Android Notification

Sang Shin
Michèle Garoche
www.javapassion.com
“Learn with Passion!”

Disclaimer

- Portions of this presentation are modifications based on work created and shared by the Android Open Source Project
  - http://code.google.com/policies.html
- They are used according to terms described in the Creative Commons 2.5 Attribution License
  - http://creativecommons.org/licenses/by/2.5/

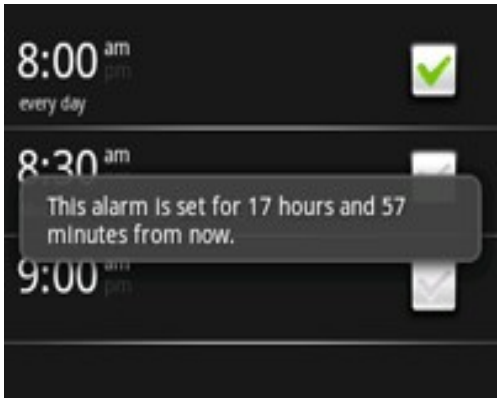
Topics

- Toast notification
- Status bar notification
- Dialog notification (we already covered this)
- Alarm manager

Toast Notification

What is Toast Notification (Message)?

- A toast notification is a message that pops up on the surface of the window.
- It only fills the amount of space required for the message and the user's current activity remains visible and interactive.
- The notification automatically fades in and out, and does not accept interaction events.
- Because a toast can be created from a background Service, it appears even if the application isn't visible.



How to create Toast Message?

```
Toast.makeText(
        getApplicationContext(),          // Context
        R.string.toast_message,           // Get string resource to display
        Toast.LENGTH_LONG).show(); // Make sure you call show()
```

