Android Course Day 3

Lucio Cossio
Luis Mazoni

Course Agenda

Day 1

- **Development environment and tools**
- **Android project structure**
  - Source, tests , resources, manifest
- **Activity**
  - Lifecycle
  - Layout interaction
  - ActionBar
- **Practice**

Day 2

- **Review Day 1**
- **Resources**
  - String
  - Dimensions
  - Layout
- **Views**
  - TextView, Edit Text, Button
  - String resources
  - View listeners
- **Animation**
- **Practice**

Day 3

- **Review Day 2**
- **Intents**
  - Explicit
    - Extras
  - Implicit
    - Intent Filters
- **Android Manifest**
  - Overview
  - Add activities
- **Notifications**
- **Practice**

Day 4

- **Review Day 3**
- **Fragments**
  - Lifecycle
  - Fragment Manager
  - Arguments
- **Practice**

Day 5

- **Review Day 4**
- **ListView**
  - Adapter
  - View Holder
- **Async Tasks**
  - UI Thread
- **Practice**

Review day 2

- Resources
- Resources qualifiers
- Layouts
- Views
- Animation

Intent

An abstract operation used to start new activities, start and bind services, communicate with background services. Notice that this also allow apps to communicate between one another

- Explicit intents
- Implicit intents

Intent – Explicit Intents

Those are the intents that specify the component to be started by **name (the fully-qualified classs name)**. You'll typically use an explicit intent to start a component in your own app, or a specific component of another app.

Starting an activity using explicit intent

```
Intent intent = new Intent(getApplicationContext(), Activity.class);
startActivity(intent);
```

Intent - Extra

Key-value pairs that carry additional information required to accomplish the started activity.

Example:

```
Intent intent = new Intent(getApplicationContext(), WelcomeActivity.class);
intent.putExtra(WelcomeActivity.EXTRA_USERNAME, username.getText().toString());
startActivity(intent);
```

```
String username = getIntent().getStringExtra(EXTRA_USERNAME);
```

Practice:
Login form that opens another activity to welcome the user

Intent – Implicit Intents

Those are the intents that declare only a general action to perform, which allows a component from any app to handle it.

