**NotificationManager** ： 是状态栏通知的管理类，负责发通知、清除通知等操作。发送消息需要调用 NotificationManager.notify() 将 Notification 对象传递给系统。

* **简单的代码使用步骤：**



上面步骤已经给出了mBuilder的部分方法解释。下面对部分方法进行详细解释：

1. 设置提醒标志符flags向通知添加声音、闪灯和振动效果等设置达到通知提醒效果，可以组合多个属性：

Notification.FLAG\_SHOW\_LIGHTS              //三色灯提醒，在使用三色灯提醒时候必须加该标志符

Notification.FLAG\_ONGOING\_EVENT          //发起正在运行事件（活动中）

Notification.FLAG\_INSISTENT   //让声音、振动无限循环，直到用户响应 （取消或者打开）

Notification.FLAG\_ONLY\_ALERT\_ONCE  //发起Notification后，铃声和震动均只执行一次

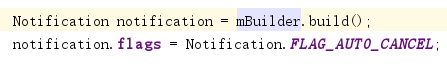
Notification.FLAG\_AUTO\_CANCEL      //用户单击通知后自动消失

Notification.FLAG\_NO\_CLEAR          //只有全部清除时，Notification才会清除 ，不清楚该通知(QQ的通知无法清除，就是用的这个)

Notification.FLAG\_FOREGROUND\_SERVICE    //表示正在运行的服务

设置方式：

1. 实例化通知栏之后通过给他添加.flags属性赋值。



2、通过setContentIntent([PendingIntent](http://blog.csdn.net/vipzjyno1/article/details/25248021" \t "http://blog.csdn.net/vipzjyno1/article/details/_blank) intent)方法中的意图设置对应的flags

1518341314(1)

1. setDefaults(int defaults):向通知添加声音、闪灯和振动效果,可以组合多个属性（和方法1中提示效果一样的）

对应属性：

Notification.DEFAULT\_VIBRATE    //添加默认震动提醒  需要 VIBRATE permission

Notification.DEFAULT\_SOUND    // 添加默认声音提醒

Notification.DEFAULT\_LIGHTS// 添加默认三色灯提醒

Notification.DEFAULT\_ALL// 添加默认以上3种全部提醒

1. setVibrate(long[] patter):设置震动方式。

1518406688(1)

1. setLights(int ledARGB ,int ledOnMS ,int ledOffMS ):android支持三色灯提醒，这个方法就是设置不同场景下的不同颜色的灯。其中ledARGB 表示灯光颜色、 ledOnMS 亮持续时间、ledOffMS 暗的时间。

注意：1）只有在设置了标志符Flags为Notification.FLAG\_SHOW\_LIGHTS的时候，才支持三色灯提醒。

        2）这边的颜色跟设备有关，不是所有的颜色都可以，要看具体设备。

1518406918(1)

1. setSound(Uri sound):设置默认或则自定义的铃声，来提醒。



1. setPriority(int pri)设置优先级

Notification.PRIORITY\_MAX： 重要而紧急的通知，通知用户这个事件是时间上紧迫的或者要立即处理的。

Notification.PRIORITY\_HIGH：高优先级用于重要的通信内容，例如短消息或者聊天，这些都是对用户来

说比较有兴趣的。

Notification.PRIORITY\_DEFAULT ： 默认优先级用于没有特殊优先级分类的通知。

Notification.PRIORITY\_LOW： 低优先级可以通知用户但又不是很紧急的事件。

Notification.PRIORITY\_MIN： 用于后台消息 (例如天气或者位置信息)。最低优先级通知将只在状态栏显

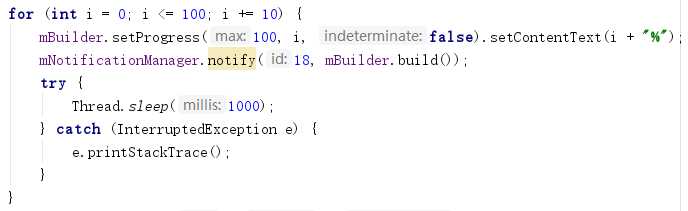
示图标，只有用户下拉通知抽屉才能看到内容。

1. setOngoing(boolean ongoing)：设置为ture，表示它为一个正在进行的通知。他们通常是用来表示一个后台任务,用户积极参与(**如播放音乐**)或以某种方式正在等待,因此占用设备(如一个文件下载,同步操作,主动网络连接)
2. setDeleteIntent(PendingIntent intent):作用于非常驻通知，当用户点击全部清除按钮时或者滑动取消通知时，响应该清除事件的Intent；
3. setProgress(int max, int progress, boolean indeterminate)： Android 4.0 及更高版本的平台上使用，对于早期版本，必须创建包括 [ProgressBar](https://developer.android.com/reference/android/widget/ProgressBar.html?hl=zh-cn) 视图的自定义通知布局。

