最简单的notification示例代码

```
NotificationManager mNotificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(MainActivity.this);
                mBuilder.setContentTitle("测试标题")//设置通知栏标题
                .setContentText("测试内容") //设置通知栏显示内容
              //  .setContentIntent(getDefalutIntent(Notification.FLAG_AUTO_CANCEL)) //设置通知栏点击意图
            //  .setNumber(number) //设置通知集合的数量
                .setTicker("测试通知来啦") //通知首次出现在通知栏，带上升动画效果的
                .setWhen(System.currentTimeMillis())//通知产生的时间，会在通知信息里显示，一般是系统获取到的时间
                .setPriority(Notification.PRIORITY_DEFAULT) //设置该通知优先级
            //  .setAutoCancel(true)//设置这个标志当用户单击面板就可以让通知将自动取消
                .setOngoing(false)//ture，设置他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
                .setDefaults(Notification.DEFAULT_VIBRATE)//向通知添加声音、闪灯和振动效果的最简单、最一致的方式是使用当前的用户
                //Notification.DEFAULT_ALL  Notification.DEFAULT_SOUND 添加声音 // requires VIBRATE permission
                .setSmallIcon(R.drawable.ic_launcher);//设置通知小ICON
                Notification notification = mBuilder.build();
                notification.flags = Notification.FLAG_NO_CLEAR;
                mNotificationManager.notify(11111, mBuilder.build());
```

解释：

Notificaiton状态通知栏：http://blog.csdn.net/vipzjyno1/article/details/25248021/

功能作用

1.显示接收到短消息、即使消息等信息（如QQ、微信、新浪、短信）
2.显示客户端的推送消息（如有新版本发布，广告，推荐新闻等）
3.显示正在进行的事物（例如：后台运行的程序）（如音乐播放器、版本更新时候的下载进度等）

思维导图结构

思维导图的大体结构（按照各个节点延伸拓展学习）

Notificaiton -- service -- BroadcastReceiver -- Intent（flag、Action等属性应用）-- PendingIntent

感慨：

一个Notificaiton通知的拓展使用就要涉及与4大组建的配合，所以学好整体的知识体系。

联系：

1.由于service 是在后台运行，所以它意图做什么我们看不到，可以通过Notificaiton 来显示提醒（如音乐的后台播放）。
2.service服务和BroadcastReceiver广播相结合，在加上Notificaiton 显示（如程序的后台更新）。
3.Intent作为意图处理，和Notificaiton的点击时间紧密结合在了一起，并且与BroadcastReceiver和service的联系也紧密不可以分割。
（service 在后台之后通过BroadcastReceiver来通知Notificaiton 显示相关东西，在通过Intent完成用户的意图操作）

相关文档：Activity启动模式 及 Intent Flags 与 栈 的关联分析

对应的官方链接

设计文档 ：
官方：http://developer.android.com/design/patterns/notifications.html
译文：http://adchs.github.io/patterns/notifications.html
使用教程 ：http://developer.android.com/training/notify-user/index.html
开发文档 ：http://developer.android.com/reference/android/app/Notification.html

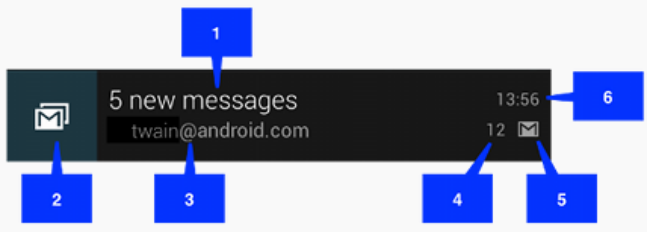
大体了解

Notification支持文字内容显示、震动、三色灯、铃声等多种提示形式，在默认情况下，Notification仅显示消息标题、消息内容、送达时间这3项内容。以下就是通知的基本布局。
通知的基本布局：

普通视图：

高度64dp

大试图的通知在展开前也显示为普通视图



元素：

1. 标题　Title/Name
2. 大图标　Icon/Photo
3. 内容文字
4. 内容信息　MESSAGE
5. 小图标 Secondary Icon
6. 通知的时间 Timestamp,默认为系统发出通知的时间，也可通过setWhen()来设置