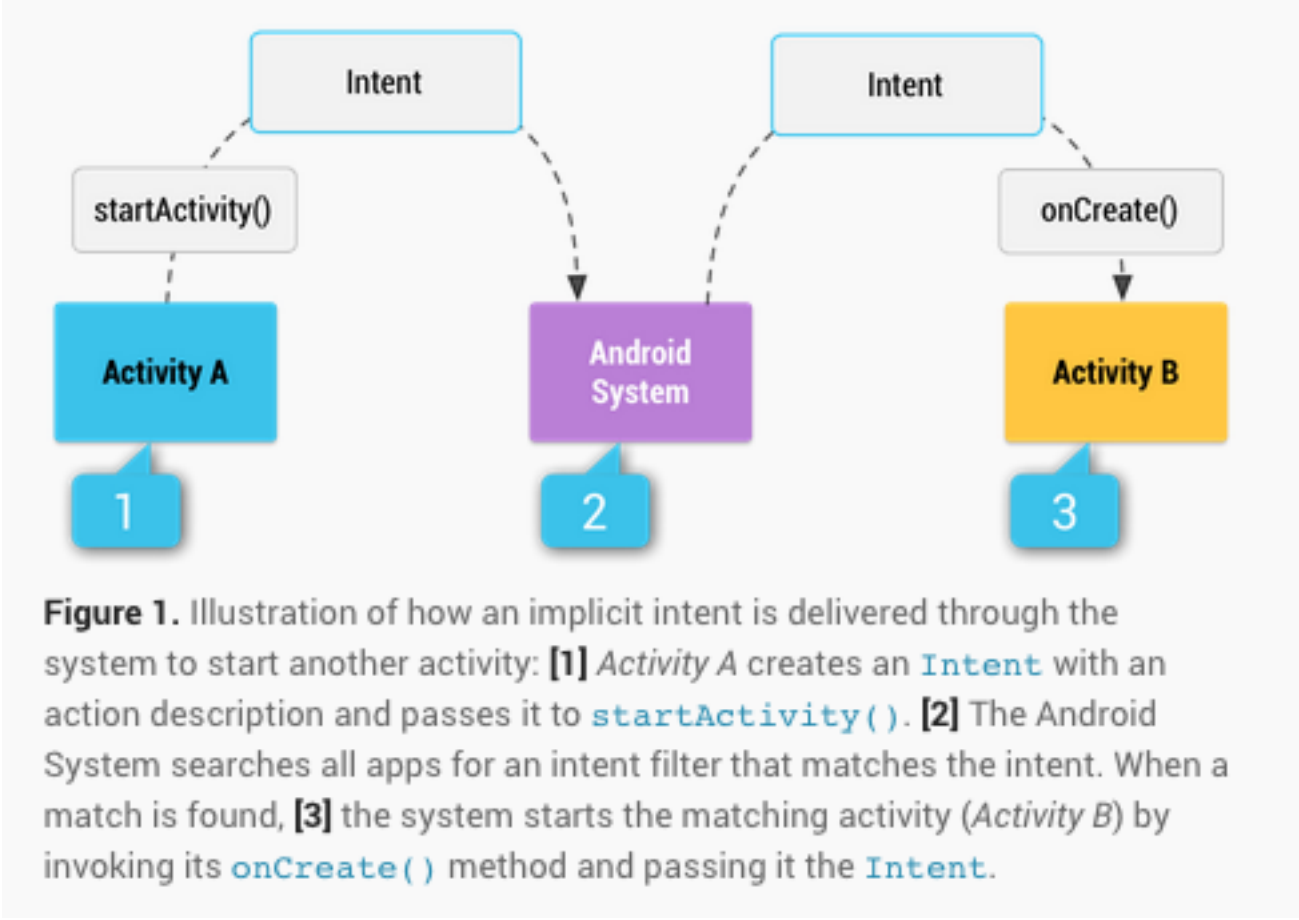


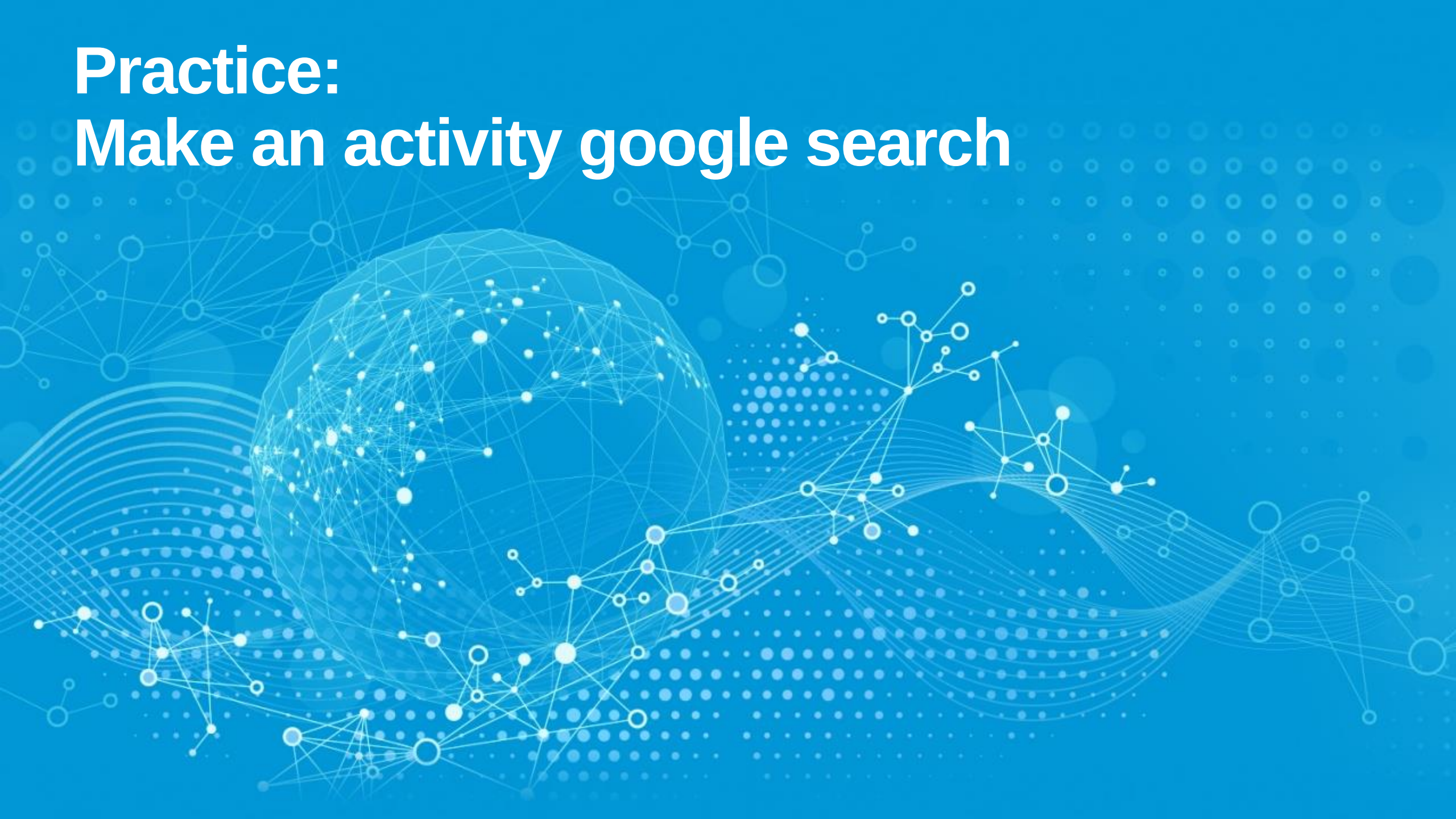
Figure 1. Illustration of how an implicit intent is delivered through the system to start another activity: **[1]** *Activity A* creates an `Intent` with an action description and passes it to `startActivity()`. **[2]** The Android System searches all apps for an intent filter that matches the intent. When a match is found, **[3]** the system starts the matching activity (*Activity B*) by invoking its `onCreate()` method and passing it the `Intent`.