max:进度条最大数值。   
progress:当前进度。

indeterminate:表示进度是否不确定。如果想要看到任务执行的百分进度则应设置为false，当任务执行完毕后调用setProgress(0, 0, **false**)移除指示条。

a、显示具体进度：



b、显示持续时间固定的进度指示器：

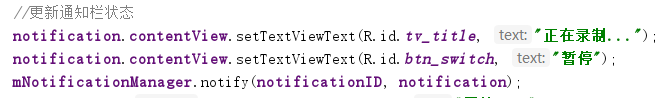
1519288513(1)

当执行完毕任务后想要一出进度条请执行：

1519288750(1)

* **更新通知**

对要将通知设置为能够更新，请通过调用 [NotificationManager.notify()](https://developer.android.com/reference/android/app/NotificationManager.html?hl=zh-cn" \l "notify(int, android.app.Notification)) 发出带有通知 ID 的通知。 要在发出之后更新此通知，请更新或创建[NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html?hl=zh-cn) 对象，从该对象构建 [Notification](https://developer.android.com/reference/android/app/Notification.html?hl=zh-cn) 对象，并发出与之前所用 ID 相同的 [Notification](https://developer.android.com/reference/android/app/Notification.html?hl=zh-cn)。如果之前的通知仍然可见，则系统会根据 [Notification](https://developer.android.com/reference/android/app/Notification.html?hl=zh-cn) 对象的内容更新该通知。相反，如果之前的通知已被清除，系统则会创建一个新通知：



* **删除通知**

除非发生以下情况之一，否则通知仍然可见：

1、用户单独或通过使用“全部清除”清除了该通知（如果通知可以清除）。

2、用户点击通知，且您在创建通知时调用了 [setAutoCancel()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html?hl=zh-cn" \l "setAutoCancel(boolean))。

3、您针对特定的通知 ID 调用了 [cancel()](https://developer.android.com/reference/android/app/NotificationManager.html?hl=zh-cn" \l "cancel(int))。此方法还会删除当前通知。

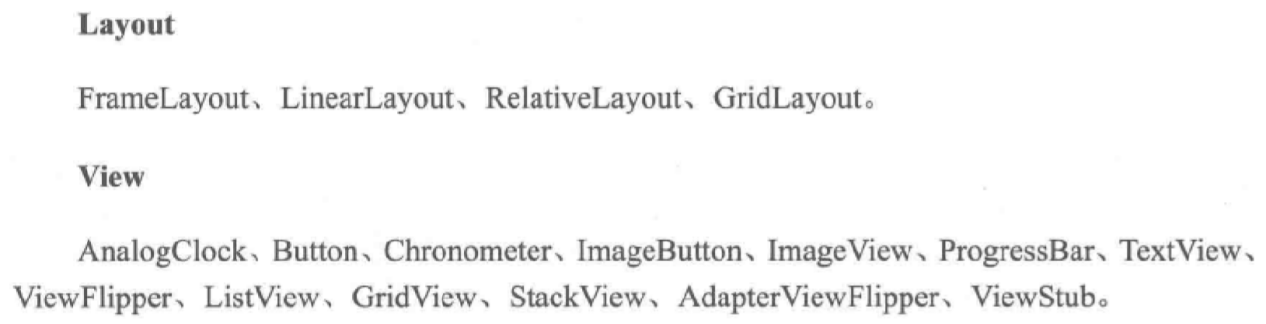
4、您调用了 [cancelAll()](https://developer.android.com/reference/android/app/NotificationManager.html?hl=zh-cn" \l "cancelAll()) 方法，该方法将删除之前发出的所有通知

