Android Protips: Advanced Topics for Expert Android App Developers

Reto Meier
May 10, 2011

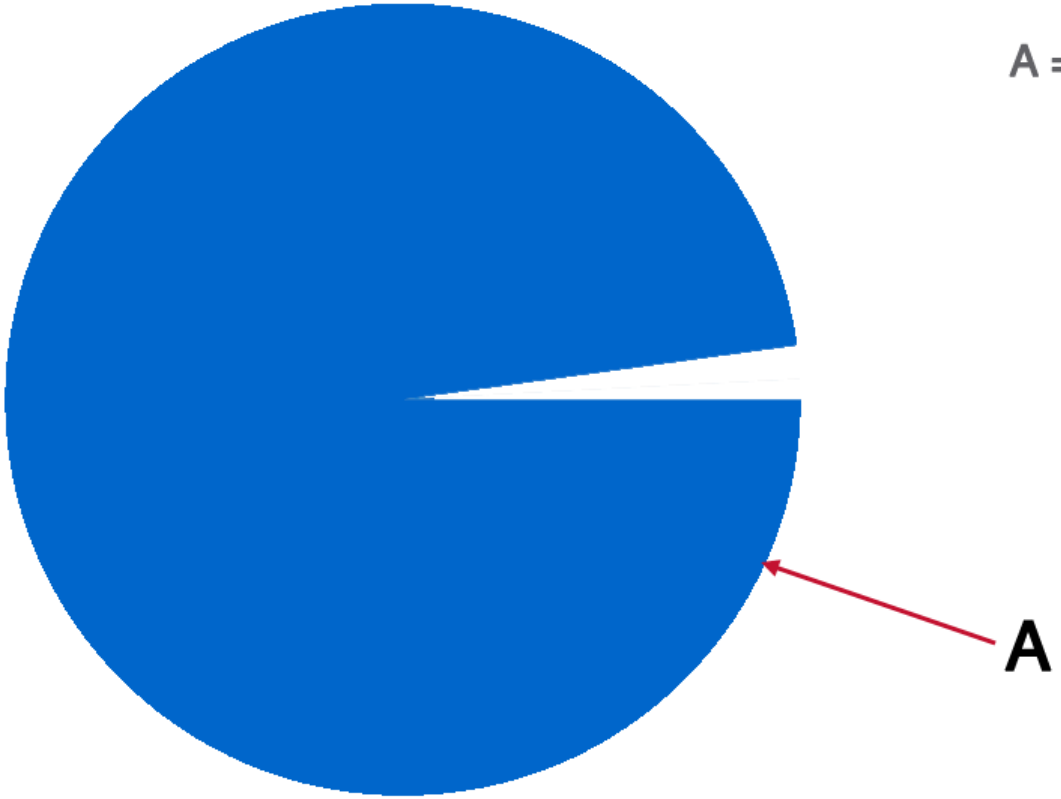
Twitter: @retomeier
This presentation at Google I/O 2011 on YouTube:
http://goo.gl/H5nFe

Google I/O 11

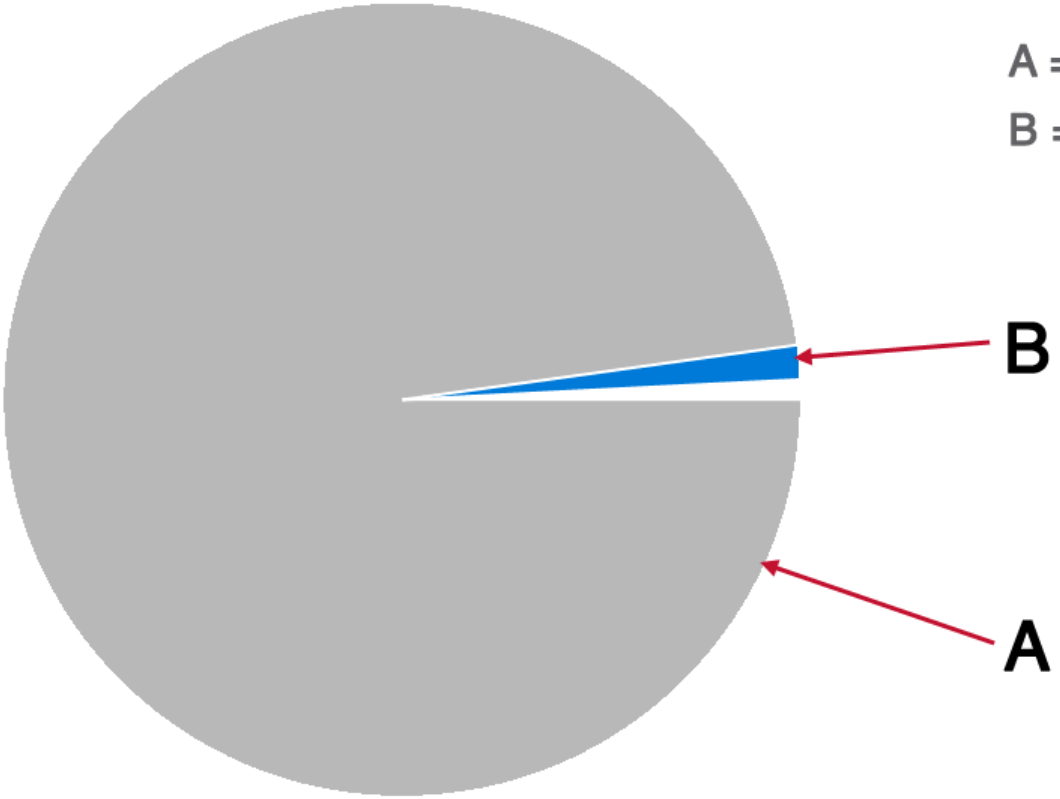
“How do I get featured?”