相关分析

状态通知栏主要涉及到2个类：　Notification 和 NotificationManager

Notification为通知信息类，它里面对应了通知栏的各个属性

NotificationManager ：　是状态栏通知的管理类，负责发通知、清除通知等操作。

注意：NotificationManager 是一个系统Service，所以必须通过 `getSystemService(NOTIFICATION_SERVICE)`方法来获取，方法如下。

```java
NotificationManager mNotificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
```

使用步骤：

流程模块：

第一步：

创建一个通知栏的Builder构造类 （Create a Notification Builder）

第二步：

定义通知栏的Action （Define the Notification's Action）

第三步：

设置通知栏点击事件　（Set the Notification's Click Behavior）

第四步：

通知　（Issue the Notification）

代码模块：

实现系统默认的通知栏效果：
第一步：获取状态通知栏管理：

```java
NotificationManager mNotificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
```

第二步：实例化通知栏构造器NotificationCompat.Builder：

```java
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this);
```

第三步：对Builder进行配置：

```java
mBuilder.setContentTitle("测试标题")//设置通知栏标题
    .setContentText("测试内容") /<span style="font-family: Arial;">/设置通知栏显示内容</span>
    .setContentIntent(getDefalutIntent(Notification.FLAG_AUTO_CANCEL)) //设置通知栏点击意图
//  .setNumber(number) //设置通知集合的数量
    .setTicker("测试通知来啦") //通知首次出现在通知栏，带上升动画效果的
    .setWhen(System.currentTimeMillis())//通知产生的时间，会在通知信息里显示，一般是系统获取到的时间
    .setPriority(Notification.PRIORITY_DEFAULT) //设置该通知优先级
//  .setAutoCancel(true)//设置这个标志当用户单击面板就可以让通知将自动取消
    .setOngoing(false)//ture，设置他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)
    .setDefaults(Notification.DEFAULT_VIBRATE)//向通知添加声音、闪灯和振动效果的最简单、最一致的方式是使用当前的用户默认设置，使用defaults属性，可以组合
    //Notification.DEFAULT_ALL  Notification.DEFAULT_SOUND 添加声音 // requires VIBRATE permission
    .setSmallIcon(R.drawable.ic_launcher);//设置通知小ICON
```

对应的各个方法的属性（部分方法以上代码中已经作注释，就不再介绍）：

（1）方法：设置提醒标志符Flags

功能：提醒标志符，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
有2种设置方法：
1.实例化通知栏之后通过给他添加.flags属性赋值。

```java
Notification notification = mBuilder.build();
notification.flags = Notification.FLAG_AUTO_CANCEL;
```

2.通过setContentIntent(PendingIntent intent)方法中的意图设置对应的flags

```java
public PendingIntent getDefalutIntent(int flags){
    PendingIntent pendingIntent= PendingIntent.getActivity(this, 1, new Intent(), flags);
    return pendingIntent;
}
```

提醒标志符成员：
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

（2）方法：.setDefaults(int defaults)　（NotificationCompat.Builder中的方法，用于提示）

功能：向通知添加声音、闪灯和振动效果的最简单、使用默认（defaults）属性，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

（3）方法：setVibrate(long[] pattern)

功能：设置震动方式。
使用：

```java
.setVibrate(new long[] {0,300,500,700});
```

实现效果：延迟0ms，然后振动300ms，在延迟500ms，接着在振动700ms。
以上方法的还有种写法是

```java
mBuilder.build().vibrate = new long[] {0,300,500,700};
```

以此类推，2种写法都可以。
如果希望设置默认振动方式，设置了方法（2）中默认为DEFAULT_VIBRATE 即可。

（4）方法：.setLights(intledARGB ,intledOnMS ,intledOffMS)

功能：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。
描述：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。