Intent – Implicit Intents

Starting an implicit intent to perform the ACTION_VIEW of certain URL

```
String url = "http://www.google.com";
Intent i = new Intent(Intent.ACTION_VIEW);
i.setData(Uri.parse(url));
startActivity(i);
```

Practice:
Make an activity google search



Intent Filters – Receiving an Implicit Intent

To advertise which implicit intents your activity can receive, declare one or more intent filters for each of your app components with an <intent-filter> element in your manifest file.

```
<activity android:name="ShareActivity">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain"/>
    </intent-filter>
</activity>
```

*Multiple <action> <category> and <data> can be added to each intent filter

Practice:
Make an activity that receives the ACTION_VIEW intents

Android Manifest

- The manifest file presents essential information about your app to the Android system.
  - It **names the Java package for the application**.
  - It **describes the components** of the application — the **activities**, **services**, **broadcast receivers**, and content providers that the application is composed of.
  - It declares which **permissions** the application must have in order to access protected parts of the API and interact with other applications.
  - It **declares the minimum level of the Android API** that the application requires.

Add Activities

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
        android:name=".LoginActivity"
        android:label="@string/app_name" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity
        android:name=".WelcomeActivity"
        android:label="@string/title_activity_welcome" >
    </activity>
</application>
```

Add Permissions

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.fscAuthenticationIntegration.demo"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="10" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />
    <application
        android:name="com.fabasoft.android.application.FSCAuthenticationFramework"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".FSCAuthenticationIntegrationDemoActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

Permissions

- ACCESS_FINE_LOCATION
- ACCESS_NETWORK_STATE
- ACCESS_WIFI_STATE
- CALL_PHONE
- CAMERA
- FLASHLIGHT
- INTERNET
- READ_SMS
- SEND_SMS
- VIBRATE
- ...
- http://developer.android.com/reference/android/Manifest.permission.html

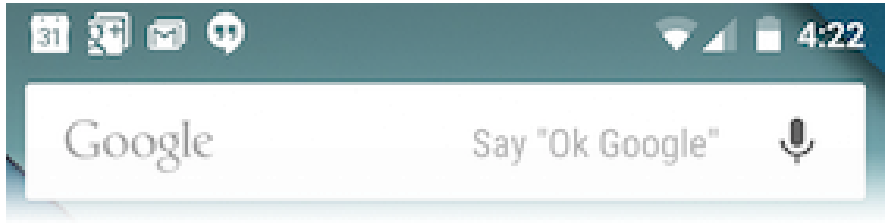
Application Icon

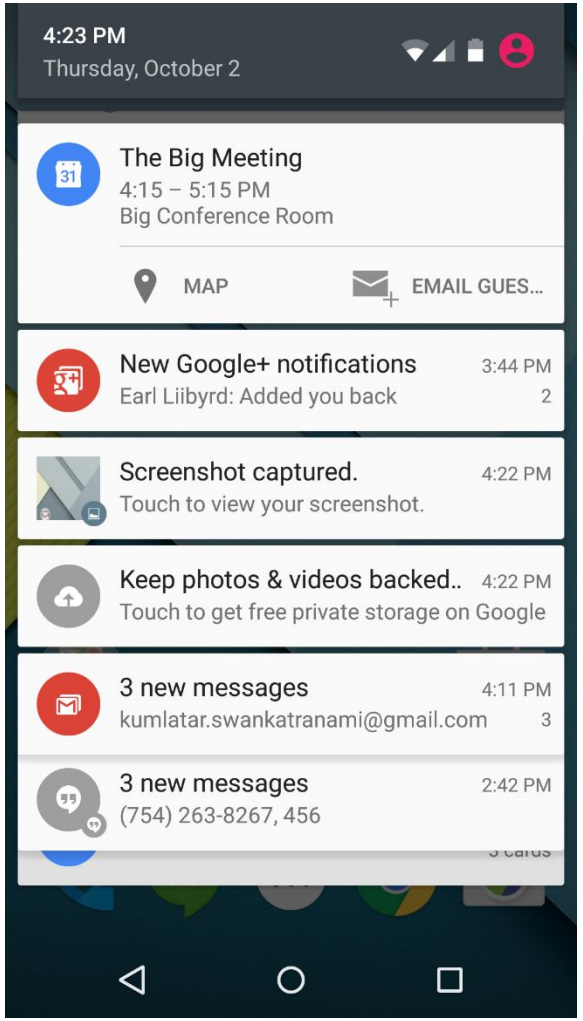
```
<application android:icon="@drawable/icon_name" android:label="@string/app_name" >
....
</application>
```

Notifications

A Notification object must contain the following:

- A small icon, set by setSmallIcon()
- A title, set by setContentTitle()
- Detail text, set by setContentText()

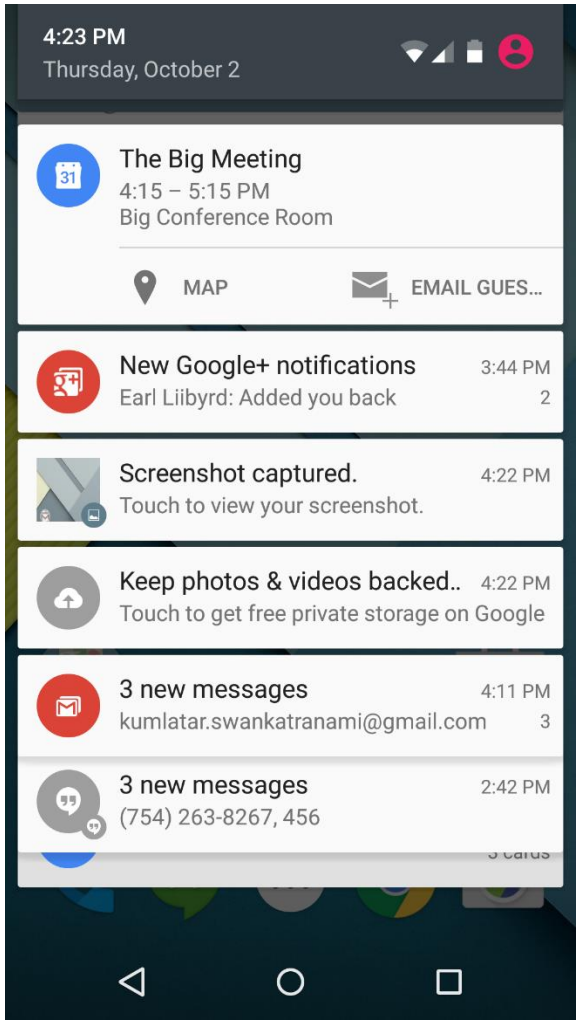




Notifications - Actions

Although they're optional, **you should add at least one action to your notification**. An action allows users to go directly from the notification to an Activity in your application, where they can look at one or more events or do further work.

The **action itself is defined by a PendingIntent containing an Intent** that starts an Activity in your application.



Notifications - Actions

```
NotificationCompat.Builder mBuilder =
        new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("My notification")
        .setContentText("Hello World!");
// Creates an explicit intent for an Activity in your app
Intent resultIntent = new Intent(this, ResultActivity.class);
```

```
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
// Adds the back stack for the Intent (but not the Intent itself)
stackBuilder.addParentStack(ResultActivity.class);
// Adds the Intent that starts the Activity to the top of the stack
stackBuilder.addNextIntent(resultIntent);
PendingIntent resultPendingIntent =
        stackBuilder.getPendingIntent(
            0,
            PendingIntent.FLAG_UPDATE_CURRENT
        );
mBuilder.setContentIntent(resultPendingIntent);
NotificationManager mNotificationManager =
    (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
// mId allows you to update the notification later on.
mNotificationManager.notify(mId, mBuilder.build());
```

Practice:
“Remind Me In” application

Thank you

Lucio Cossio
Luis Mazoni