

# *Around Android*

*Essential Android features illustrated by a walk through a practical example*

By Stefan Meisner Larsen, Trifork. [sml@trifork.dk](mailto:sml@trifork.dk). Twitter: stefanmeisner

# *Agenda*

- Introduction to *MoGuard Alert*
- Development Tools
- Creating the UI: Activity, layout and resources
- Gluing things together: Intent and Broadcast Receiver
- Writing a Service
- The Android Manifest
- Going to the Market



## *Features of MoGuard Alert*

- Start an alarm when a SMS message is received
  - Restrictions based on who send the message and message text
  - The volume is set to MAX level during the alarm, unless a headset is attached
  - Other sounds are muted during the alarm
- When the alarm sounds, the user is presented with a button to stop the alarm

## *Features of MoGuard Alert*

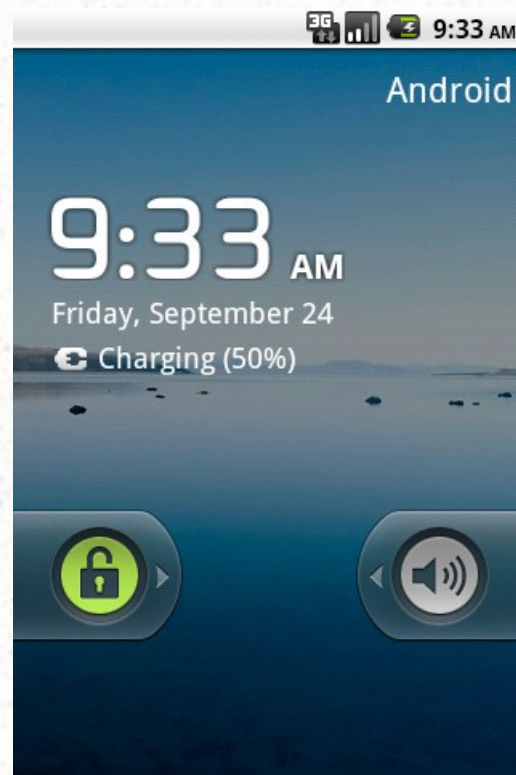
- Alarm Contacts' are the set of contacts which are allowed to start the alarm. The following options exists for each alarm contact
  - Alarm sound (Chosen between alarm sounds and ring tones)
  - Trigger text
  - Alternative Sender ID



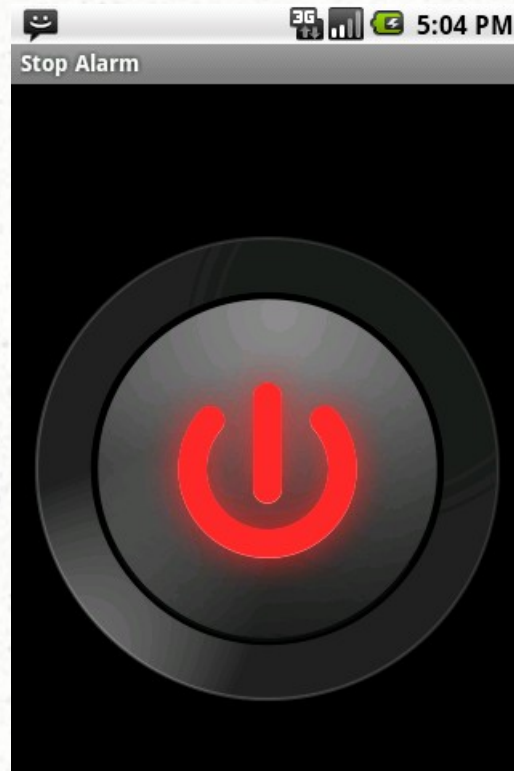
# *Features of MoGuard Alert*

- Send a predefined alarm (SMS) by pressing a button
- Settings
  - Default alarm sound
  - Duration of alarm
  - Alarm receiver