5、关闭常驻（前台）通知，即以startForeground开启的通知，只需要调用stopForeground方法即可。

**自定义布局通知RemoteViews**

自定义通知布局必须通过RemoteViews来实现，**RemoteViews表示的是一个View结构，它的作用是在其他进程中显示并更新界面。**它在Android中使用的场景有两种：通知栏和桌面小部件的界面。它们的View界面不能直接在Android四大组件中直接更新，这是因为它们都运行在其它的进程中，确切的说是在系统的SystemServer进程。为了跨进程更新界面，RemoteViews提供了一系列set方法，并且这些方法只是View的全部方法子集。

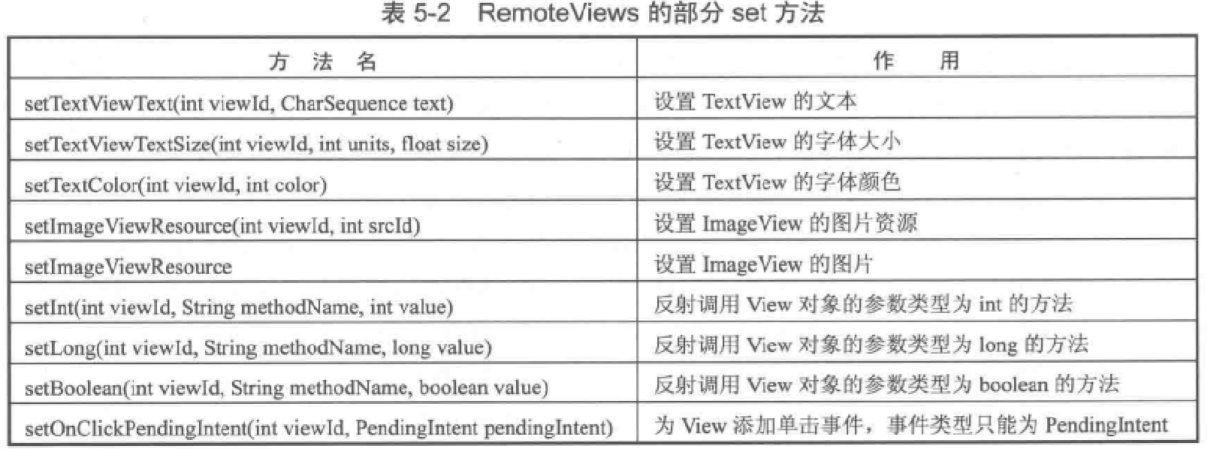
* **RemoteViews所支持加载的View类型**



RemoteViews只支持加载以上所列出的View类型，不支持它们的子类以及其他的类型否则会抛出异常。

* **RemoteViews更新页面**

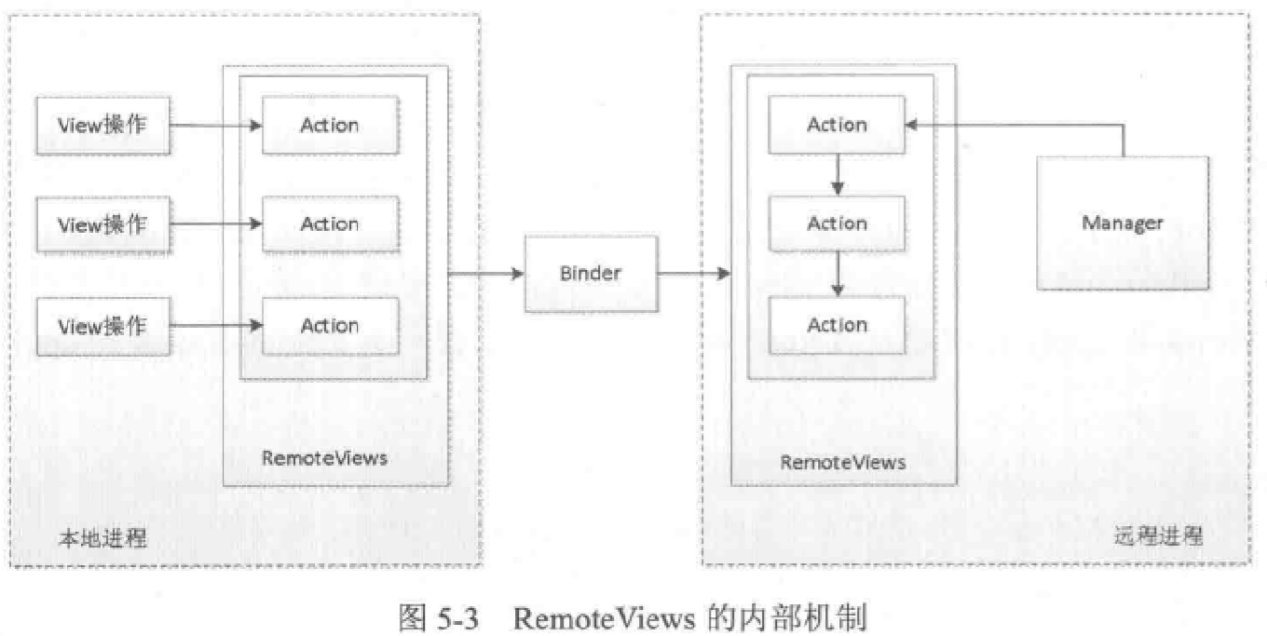
RemoteViews没有提供findViewById方法，因此我们无法在当前进程直接访问远程进程的View元素，而必须通过一系列set方法来完成，大部分set方法是通过反射来完成的：