Status Bar Notification

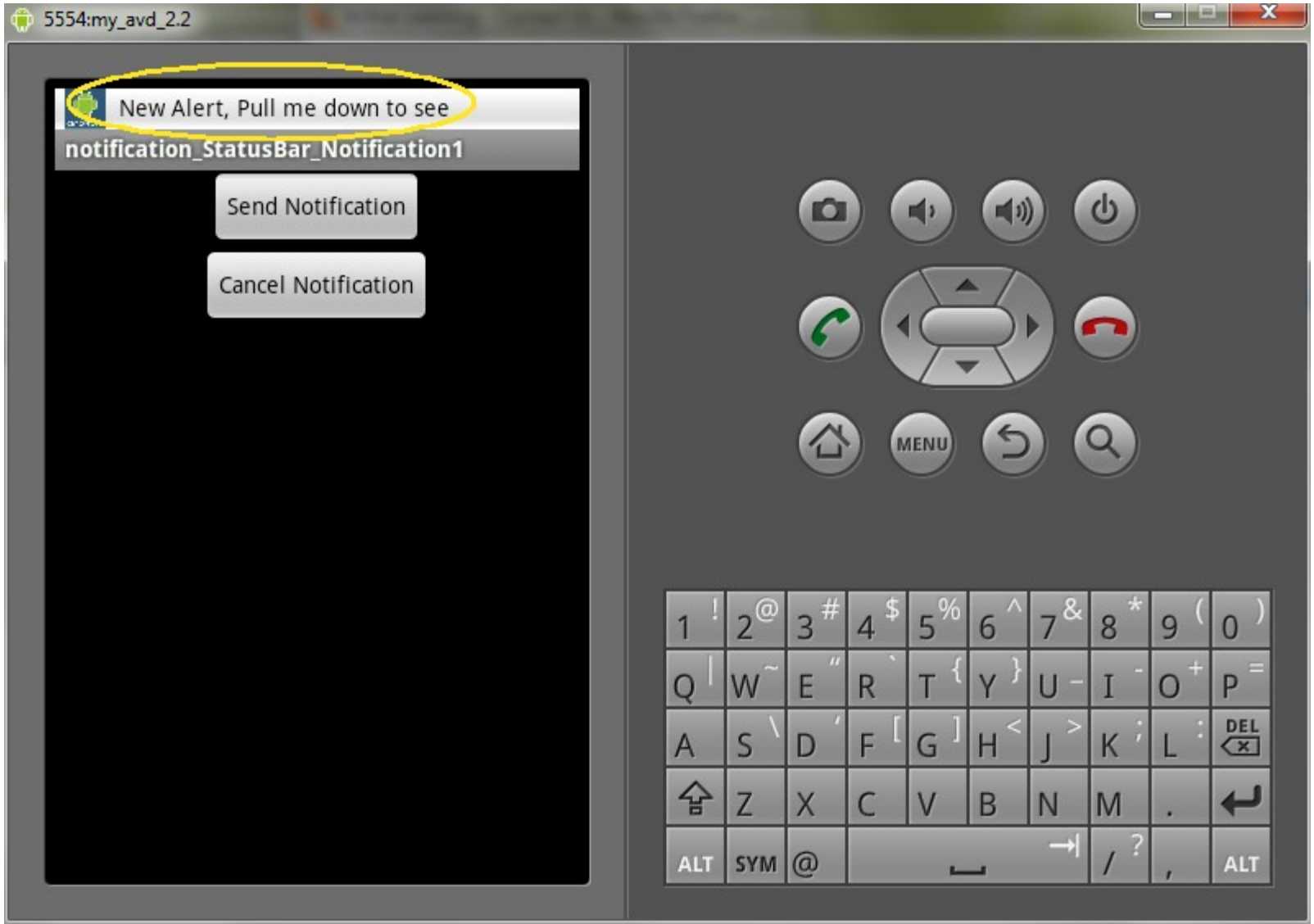
Phases of Status Bar Notification

- Phase #1 - A status bar notification adds an icon to the system's status bar (with an optional ticker-text message)
- Phase #2 - When the icon/ticker text message is pulled down, an expanded message (notification detail) gets displayed
- Phase #3 - When the user selects the expanded message (notification detail), Android fires an Intent that is defined by the notification (usually to launch an Activity)

Configuration of Status Bar Notification

- You can also configure the notification to alert the user with a sound, a vibration, and flashing lights on the device.