# *Incoming Alarm*

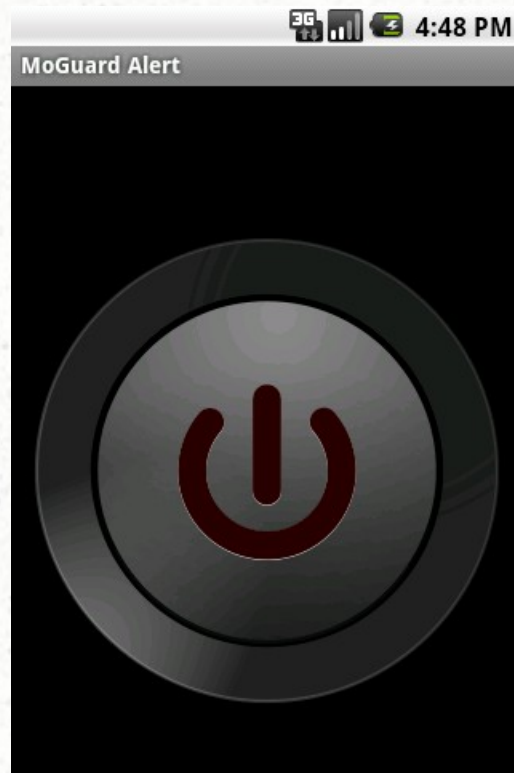


# *Incoming Alarm*



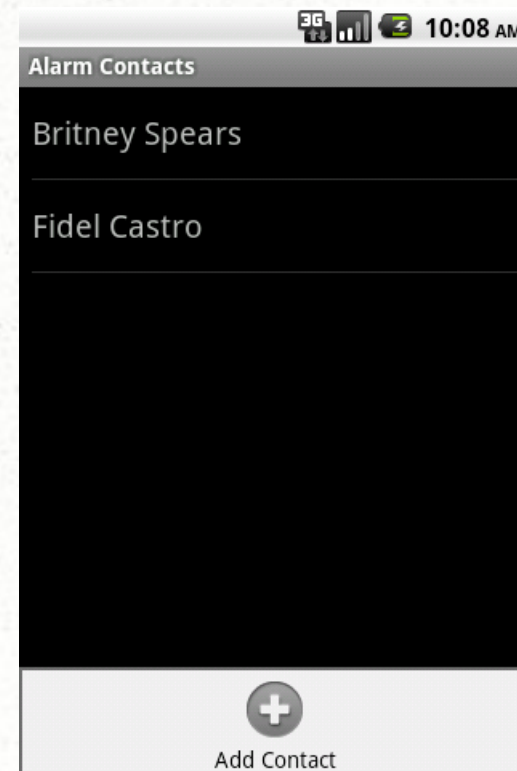
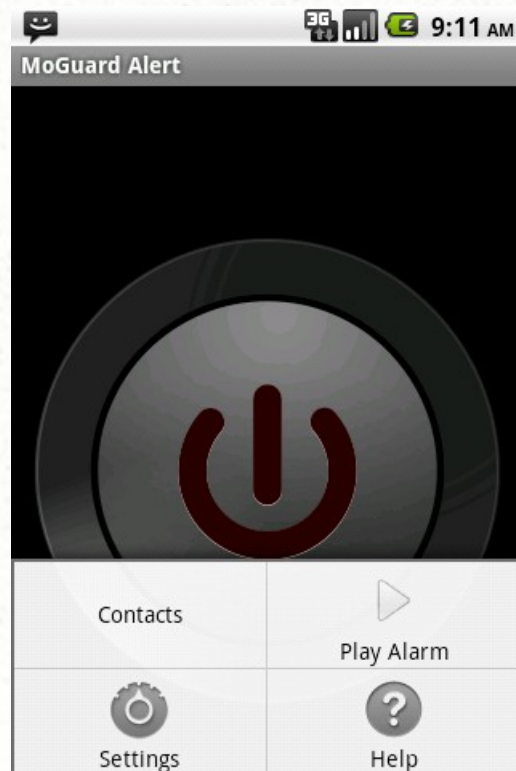


# *Incoming Alarm*

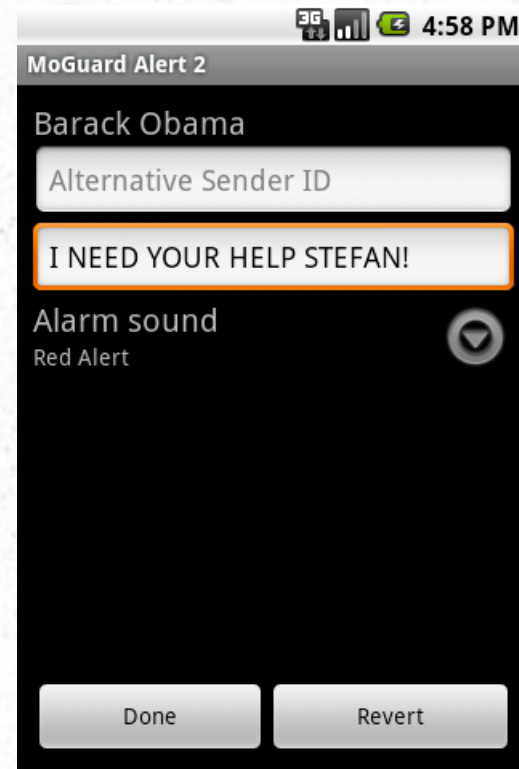
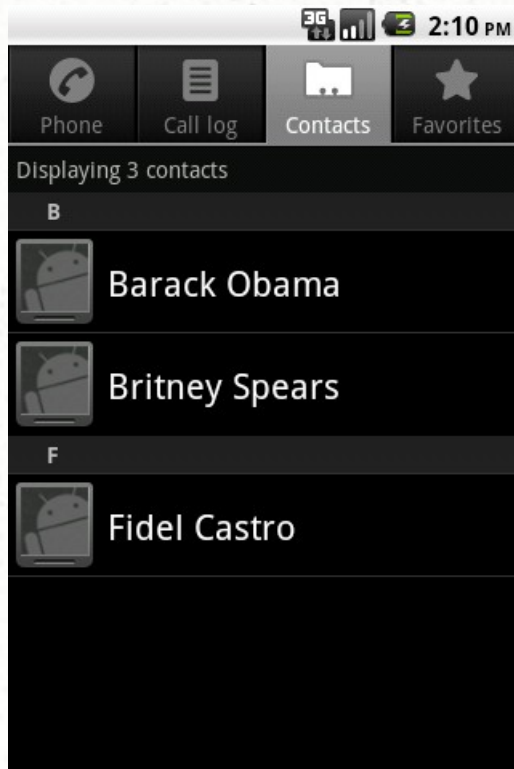