* **RemoteViews内部机制**

通知栏部件由NotificationManager管理着，而NotificationManager通过Binder和SystemServer进程中NotificationMnagerService进行通信，所以通知栏中的布局文件是在NotificationMnagerService中被加载的，而它运行在系统的SystemServer进程中，被视为远程进程，和我们应用的进程正好构成了跨进程通信的场景，理论上的内部机制如下：

1. RemoteViews会通过Binder传递到SystemServer进程中，这里RemoteViews实现了Parcelable接口，所以它可以跨进程传输。
2. 系统根据RemoteViews中的包名去获取我们应用中的资源，并通过LayoutInflater去加载RemoteViews中的布局文件，在SystemServer中加载的布局文件就是一个普通的View，只不过对于我们的app进程而言它是一个RemoveViews而已。
3. 接着系统会对View进行一系列更新操作，这些更新就是我们通过set方法来提交的。set方法对View所做的更新并不是立即执行的，是在RemoteViews内部记录所有的操作，具体执行的时机是在RemoteViews加载完了之后才会被执行，这样我们就可以在SystemServer进程中显示界面了，这就是我们见到的通知栏或者是桌面小部件。当需要更新SystemServer进程中的界面，可以再次通过set方法来提交更新任务，具体的操作也是在SystemServer进程中执行。
4. RemoteViews界面更新set方法的机制：从理论上来讲，系统完全可以通过Binder去支持所有的View和View操作，但是这样会引起大量的IPC操作，从而影响效率。为了解决这个问题，系统并没有通过Binder去支持View的跨进程操作。而是提供了一个Action的概念，我们每调用一次set方法，RemoteViews中就会添加一个对应的Action对象，一个Action对象中就封装着一个View操作。当我们通过NotificationManager的notify方法来提交我们的更新时，这些Action就会被传递到远程进程中并被依次执行：

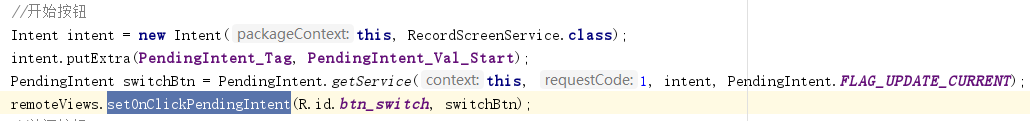


远程进程RemoteViews的apply方法进行View的更新操作，这个apply方法内部则会去遍历所有的Action对象并调用Action对象的apply方法，所以具体的View更新操作是由Action的apply方法来完成的。这种通过在远程批量执行RemoteViews的修改操作就避免了大量的IPC操作提高了程序的性能。

* **RemoteViews添加单击事件**

RemoteViews只支持发起PendingIntent,不支持onClickListener那种模式，另外需要注意setOnClickPendingIntent、setOnClickFillInIntent、setPendingIntentTemplate之间的区别和联系：setOnClickPendingIntent用于给普通的View添加点击事件；如果要给ListView和StackView中的Item添加点击事件，必须将setOnClickFillInIntent、setPendingIntentTemplate组合才能使用。

例如：在一个Service中心发送了一个通知，并在服务中相应通知布局中按钮的点击事件：



需要在Service的onStartCommand方法中接收这个点击事件的回调：