Phase #1: Icon and Ticker-text Message

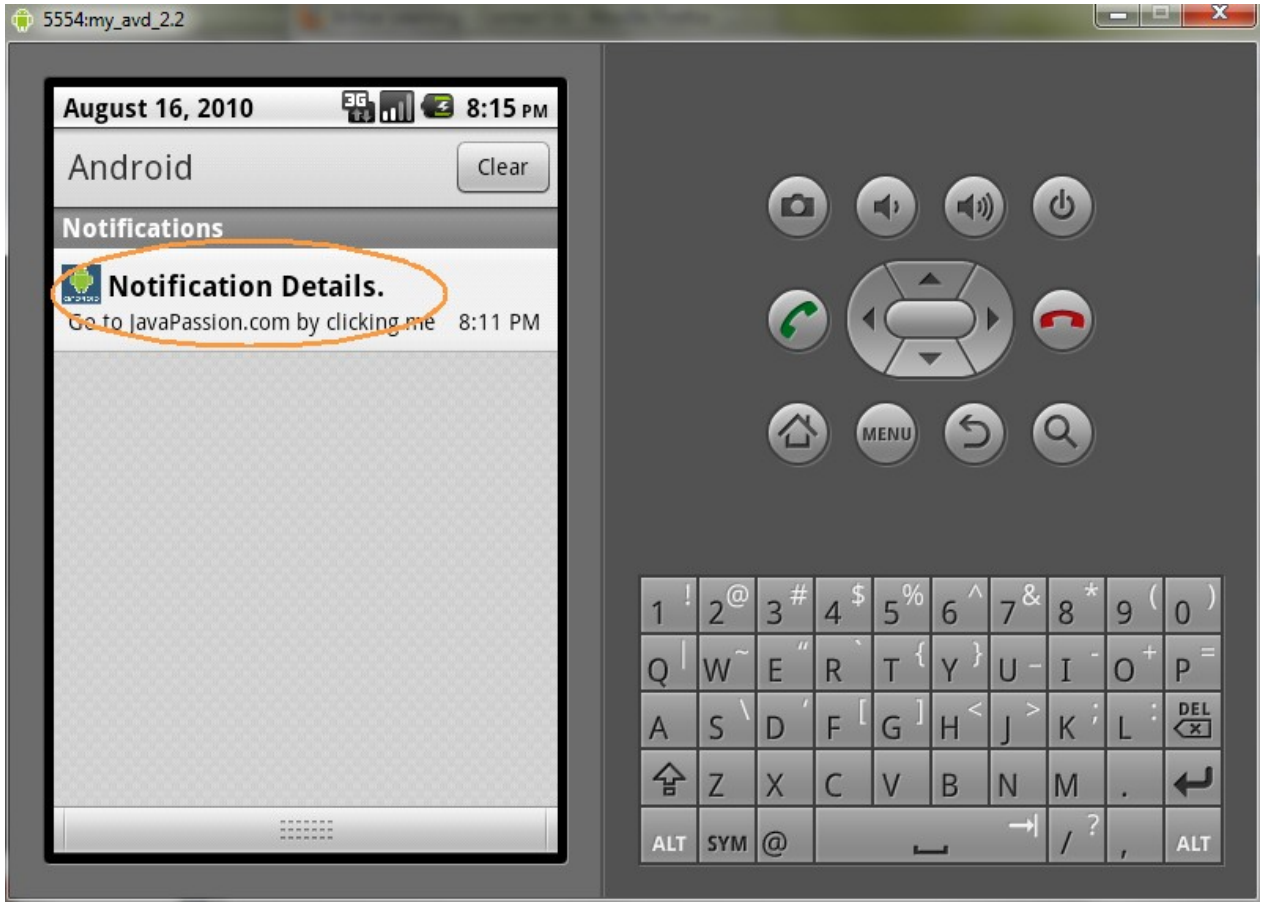


Code that Displays Icon/Ticker-text

```
// Get a reference to the NotificationManager:
NotificationManager  mNotificationManager =
                    (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

// Instantiate the Notification.  Use an instance of the Notification class to
// define the properties of your status bar notification, such as the status bar icon,
// the notification message (ticker-text), and extra settings such as a sound to play.
final Notification mNotification = new Notification(
        R.drawable.android,          // icon
        "New Alert, Pull me down to see Notification details!",  // notification message
        System.currentTimeMillis());     // current time

....
...

mNotificationManager.notify(
                    SIMPLE_NOTFICATION_ID,  // unique id within your application
                    mNotification);
```

Phase #2: Expanded Message (Notification Detail)

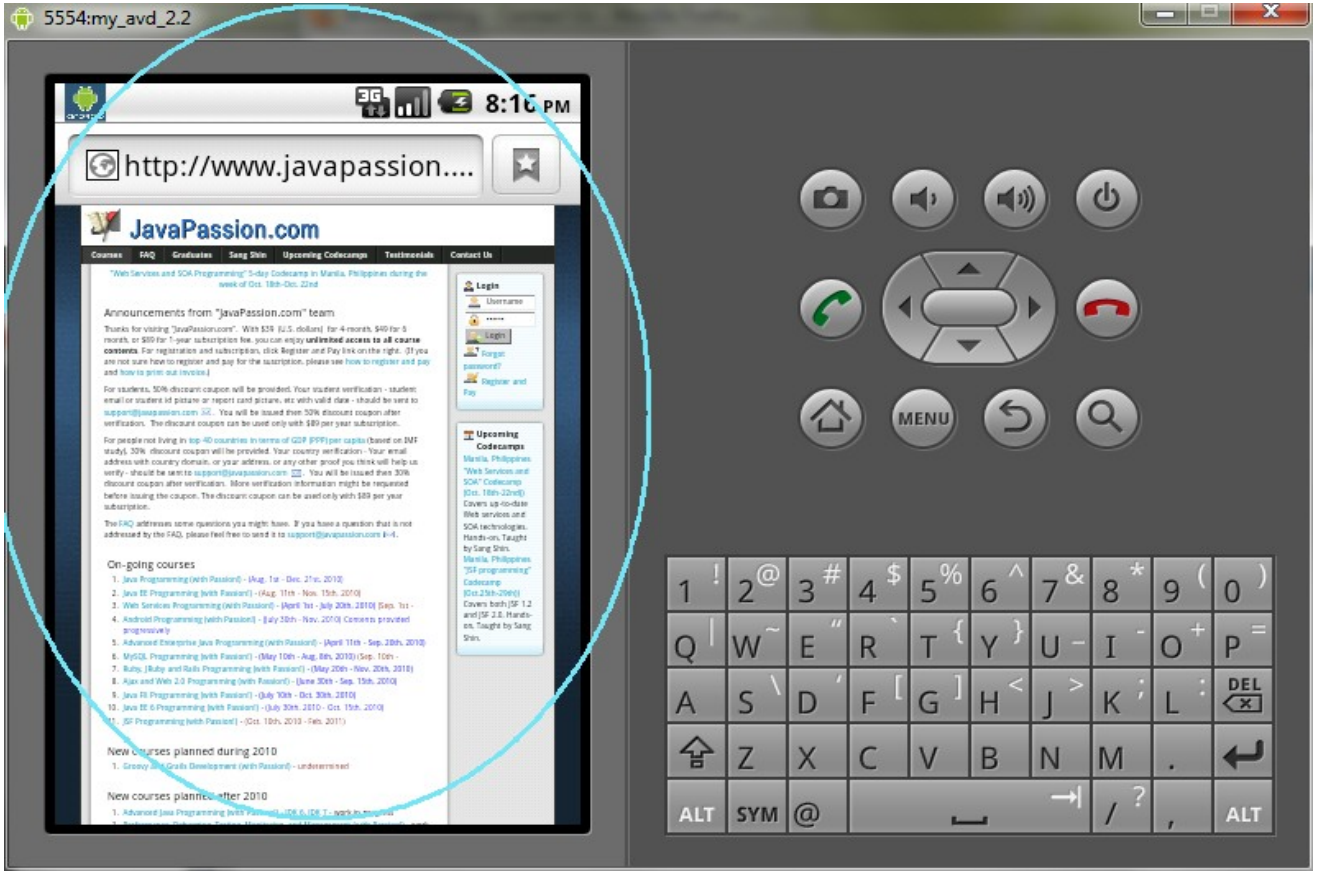
- Appears when icon/ticker-text is pulled down