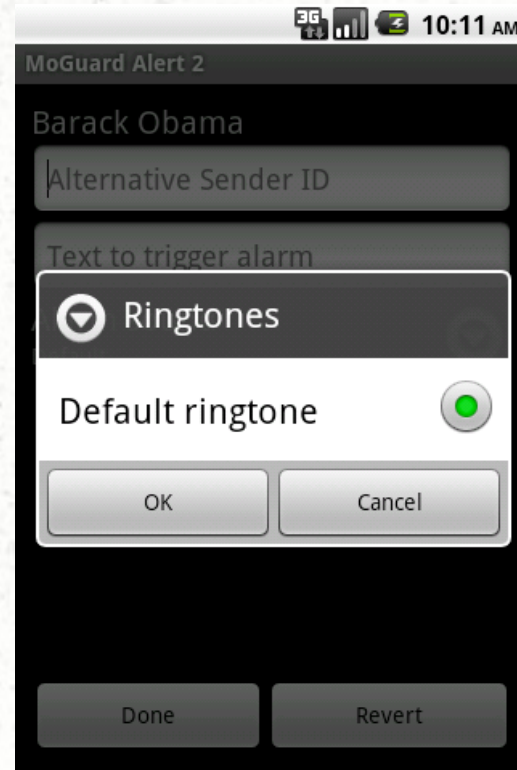
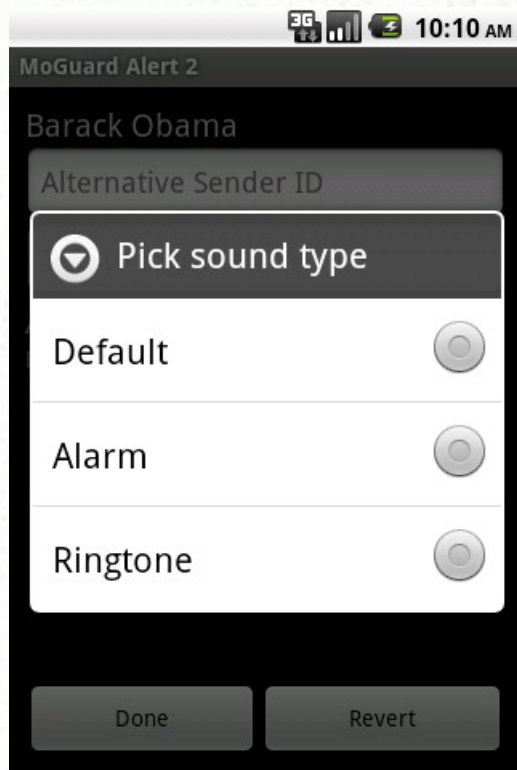
# *Adding a new Alarm Contact*



# *Alarm Contact options*



# *Selecting the sound*





# *Agenda*

- Introduction to *MoGuard Alert*
- **Development Tools**
- Creating the UI: Activity, layout and resources
- Gluing things together: Intent and Broadcast Receiver
- Writing a Service
- The Android Manifest
- Going to the Market

# *Android SDK essentials*

- Android libraries
- The SDK and AVD manager, for maintaining the SDK components and creating virtual devices
- ADB – Android Debug Bridge
- LogCat to capture logs from running device
- DDMS – Dalvik Debug Monitor
- Tools to create installable .apk files

# *SDK and AVD Manager*


Virtual Devices


Installed Packages


Available Packages


SDK Location: /home/stefan/trifork/android/android-sdk-linux\_86/


Installed Packages


 Android SDK Tools, revision 6


 Documentation for Android SDK, API 8, revision 1


 SDK Platform Android 2.2, API 8, revision 2


 SDK Platform Android 2.1-update1, API 7, revision 2


 SDK Platform Android 2.0.1, API 6, revision 1 (Obsolete)


 SDK Platform Android 2.0, API 5, revision 1 (Obsolete)


 SDK Platform Android 1.6, API 4, revision 3


 SDK Platform Android 1.5, API 3, revision 4


 Samples for SDK API 8, revision 1


 Samples for SDK API 7, revision 1


 Google APIs by Google Inc., Android API 8, revision 2

 Google APIs by Google Inc., Android API 7, revision 1

 Google APIs by Google Inc., Android API 6, revision 1 (Obsolete)

 Google APIs by Google Inc., Android API 5, revision 1 (Obsolete)

 Google APIs by Google Inc., Android API 4, revision 2

 Google APIs by Google Inc., Android API 3, revision 3

Description

Update All...

Delete...

Refresh



# Creating virtual devices

You can create virtual devices for different versions of Android, and different hardware configurations.

Virtual Devices

Installed Packages

Available Packages

List of existing Android Virtual Devices located at /home/stefan/.android/avd

AVD Name	Target Name	Platform	API Level
✓ Api4HVGA	Google APIs (Google Inc.)	1.6	4
✓ Api4QVGA	Google APIs (Google Inc.)	1.6	4
✓ Dev2.0	Android 2.0	2.0	5
✓ Dev2.1	Android 2.1-update1	2.1-updat	7
✓ MyHVGA	Android 2.1-update1	2.1-updat	7
✓ MaxDev	Android 2.2	2.2	8
✓ Buddy	Google APIs (Google Inc.)	2.2	8

✓ A valid Android Virtual Device.  A repairable Android Virtual Device.

✗ An Android Virtual Device that failed to load. Click 'Details' to see the error.

New...

Delete...

Repair...

Details...

Start...

Refresh

## *ADT – The Eclipse Plugin*

- Integrates the Android SDK with Eclipse
- Easy setup of new Android Projects
- Debugging application in emulator or on development device
- DDMS perspective

And the best: The choice is yours, you *can* do without it!

