Around Android

Essential Android features illustrated by a walk through a practical example

By Stefan Meisner Larsen, Trifork. 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 exist 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

Displaying 3 contacts

- Barack Obama
- Britney Spears
- Fidel Castro

MoGuard Alert 2

Barack Obama

Alternative Sender ID

I NEED YOUR HELP STEFAN!

Alarm sound

Red Alert

Done
Revert
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**

<table>
<thead>
<tr>
<th>Virtual Devices</th>
<th>SDK Location: /home/stefan/trifork/android/android-sdk-linux_86/</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Installed Packages</strong></td>
</tr>
<tr>
<td></td>
<td>X Android SDK Tools, revision 6</td>
</tr>
<tr>
<td></td>
<td>• Documentation for Android SDK, API 8, revision 1</td>
</tr>
<tr>
<td></td>
<td>• SDK Platform Android 2.2, API 8, revision 2</td>
</tr>
<tr>
<td></td>
<td>• SDK Platform Android 2.1-update1, API 7, revision 2</td>
</tr>
<tr>
<td></td>
<td>• SDK Platform Android 2.0.1, API 6, revision 1 (Obsolete)</td>
</tr>
<tr>
<td></td>
<td>• SDK Platform Android 2.0, API 5, revision 1 (Obsolete)</td>
</tr>
<tr>
<td></td>
<td>• SDK Platform Android 1.6, API 4, revision 3</td>
</tr>
<tr>
<td></td>
<td>• SDK Platform Android 1.5, API 3, revision 4</td>
</tr>
<tr>
<td></td>
<td>• Samples for SDK API 8, revision 1</td>
</tr>
<tr>
<td></td>
<td>• Samples for SDK API 7, revision 1</td>
</tr>
<tr>
<td></td>
<td>• Google APIs by Google Inc., Android API 8, revision 2</td>
</tr>
<tr>
<td></td>
<td>• Google APIs by Google Inc., Android API 7, revision 1</td>
</tr>
<tr>
<td></td>
<td>• Google APIs by Google Inc., Android API 6, revision 1 (Obsolete)</td>
</tr>
<tr>
<td></td>
<td>• Google APIs by Google Inc., Android API 5, revision 1 (Obsolete)</td>
</tr>
<tr>
<td></td>
<td>• Google APIs by Google Inc., Android API 4, revision 2</td>
</tr>
<tr>
<td></td>
<td>• Google APIs by Google Inc., Android API 3, revision 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Update All...</th>
<th>Delete...</th>
<th>Refresh</th>
</tr>
</thead>
</table>
Creating virtual devices

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

<table>
<thead>
<tr>
<th>AVD Name</th>
<th>Target Name</th>
<th>Platform</th>
<th>API Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Api4HVGA</td>
<td>Google APIs (Google Inc.)</td>
<td>1.6</td>
<td>4</td>
</tr>
<tr>
<td>Api4QVGA</td>
<td>Google APIs (Google Inc.)</td>
<td>1.6</td>
<td>4</td>
</tr>
<tr>
<td>Dev2.0</td>
<td>Android 2.0</td>
<td>2.0</td>
<td>5</td>
</tr>
<tr>
<td>Dev2.1</td>
<td>Android 2.1-update1</td>
<td>2.1-update 7</td>
<td>7</td>
</tr>
<tr>
<td>MyHVGA</td>
<td>Android 2.1-update1</td>
<td>2.1-update 7</td>
<td>7</td>
</tr>
<tr>
<td>MaxDev</td>
<td>Android 2.2</td>
<td>2.2</td>
<td>8</td>
</tr>
<tr>
<td>Buddy</td>
<td>Google APIs (Google Inc.)</td>
<td>2.2</td>
<td>8</td>
</tr>
</tbody>
</table>

- ✓ A valid Android Virtual Device.  🗿 A repairable Android Virtual Device.  ❌ An Android Virtual Device that failed to load. Click 'Details' to see the error.
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
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
Activity

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

Let's take a look at MainActivity
```java
public class MainActivity extends AppCompatActivity {
    private StartAlarmButton startAlarmButton;
    private boolean soundAlarmLocal;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        startAlarmButton = (StartAlarmButton) findViewById(R.id.start_alarm_button);

        startAlarmButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                Preferences preferences = new Preferences(MainActivity.this);
                if (soundAlarmLocal) {
                    if (isAlarmOn()) {
                        stopAlarm();
                    } else {
                        startAlarm();
                    }
                } else {
                    sendAlarm(preferences.getAlarmReceiver());
                    startAlarmButton.setCheckedDelayed(false, 200);
                }
            }
        });
    }
}```
```
@override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.app_menu, menu);
    return true;
}

@override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.choose_controllers:
        startActivity(new Intent(this, AlarmControllersActivity.class));
        return true;

    case R.id.show_preferences:
        startActivity(new Intent(this, PreferenceActivity.class));
        return true;

    case R.id.start_alarm:
        playCurrentAlarmSound();
        return true;

    case R.id.help:
        startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(getString(R.string.help_url))));
        break;

    default:
        return super.onOptionsItemSelected(item);
    }
}

@Override
protected void onResume() {
    super.onResume();
    Preferences preferences = new Preferences(MainActivity.this);
    soundAlarmLocal = (preferences.getAlarmReceiver() == null || preferences.getAlarmReceiver().length() == 0);
}
```
The lifecycle of an Activity
But how did that happen?

By Intent!
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
```java
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.app_menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.choose_controllers:
        startActivity(new Intent(this, AlarmControllersActivity.class));
        return true;

    case R.id.show_preferences:
        startActivity(new Intent(this, PreferenceActivity.class));
        return true;

    case R.id.start_alarm:
        playCurrentAlarmSound();
        return true;

    case R.id.help:
        startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(getString(R.string.help_url))));
        break;

    default:
        return super.onOptionsItemSelected(item);
    }
}

@Override
protected void onResume() {
    super.onResume();
    Preferences preferences = new Preferences(MainActivity.this);
    soundAlarmLocal = (preferences.getAlarmReceiver() == null || preferences.getAlarmReceiver().length() == 0);
}
Using Intent to start the AlarmService

```java
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

```java
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)) {
                    playAlertTone(context, contactInfo);
                    Toast.makeText(context, context.getResources().getString(R.string.msg_alert_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_STARTED_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.ARM_STOP_REQUEST_ACTION));
    return START_NOT_STICKY;
}

@Override
public void onDestroy() {
    Log.d(TAG, "Destroyed");
    unregisterReceiver(stopAlarmReceiver);
    player.stop();
}
Intent flow in MoGuard Alert

SMS

AlarmService

SS

BC: ALARM_STOP_REQ

AlarmStarted

StopAlarm

BC: ALARM_STOPPED

Main

AlarmControllers

Contacts

AlarmController Detail

SA
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: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 android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
</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 was the last thing my phone was doing before it crashed.
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