Code that Displays Expanded Message (Notification Detail)

```
// Define the Notification's expanded message (notification detail)
// and Intent. The notification detail message is the one that gets
// displayed when a user drags the notification downward.
CharSequence contentTitle = "Notification Details.";
CharSequence contentText = "Go to JavaPassion.com by clicking me";


....
...

// Sets the contentView field to be a view with the standard "Latest Event"
// layout. "mPendingIntent" is an intent to launch when the user clicks
// the expanded notification
mNotification.setLatestEventInfo(getApplicationContext(),
                        contentTitle,
                        contentText,
                        mPendingIntent);
```

Phase #3: Activity That Gets Started When Notification Detail is Clicked



Code that Starts an Activity when Notification Detail is clicked

```
// The Intent is to define an action that gets executed when a
// user clicks the notification detail.
intent notifyIntent = new Intent(
                    android.content.Intent.ACTION_VIEW,
                    Uri.parse("http://www.javapassion.com"));

// The PendingIntent can be handed to other applications so that they can
// perform the action you described on your behalf at a later time.
// By giving a PendingIntent to another application, you are granting it the
// right to perform the operation you have specified as if the other application
// was yourself (with the same permissions and identity).
PendingIntent mPendingIntent = PendingIntent.getActivity(
                    StatusBarNotification.this, 0, notifyIntent,
                    android.content.Intent.FLAG_ACTIVITY_NEW_TASK);

// Sets the contentView field to be a view with the standard "Latest Event"
// layout. "mPendingIntent" is an intent to launch when the user clicks
// the expanded notification
mNotification.setLatestEventInfo(getApplicationContext(),
                    contentTitle,
                    contentText,
                    mPendingIntent);
```

When to Use What

Which one to use When?

- Use toast message for short text messages, such as "File saved," when you're fairly certain the user is paying attention to the screen.
- Use status bar notification when your application is working in a background Service and needs to notify the user about an event.
- If you need to alert the user about an event that occurs while your Activity is still in focus, consider using a Dialog Notification instead

Alarm Manager

What is AlarmManager?

- The *AlarmManager* class provides access to the system alarm services.
- These allow you to schedule sending an Intent to your target component at some point in the future
- Single or repeating
- When an alarm goes off, the Intent that had been registered for it is sent by the system, automatically starting the target component if it is not already running.

One shot alarm to Broadcast Receiver

```
// When the alarm goes off, we want to broadcast an Intent to our
// BroadcastReceiver.
Intent intent = new Intent(AlarmController.this,
                                    MyBroadcastReceiver.class);
PendingIntent pendingIntent = PendingIntent.getBroadcast(AlarmController.this,
                0, intent, 0);

// We want the alarm to go off 10 seconds from now.
Calendar calendar = Calendar.getInstance();
calendar.setTimeInMillis(System.currentTimeMillis());
calendar.add(Calendar.SECOND, 10);

// Schedule the alarm!
AlarmManager am = (AlarmManager)getSystemService(ALARM_SERVICE);
am.set(AlarmManager.RTC_WAKEUP, calendar.getTimeInMillis(), pendingIntent);
```

Repeating alarm to broadcast receiver

```
// When the alarm goes off, we want to broadcast an Intent to our
// BroadcastReceiver.
Intent intent = new Intent(AlarmController.this,
                                    MyBroadcastReceiver.class);
PendingIntent pendingIntent = PendingIntent.getBroadcast(AlarmController.this,
                0, intent, 0);

// We want the alarm to go off 10 seconds from now.
long firstTime = SystemClock.elapsedRealtime();
firstTime += 10*1000;

// Schedule the alarm!
AlarmManager am = (AlarmManager)getSystemService(ALARM_SERVICE);
am.setRepeating(AlarmManager.ELAPSED_REALTIME_WAKEUP,
                        firstTime, 7*1000, pendingIntent);
```

Repeating alarm to a service

```
// When the alarm goes off, we want to broadcast an Intent to our
// BroadcastReceiver.
Intent intent = new Intent(AlarmController.this,
                                    AlarmService.class);
PendingIntent pendingIntent = PendingIntent.getBroadcast(AlarmController.this,
                0, intent, 0);

// We want the alarm to go off 10 seconds from now.
long firstTime = SystemClock.elapsedRealtime();
firstTime += 10*1000;

// Schedule the alarm!
AlarmManager am = (AlarmManager)getSystemService(ALARM_SERVICE);
am.setRepeating(AlarmManager.ELAPSED_REALTIME_WAKEUP,
                        firstTime, 7*1000, pendingIntent);
```

Thank you!

Check JavaPassion.com Codecamps!
http://www.javapassion.com/codecamps
"Learn with Passion!"