# *The DDMS perspective*

The screenshot displays the DDMS interface within an IDE, providing a comprehensive view of the Android emulator's internal state. The interface is organized into several panels:

- Devices:** A list of virtual devices. The selected device is `com.trifork.w` with PID 308 and memory address 8656 / 8700.
- Threads:** A table showing the current threads of the application. The threads are as follows:

ID	Tid	Status	utime	stime	Name
1	308	wait	778	280	main
*2	309	vmwait	17	20	HeapWorker
*3	310	vmwait	0	0	Signal Catcher
*4	311	running	29	28	JDWP
5	312	native	0	0	Binder Thread #1
6	313	native	0	0	Binder Thread #2
7	355	native	2	0	Binder Thread #3
- Heap:** A section for monitoring memory usage, including a **Refresh** button and a table with columns for Class, Method, File, Line, and Native.
- Allocation Tracker:** A tool for tracking memory allocations and deallocations.
- File Explorer:** A view of the file system within the emulator.
- Emulator Control:** A panel for controlling the emulator's behavior, including settings for Voice, Speed, Data, Latency, and Incoming number.
- LogCat:** A window for viewing the system log, showing messages from various components like `StartAlarmButton`, `dalvikvm`, and `Player`.

The status bar at the bottom indicates the current operation: `Launching MoGuardAler...ndroid 2.0)`.





## Package Explorer Hierarchy

MoGuardAlert2

- src
  - dk.moguardalert
  - dk.moguardalert.contacts
  - dk.moguardalert.service
  - dk.moguardalert.ui
- gen [Generated Java Files]
  - com.trifork.wakeup4
    - R.java
- Android 2.1-update1
  - android.jar - /home/stefan/trifork/android/an
  - assets
- doc
- res
  - drawable
    - drawable-hdpi
    - drawable-ldpi
    - drawable-mdpi
  - layout
  - menu
  - raw
  - values
  - xml
- AndroidManifest.xml
- default.properties
- todo.txt

Problems @ Javadoc Declaration

0 items

Description

Resource

# *Agenda*

- Introduction to *MoGuard Alert*
- Development Tools
- **Creating the UI: Activity, layout and resources**
- Gluing things together: Intent and Broadcast Receiver
- Writing a Service
- The Android Manifest
- Going to the Market

# *Creating the UI*

The layout of the user interface is declared in XML files.

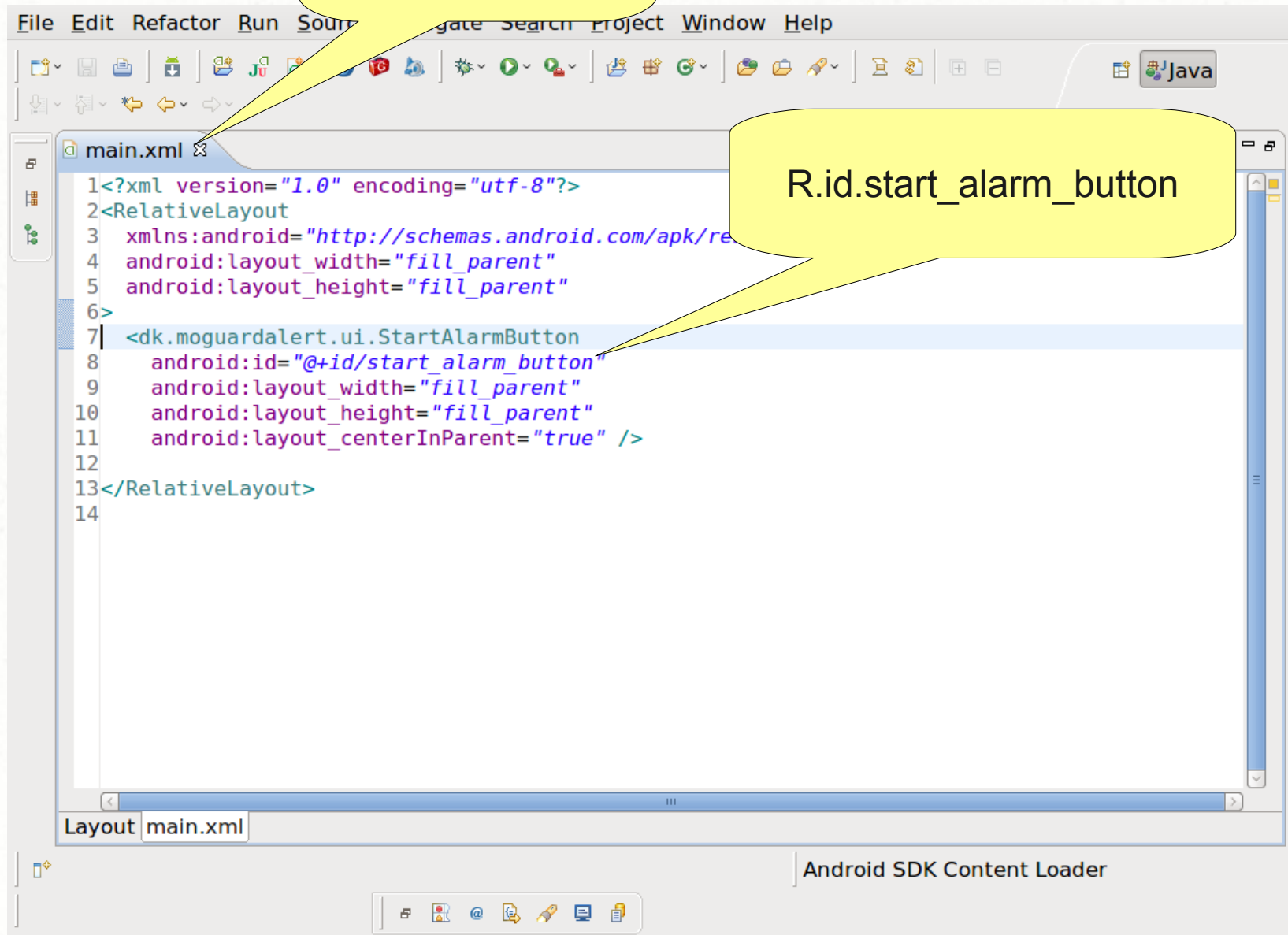
Widgets are referred to by tags with the same name as the class

Nice feature: Custom widgets is referred to as any other widget, by using the classname as an XML tag.