使用：

```java
.setLights(0xff0000ff, 300, 0)
```

同理，以下方法也可以设置同样效果：

```java
Notification notify = mBuilder.build();
notify.flags = Notification.FLAG_SHOW_LIGHTS;
notify.ledARGB = 0xff0000ff;
notify.ledOnMS = 300;
notify.ledOffMS = 300;
```

如果希望使用默认的三色灯提醒，设置了方法（2）中默认为DEFAULT_LIGHTS即可。

（5）方法：.setSound(Uri sound)

功能：设置默认或则自定义的铃声，来提醒。

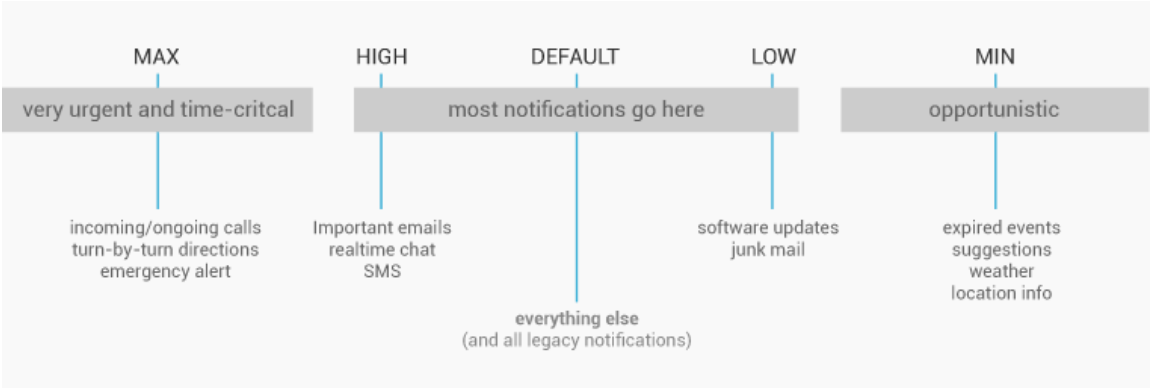
```java
//获取默认铃声
.setDefaults(Notification.DEFAULT_SOUND)
//获取自定义铃声
.setSound(Uri.parse("file:///sdcard/xx/xx.mp3"))
//获取Android多媒体库内的铃声
.setSound(Uri.withAppendedPath(Audio.Media.INTERNAL_CONTENT_URI, "5"))
```

同理相同效果的另一种设置方法这边就不讲， 和上面的都是一样的。

（6）方法：.setPriority(int pri)

功能：设置优先级
对应优先级描述如下图：

优先级	用户
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需要立即处理的。
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。
DEFAULT	默认优先级用于没有特殊优先级分类的通知。
LOW	低优先级可以通知用户但又不是很紧急的事件。
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。



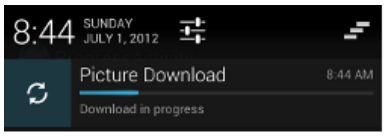
对应属性（作用看上图就可知道）：
Notification.PRIORITY_DEFAULT
Notification.PRIORITY_HIGH
Notification.PRIORITY_LOW
Notification.PRIORITY_MAX
Notification.PRIORITY_MIN

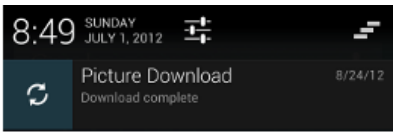
（7）方法：setOngoing(boolean ongoing)

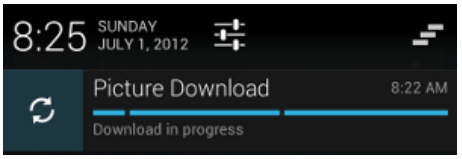
功能：设置为ture，表示它为一个正在进行的通知。他们通常是用来表示一个后台任务,用户积极参与(如播放音乐)或以某种方式正在等待,因此占用设备(如一个文件下载,同步操作,主动网络连接)

（8）方法：setProgress(int max, int progress,boolean indeterminate)

属性：max:进度条最大数值 、progress:当前进度、indeterminate:表示进度是否不确定，true为不确定，如下第3幅图所示 ，false为确定下第1幅图所示
功能：设置带进度条的通知，可以在下载中使用
效果图如下：







