Useful Services & Utilities in Android

Some useful Android Services & Utilities

Alarm, PowerManager

Notification, SMS, Call

A.R. Kazemi

Services/Utilities

- Using **Alarm** Events
- Making **Notification**s
- Playing **Vibration**s
- Sending **SMS**
- Making **Call**

Access the Alarm Manager Service

- Get the reference to Alarm Service as AlarmManager:

```
AlarmManager am = (AlarmManager)
                  context.getSystemService(Context.ALARM_SERVICE);
```

- AlarmManager class Methods:

Method	Description
set()	Schedules an alarm for one time.
setInexactRepeating()	Schedules an alarm with inexact repeating. Trigger time doesn't follow any strict restriction.
setRepeating()	Schedules an alarm with exact repeating time.
setTime()	Sets the system's wall clock time.
setTimeZone()	Sets the system's default time zone.

Create a One Time Alarm Event

- Set an Alarm Wakeup event
  - Alarm events are set using an **Intent** which is nested in a **PendingIntent** as their callback arguments/parameters

```
AlarmManager am =(AlarmManager)
                    context.getSystemService(Context.ALARM_SERVICE);

// Create an Intent with Your BroadcastReceiver class which needs
//   to receive the AlarmEvent
Intent intent = new Intent(context,
                           AlarmManagerBroadcastReceiver.class);
intent.putExtra("MY_FLAG_PARAM", Boolean.TRUE);
intent.putExtra("MY_OTHER_PARAM", "Another Param which is String");

PendingIntent pI = PendingIntent.getBroadcast(context,0, intent,0);

am.set(AlarmManager.RTC_WAKEUP,System.currentTimeMillis()+1000, pI);
```

Create a Repeating Alarm Event

- Same as before but just replace call of:
  - **AlamManager.set()** by **AlarmManger.setRepeating()**

```
AlarmManager am =(AlarmManager)
                    context.getSystemService(Context.ALARM_SERVICE);

// Create an Intent with Your BroadcastReceiver class which needs
//   to receive the AlarmEvent
Intent intent = new Intent(context,
                           AlarmManagerBroadcastReceiver.class);
intent.putExtra("MY_FLAG_PARAM", Boolean.TRUE);
intent.putExtra("MY_OTHER_PARAM", "Another Param which is String");

PendingIntent pI = PendingIntent.getBroadcast(context,0, intent,0);
am.setRepeating(AlarmManager.RTC_WAKEUP,
                System.currentTimeMillis(), period  , pI);
```

Cancle an Alarm Event

- Use following method by passing the same intent which created the Alarm event
  - **AlamManager.cancel()**

```
Intent intent = new Intent(context, AlarmManagerBroadcastReceiver.class);
PendingIntent sender = PendingIntent.getBroadcast(context, 0, intent, 0);

AlarmManager alarmManager = (AlarmManager)
                            context.getSystemService(Context.ALARM_SERVICE);
alarmManager.cancel(sender);
```

Handle the Alarm Event when it occur

- Just implement and register a BroadcastReceiver component

```
public class AlarmManagerBroadcastReceiver extends BroadcastReceiver {
  final public static String ONE_TIME = "onetime";
  @Override
  public void onReceive(Context context, Intent intent) {
    // You can use the AlarmEvent Callback params which user has
    //  set in the intent when he/she has created the Alarm Event
    Bundle extras = intent.getExtras();
    StringBuilder msgStr = new StringBuilder();
    if(extras != null && extras.getBoolean(ONE_TIME, Boolean.FALSE)){
        msgStr.append("One time Timer: ");
    }

    Format formatter = new SimpleDateFormat("hh:mm:ss a");
    msgStr.append(formatter.format(new Date()));

    Toast.makeText(context, msgStr, Toast.LENGTH_LONG).show();
  }
}
```