R.layout.main

R.id.start\_alarm\_button



File Edit Refactor Run Navigate Search Project Window Help



main.xml

Editing config: default

ADP2 Portrait Any loc No Doc Day tin Theme Create...

Layouts

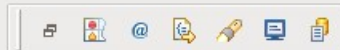
- Absolute...
- DialerFil...
- Expanda...
- FrameLa...
- GridView
- Horizont...
- ImageS...

Views

- Gesture...
- SurfaceV...
- View
- ViewStub
- AnalogCl...
- AutoCo...
- Button

A large, dark gray circular button with a red power symbol (a circle with a vertical line through it) in the center. The button has a slight 3D effect with a shadow.

Android SDK Content Loader



# *Activity*

- An Activity is used for interacting with the user.
- Activities has a well-defined life cycle

Lets take a look at MainActivity



# MainActivity

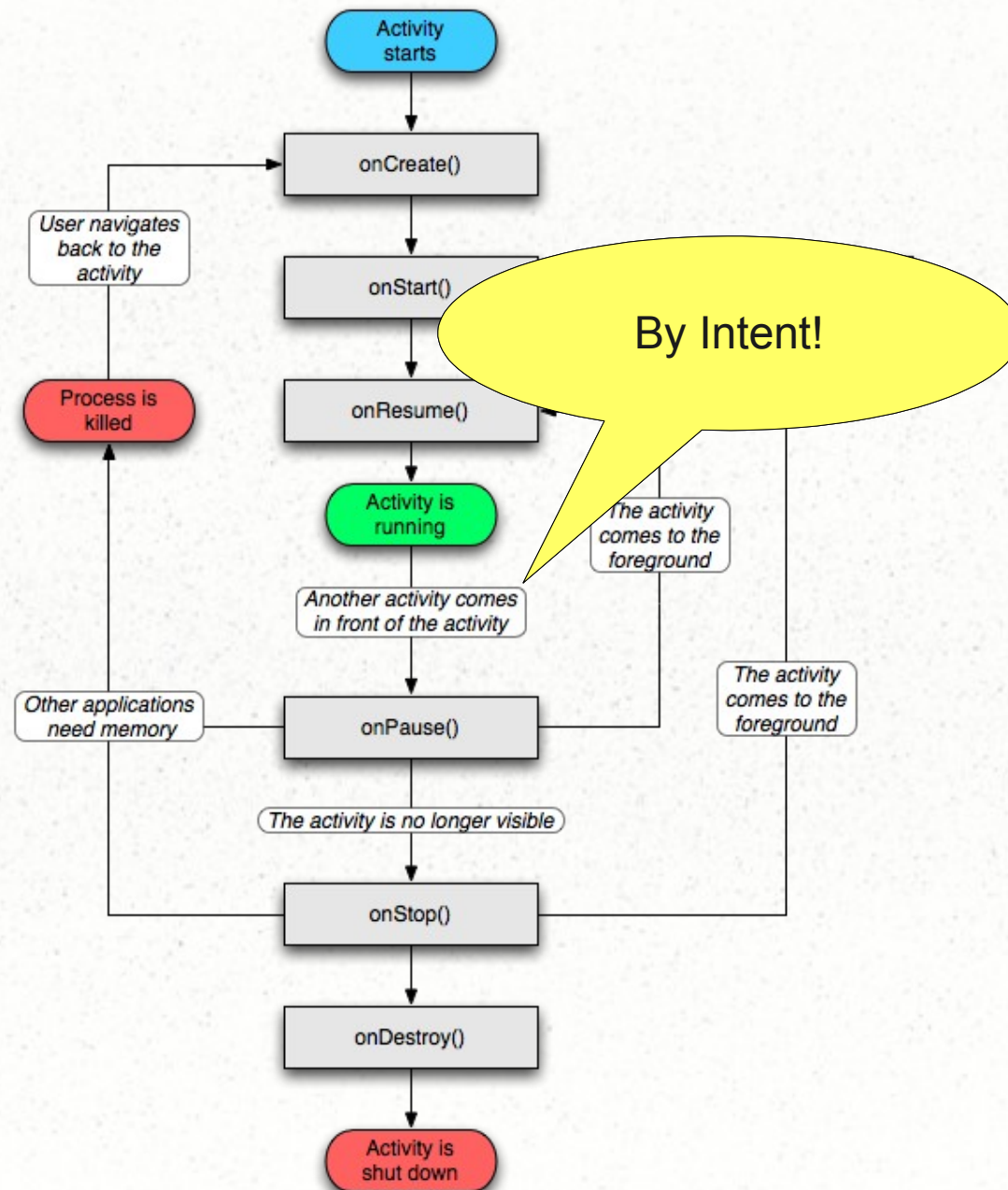
```
15
16 public class MainActivity extends AlarmActivity {
17     private StartAlarmButton startAlarmButton;
18     private boolean soundAlarmLocal;
19
20     @Override
21     public void onCreate(Bundle savedInstanceState) {
22         super.onCreate(savedInstanceState);
23
24         setContentView(R.layout.main);
25
26         startAlarmButton = (StartAlarmButton) findViewById(R.id.start_alarm_button);
27
28         startAlarmButton.setOnClickListener(new OnClickListener() {
29
30             @Override
31             public void onClick(View v) {
32                 Preferences preferences = new Preferences(MainActivity.this);
33                 if (soundAlarmLocal) {
34                     if (isAlarmOn()) {
35                         stopAlarm();
36                     } else {
37                         startAlarm();
38                     }
39                 } else {
40                     sendAlarm(preferences.getAlarmReceiver());
41                     startAlarmButton.setCheckedDelayed(false, 200);
42                 }
43             }
44         });
45     }
46 }
```