注意：此方法在4.0及以后版本才有用，如果为早期版本：需要自定义通知布局，其中包含ProgressBar视图
使用：如果为确定的进度条：调用setProgress(max, progress, false)来设置通知，在更新进度的时候在此发起通知更新progress，并且在下载完成后要移除进度条，通过调用setProgress(0, 0, false)既可。
如果为不确定（持续活动）的进度条，这是在处理进度无法准确获知时显示活动正在持续，所以调用`setProgress(0, 0, true)`，操作结束时，调用`setProgress(0, 0, false)`并更新通知以移除指示条

第四步：设置通知栏PendingIntent（点击动作事件等都包含在这里）

在第三步中，没有提到一个方法，就是setContentIntent(PendingIntent intent)这个方法，这里拿到这里讲。

知识点

1）什么是PendingIntent

PendingIntent和Intent略有不同，它可以设置执行次数，主要用于远程服务通信、闹铃、通知、启动器、短信中，在一般情况下用的比较少。

2）PendingIntent什么用

Notification支持多种Intent来响应单击事件、消除事件、处理紧急状态的全屏事件等。
这里就用到了setContentIntent(PendingIntent intent)来处理以上这么多的事件。

3）相关属性和方法

属性：

PendingIntent的位标识符：

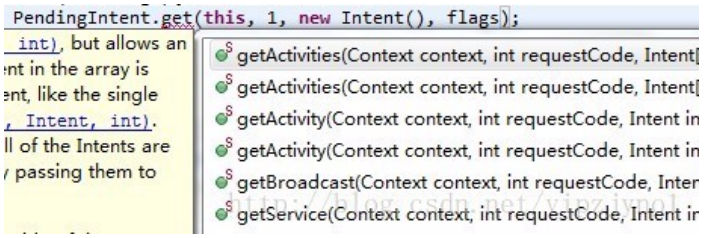
FLAG_ONE_SHOT　表示返回的PendingIntent仅能执行一次，执行完后自动取消

FLAG_NO_CREATE　表示如果描述的PendingIntent不存在，并不创建相应的PendingIntent，而是返回NULL

FLAG_CANCEL_CURRENT　表示相应的PendingIntent已经存在，则取消前者，然后创建新的PendingIntent，这个有利于数据保持为最新的，可以用于即时通信的通信场景

FLAG_UPDATE_CURRENT　表示更新的PendingIntent

方法：



可以看出，它支持多种相应方式，有Activity、Broadcast、Service，就根据你自身需求去选择。

在各种情况下情况下它还会根据各种情况出发效果：
contentIntent：在通知窗口区域，Notification被单击时的响应事件由该intent触发；
deleteIntent：当用户点击全部清除按钮时，响应该清除事件的Intent；
fullScreenIntent：响应紧急状态的全屏事件（例如来电事件），也就是说通知来的时候，跳过在通知区域点击通知这一步，直接执行fullScreenIntent代表的事件。

例如：在执行了点击通知之后要跳转到指定的XXX的Activity的时候，可以设置以下方法来相应点击事件：

```java
Intent intent = new Intent(context,XXX.class);
PendingIntent pendingIntent = PendingIntent.getActivity(context, 0, intent, 0);
mBuilder.setContentIntent(pendingIntent)
```

例如：在执行了清空全部的通知操作时候，可以设置以下方法来相应这个事件：
采用setDeleteIntent(PendingIntent intent)方法或按照以下写法

```java
Intent deleteIntent = new Intent();
deleteIntent.setClass(context, XXXReceiver.class);
deleteIntent.setAction(DELETE_ACTION);
notification.deleteIntent = PendingIntent.getBroadcast(context, 0, deleteIntent, 0);
```

例如：在响应紧急事件（如来电）时候，可以设置以下方法来相应这个事件：
采用setFullScreenIntent(PendingIntent intent, boolean highPriority)

第五步，最简单的一部，发送通知请求

```java
mNotificationManager.notify(notifyId, mBuilder.build());
```

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