Register Your BroadcastReceiver

- Don't forget to register your Broadcast Receiver either
  - Statically in App Manifest, or

```
<manifest android:versioncode="1" android:versionname="1.0"
      package="com.rakesh.alarmmanagerexample"
      xmlns:android="http://schemas.android.com/apk/res/android">
    ...
    <application android:icon="@drawable/ic_launcher"
      android:label="@string/app_name" android:theme="@style/AppTheme">
       <activity android:label="@string/title_activity_alarm_manager"
          android:name="my.alarmmanagerexample.AlarmManagerActivity">
           <intent-filter>
               <action android:name="android.intent.action.MAIN"/>
               <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
       </activity>
       <receiver android:name="my.alarmmanagerexample.AlarmManagerBroadcastReceiver">
       </receiver>
    </application>
</manifest>
```

Repeating Alarms and Android 5.1+

- Problem!
  - In Androids V 5.1+ there is a minimum period of 1 minute for repeating alarms.
  - Also android device manufacturers such as LG, SAMSUNG, HUAWI, etc may config different minimum periods such as 2min, 5min
  - So what if we need repeating alarms of say 10sec?
- Solution
  - Set a onetime alarm and set the next alarm in the onReceive() method of the alarm BroadcastReceiver

Receiving alarms while device is in sleep mode

- Problem!
  - What if when the alarm is to be activated, the device is in sleep mode and screen is off
- Solution 1: use PowerManager Wake Lock
  - Acquire the power manager wake lock at start of onReceive()

```
PowerManager.WakeLock wakeLock;
PowerManager pm = (PowerManager) ctx.getSystemService(Context.POWER_SERVICE);
wakeLock = pm.newWakeLock(PowerManager.FULL_WAKE_LOCK |
        PowerManager.ACQUIRE_CAUSES_WAKEUP |
        PowerManager.ON_AFTER_RELEASE, MainActivity.APP_TAG);
wakeLock.acquire();
```

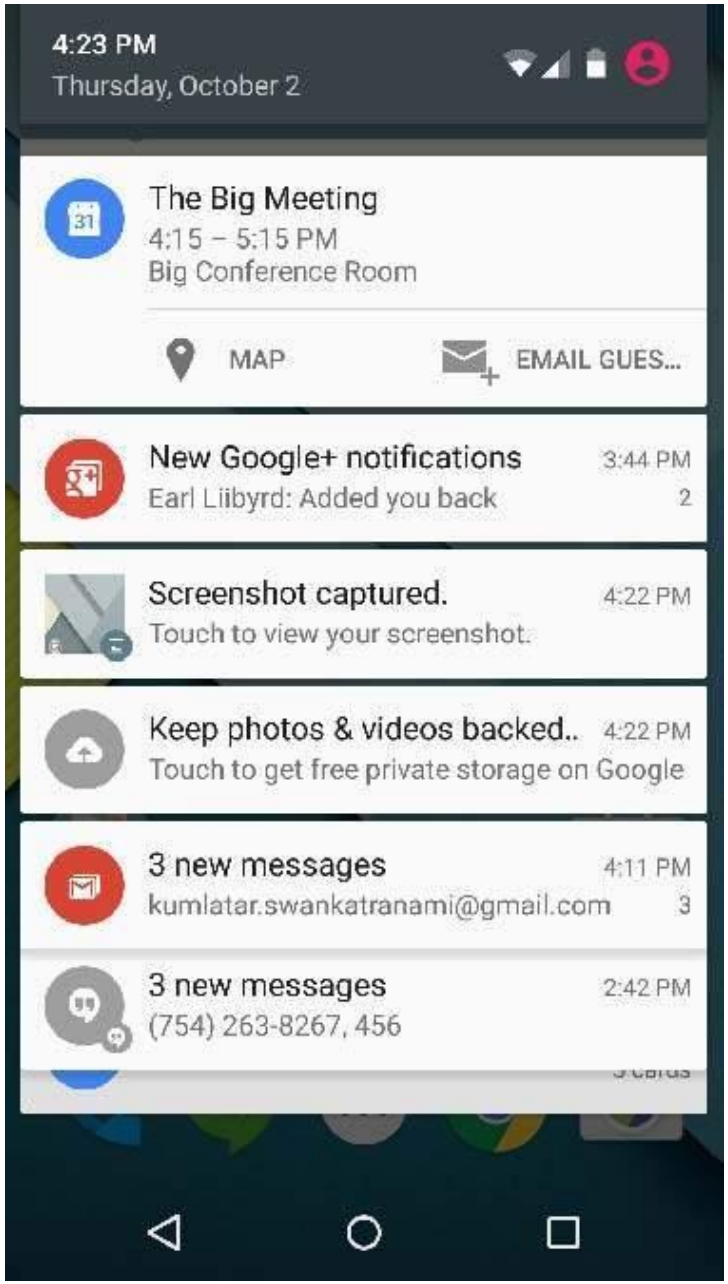
  - Release it at the end of your task

```
wakeLock.release(); wakeLock = null;
```