# MainActivity

```
52 @Override
53 public boolean onCreateOptionsMenu(Menu menu) {
54     MenuInflater inflater = getMenuInflater();
55     inflater.inflate(R.menu.app_menu, menu);
56     return true;
57 }
58
59 @Override
60 public boolean onOptionsItemSelected(MenuItem item) {
61     switch (item.getItemId()) {
62         case R.id.choose_controllers:
63             startActivity(new Intent(this, AlarmControllersActivity.class));
64             return true;
65
66         case R.id.show_preferences:
67             startActivity(new Intent(this, PreferenceActivity.class));
68             return true;
69
70         case R.id.start_alarm:
71             playCurrentAlarmSound();
72             return true;
73
74         case R.id.help:
75             startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(getString(R.string.help_url))));
76
77         default:
78             return super.onOptionsItemSelected(item);
79     }
80 }
81
82 @Override
83 protected void onResume() {
84     super.onResume();
85     Preferences preferences = new Preferences(MainActivity.this);
86     soundAlarmLocal = (preferences.getAlarmReceiver() == null || preferences.getAlarmReceiver().length() == 0);
87 }
```

# *The lifecycle of an Activity*





# *Agenda*

- Introduction to *MoGuard Alert*
- Development Tools
- Creating the UI: Activity, layout and resources
- Gluing things together: Intent and Broadcast Receiver
- Writing a Service
- The Android Manifest
- Going to the Market



# *Intents*

From the API description:

*An intent is an abstract description of an operation to be performed*

- Starting an Activity
- Sending a broadcast
- Starting a Service



# MainActivity

```
52 @Override
53 public boolean onCreateOptionsMenu(Menu menu) {
54     MenuInflater inflater = getMenuInflater();
55     inflater.inflate(R.menu.app_menu, menu);
56     return true;
57 }
58
59 @Override
60 public boolean onOptionsItemSelected(MenuItem item) {
61     switch (item.getItemId()) {
62         case R.id.choose_controllers:
63             startActivity(new Intent(this, AlarmControllersActivity.class));
64             return true;
65
66         case R.id.show_preferences:
67             startActivity(new Intent(this, PreferenceActivity.class));
68             return true;
69
70         case R.id.start_alarm:
71             playCurrentAlarmSound();
72             return true;
73
74         case R.id.help:
75             startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(getString(R.string.help_url))));
76
77         default:
78             return super.onOptionsItemSelected(item);
79     }
80 }
81
82 @Override
83 protected void onResume() {
84     super.onResume();
85     Preferences preferences = new Preferences(MainActivity.this);
86     soundAlarmLocal = (preferences.getAlarmReceiver() == null || preferences.getAlarmReceiver().length() == 0);
87 }
```

# *Using Intent to start the AlarmService*

```
protected void startAlarm() {  
    Log.d(TAG, "Starting alarm service");  
    Intent intent = new Intent(this, AlarmService.class);  
    intent.putExtra(AlarmService.EXTRA_SEND_STARTED_BROADCAST, false);  
    startService(intent);  
    alarmOn = true;  
}
```

# *Sending a broadcast Intent*

```
protected void stopAlarm() {  
    Log.d(TAG, "Stopping alarm by sending broadcast ALARM_STOP_REQUEST_ACTION");  
    sendBroadcast(new Intent(AlarmService.ALARM_STOP_REQUEST_ACTION));  
}
```



# *Listening for broadcasts*

```
public class SmsBroadcastReceiver extends BroadcastReceiver {

    private static final String TAG = SmsBroadcastReceiver.class.getSimpleName();

    @Override
    public void onReceive(Context context, Intent intent) {
        Log.d(TAG, "SMS received");
        final ContactInfoRepository contactInfoRepository = new ContactInfoRepositoryImpl(context);

        ContactInfo contactInfo = null;
        String originator = null;
        for (SmsMessage message : getSmsMessages(intent)) {
            Log.i(TAG, "Received SMS from " + message.getOriginatingAddress());
            originator = message.getOriginatingAddress();
            if (originator != null) {
                contactInfo = contactInfoRepository.lookupByPhoneNumber(context, originator);
                if (contactInfo != null && isAcceptableTriggerText(contactInfo, message)) {
                    playAlarmTone(context, contactInfo);
                    Toast.makeText(context, context.getResources().getString(R.string.msg_alarm_from) + " " + originator, Toast.LENGTH_SHORT).show();
                    break;
                } else {
                    Log.w(TAG, "No information on " + originator + " found");
                }
            }
        }
    }
}
```

# *Agenda*

- Introduction to *MoGuard Alert*
- Development Tools
- Creating the UI: Activity, layout and resources
- Gluing things together: Intent and Broadcast Receiver
- **Writing a Service**
- The Android Manifest
- Going to the Market

## *A Service is...*

- A way of telling Android that this piece of code should run even when the application is in the background (...please don't kill me...!)
- Running on the main thread
  - But feel free to start a new thread!