- Solution 2: extend the **WakefulBroadcastReceiver**
  - Your **AlarmBroadcastReceiver** class should extend this class instead of the basic **BroadcastReceiver** class

Android Notifications

- Short Messages which are shown in taskbar/top menu
- Notify user about important events/changes/actions



Android Notifications

- Create a Notification Builder Object

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

- Set the notification attributes in it

```
mBuilder.setSmallIcon(R.drawable.notification_icon);
mBuilder.setContentTitle("Notification Alert, Click Me!");
mBuilder.setContentText("Hi, This is Android Notification Detail!");
```

- You may attach Actions to happen when user clicks it
  - See next Slide
- Finally call **mBuilder.build()** to create the notification object

Android Notifications

- Attaching an action to the notification using
  - An Intent nested in a Pending Intent

```
Intent resultIntent = new Intent(this, ResultActivity.class);

// Add the Intent that starts the Activity to top of the stack
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
stackBuilder.addParentStack(ResultActivity.class);
stackBuilder.addNextIntent(resultIntent);

PendingIntent pI = stackBuilder.getPendingIntent(0,
                                PendingIntent.FLAG_UPDATE_CURRENT);
mBuilder.setContentIntent(pI);
```

Android Notifications

- Finally issue it
  - First get access to Notification Manager Service

```
NotificationManager nm = (NotificationManager)
                    getSystemService(Context.NOTIFICATION_SERVICE);
```

  - Now create and submit the notification
    - Create by calling **NotificationBuilder.build()**
    - Submit by calling the **NotificationManager.notify()**

```
// notificationID allows you to update the notification later on.
mNotificationManager.notify(notificationID, mBuilder.build());
```

Android Device Vibration

- Access the Vibration Service

```
Vibrator v = (Vibrator)getSystemService(Context.VIBRATOR_SERVICE);
```

- Now call simple or time-pattern based vibration

```
// 1- virrate for a fixed number of milli-seconds
v.vibrate(2000);

// 2- If you want you can adopt an off/on time pattern
long pattern[] = {0,800,200,1200,300,2000,400,4000};
v.vibrate(pattern, 1); // 2nd argument is for repetition
```

- **Notice:**
  - If you want to apply long vibration call it in a background service
  - In all cases you need the **vibration permission** (add in **Manifest**)

```
<uses-permission android:name="android.permission.VIBRATE" />
```

Android Sending SMS

- Send SMS using the **SmsManager API**:

```
SmsManager smsManager = SmsManager.getDefault();
smsManager.sendTextMessage("phoneNo", null, "sms message", null, null);
```

- Sending SMS using the Built-in SMS App/Activity

```
Intent sendIntent = new Intent(Intent.ACTION_VIEW);
sendIntent.putExtra("sms_body", "default content");
sendIntent.setType("vnd.android-dir/mms-sms");
startActivity(sendIntent);
```

- **Notice:** in either cases you need SMS send **permission** (add in **Manifest**)

```
<uses-permission android:name="android.permission.SEND_SMS" />
```

Android Making a Call

- Making Call using the Built-in SMS App/Activity
  - **Notice:** the **ACTION_CALL** action of intent

```
Intent phoneIntent = new Intent(Intent.ACTION_CALL);
```

- Set the phone number to dial as a Uri Data
  - **tel:phone number**

```
phoneIntent.setData(Uri.parse("tel:91-000-000-0000"));
```

- Finally start the Phone Dialer Activity
  - **[Context.]startActivity(phoneIntent);**
- Notice: You need PHONE_CALL permission

```
<uses-permission android:name="android.permission.CALL_PHONE" />
```

Android Making a Call