## *A Service can...*

- Expose a remote interface for other applications to use.
- The interface definition is written in AIDL – Android Interface Definition Language, which is translated into a Java interface, that your service will implement.

# *The remote interface to MoGuard Alert*

- TBD

## *The AlarmService*

- The service is started by calling `startService(Intent intent)`
- Communication with the service is based solely on Intents



```

public class AlarmService extends Service {

    public static final String TAG = "AlarmService";
    public static final String EXTRA_DURATION = "duration";
    public static final String EXTRA_MAX_VOLUME = "maxvolume";
    public static final String EXTRA_SEND_STOPPED_BROADCAST = "sendstoppedbc";
    public static final String EXTRA_RINGTONE = "ringtone";
    public static final String EXTRA_SOUND_RESOURCE = "soundresource";

    public static final String ALARM_STARTED_ACTION = "dk.moguardalert.intent.action.STARTED";
    public static final String ALARM_STOPPED_ACTION = "dk.moguardalert.intent.action.STOPPED";
    public static final String ALARM_STOP_REQUEST_ACTION = "dk.moguardalert.intent.action.STOP_REQUEST";
    private Player player;
    private BroadcastReceiver stopAlarmReceiver;
    private AlarmStopListener alarmStopListener;

    @Override
    public void onCreate() {
        super.onCreate();

        alarmStopListener = new AlarmStopListener() {

            @Override
            public void onAlarmStop() {
                Log.d(TAG, "Player stopped");
                sendBroadcast(new Intent(ALARM_STOPPED_ACTION));
            }
        };

        stopAlarmReceiver = new BroadcastReceiver() {

            @Override
            public void onReceive(Context context, Intent intent) {
                Log.d(TAG, "Stopping player, requested by broadcast ALARM_STOP_REQUEST_ACTION");
                player.stop();
            }
        };
    }
}

```

@Override

**public int** onStartCommand(Intent intent, **int** flags, **int** startId) {

```
    if (player != null && !player.isStopped()) {  
        Log.w(TAG, "Ignoring Alarm, player already running");  
        return START_NOT_STICKY;  
    }
```

```
    Log.d(TAG, "Started");
```

```
    Preferences preferences = new Preferences(this);
```

```
    int duration = intent.getIntExtra(EXTRA_DURATION, preferences.getAlarmDuration());
```

```
    boolean playWithMaxVolume = intent.getBooleanExtra(EXTRA_MAX_VOLUME, true);
```

```
    boolean sendBroadcastWhenAlarmStarted = intent.getBooleanExtra(EXTRA_SEND_STARTED_BROADCAST, true);
```

```
    Uri ringTone = intent.getParcelableExtra(EXTRA_RINGTONE);
```

```
    int resourceId = intent.getIntExtra(EXTRA_SOUND_RESOURCE, -1);
```

```
    if (ringTone == null) {  
        player = new Player(this, preferences.getAlarmResource(resourceId), duration, playWithMaxVolume, alarmStopListener);  
    } else {  
        player = new Player(this, ringTone, duration, playWithMaxVolume, alarmStopListener);  
    }
```

```
    if (sendBroadcastWhenAlarmStarted) {
```

```
        Log.d(TAG, "Sending ALARM_STARTED_ACTION broadcast");
```

```
        sendBroadcast(new Intent(ALARM_STARTED_ACTION));
```

```
    }
```

```
    new Thread(player).start();
```

```
    registerReceiver(stopAlarmReceiver, new IntentFilter(dk.moguardalert.service.AlarmService.ALARM_STOP_REQUEST_ACTION));
```

```
    return START_NOT_STICKY;
```

```
}
```

@Override

**public void** onDestroy() {

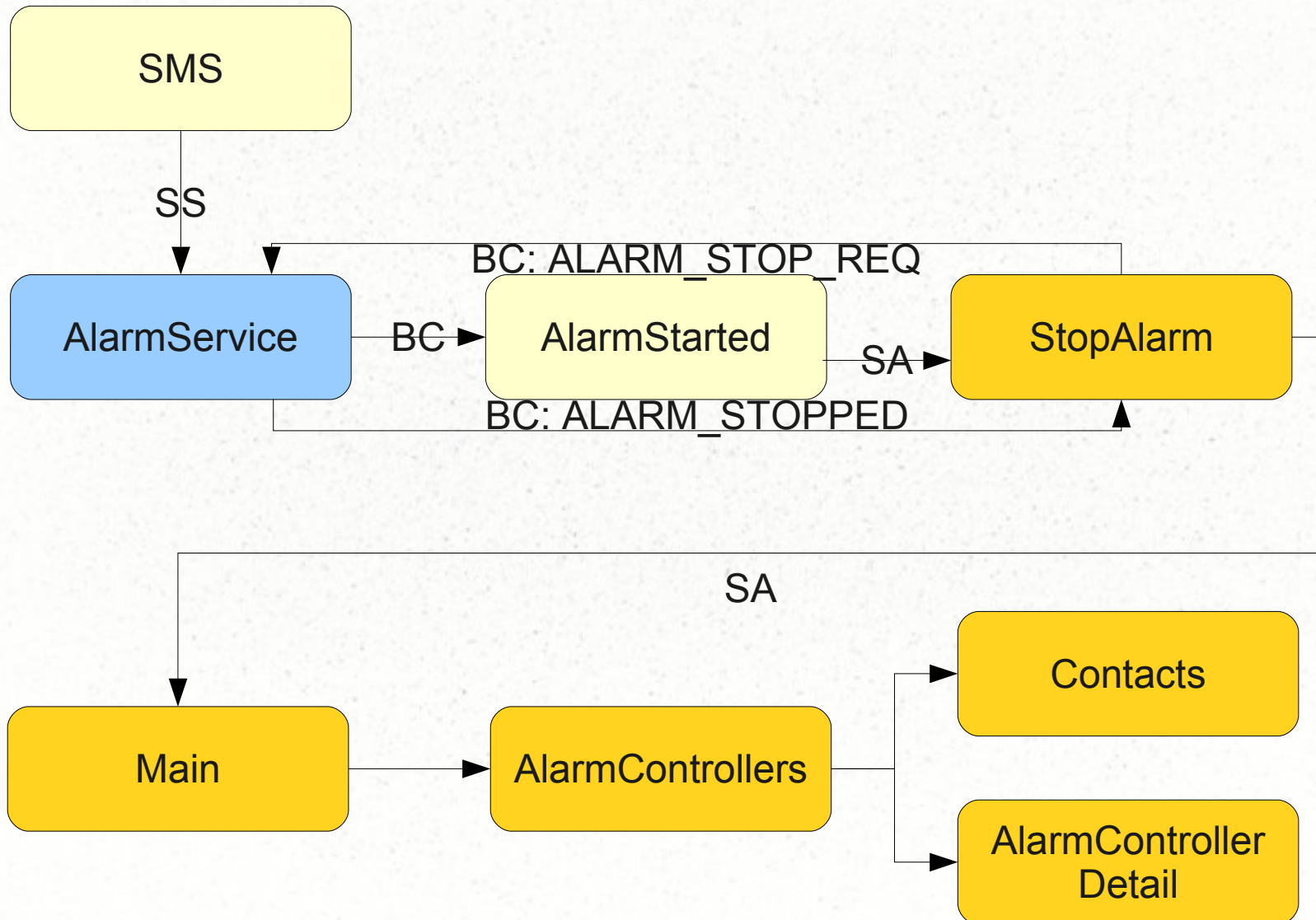
```
    Log.d(TAG, "Destroyed");
```

```
    unregisterReceiver(stopAlarmReceiver);
```

```
    player.stop();
```

```
}
```

# *Intent flow in MoGuard Alert*





# *Agenda*

- Introduction to *MoGuard Alert*
- Development Tools
- Creating the UI: Activity, layout and resources
- Gluing things together: Intent and Broadcast Receiver
- Writing a Service
- **The Android Manifest**
- Going to the Market

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.trifork.wakeup4"
    android:versionCode="13" android:versionName="2.2">
    <application android:icon="@drawable/icon" android:label="@string/app_name">

        <activity android:name="dk.moguardalert.ui.MainActivity" android:label="MoGuard Alert">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name="dk.moguardalert.ui.AlarmControllersActivity" android:label="Alarm Contacts"/>
        <activity android:name="dk.moguardalert.ui.AlarmControllerDetailsActivity" />
        <activity android:name="dk.moguardalert.ui.StopAlarmActivity" android:label="Stop Alarm"
            android:launchMode="singleTop"/>
        <activity android:name="dk.moguardalert.ui.PreferenceActivity" android:label="Settings" />

        <receiver android:name="dk.moguardalert.AlarmStartedReceiver">
            <intent-filter>
                <action android:name="dk.moguardalert.intent.action.STARTED" />
            </intent-filter>
        </receiver>

        <receiver android:name="dk.moguardalert.SmsBroadcastReceiver">
            <intent-filter>
                <action android:name="android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
        </receiver>

        <service android:label="AlarmService" android:name="dk.moguardalert.service.AlarmService" />

    </application>
    <uses-sdk android:minSdkVersion="7" />

    <uses-permission android:name="android.permission.RECEIVE_SMS"></uses-permission>
    <uses-permission android:name="android.permission.SEND_SMS"></uses-permission>
    <uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"></uses-permission>

</manifest>

```



# *Agenda*

- Introduction to *MoGuard Alert*
- Development Tools
- Creating the UI: Activity, layout and resources
- Gluing things together: Intent and Broadcast Receiver
- Writing a Service
- The Android Manifest
- Going to the Market



# *Developer Console*

[Link to Developer Console](#)

# *A mail conversation with Adam...*

Adam:

This app erased ALL my phone data. Thanks for ruining my life!

# *A mail conversation with Adam...*

Stefan:

Hi Adam

Sorry to hear about your problems.

But my app can in no way erase your phone data,  
so you must have some other troubles with your  
phone.

Regards,

Stefan Meisner Larsen



# *A mail conversation with Adam...*

Adam:

I installed the app.

Opened it from the task menu and my phone locked up.

I had to remove the battery off my Droid and when I turned It on, everything was gone.

I've had this phone 4 days and I have 10 apps...

**ironic that your app us the last thing my phone was doing before it crashed.**

# *A mail conversation with Adam*

Hi Adam,

Well, this app has been installed on > 1000 phones and is running on > 250 phones.

The application simply doesn't have the rights to erase your phone data, so even if it tried it wouldn't succeed due to the security model implemented in Android. You might have a hardware problem which was somehow triggered by MoGuard Alert.

Regards,  
Stefan Meisner Larsen