Every Android app developer who isn’t featured

A = Want to be featured



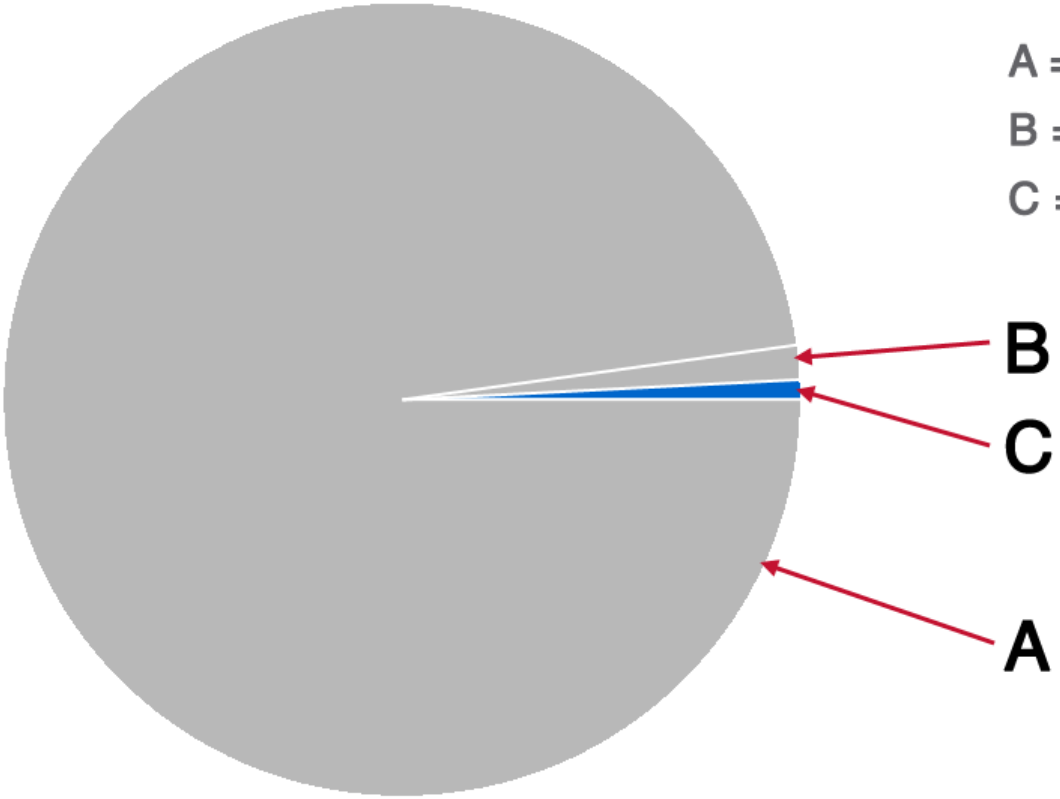
All Apps in Android Market



A = Want to be featured

B = Don't know about featuring

All Apps in Android Market

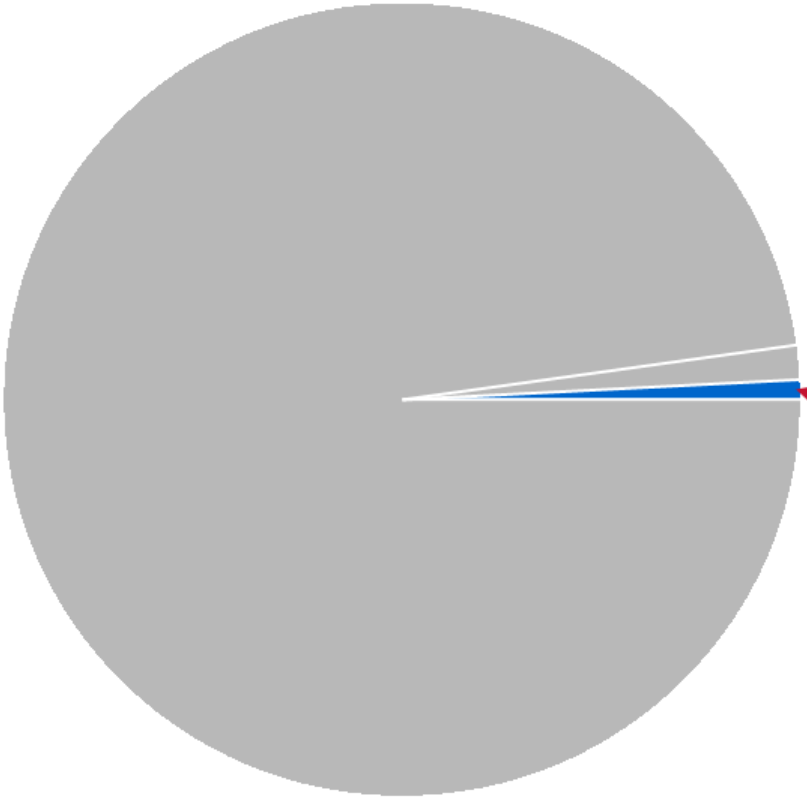


All Apps in Android Market

A = Want to be featured

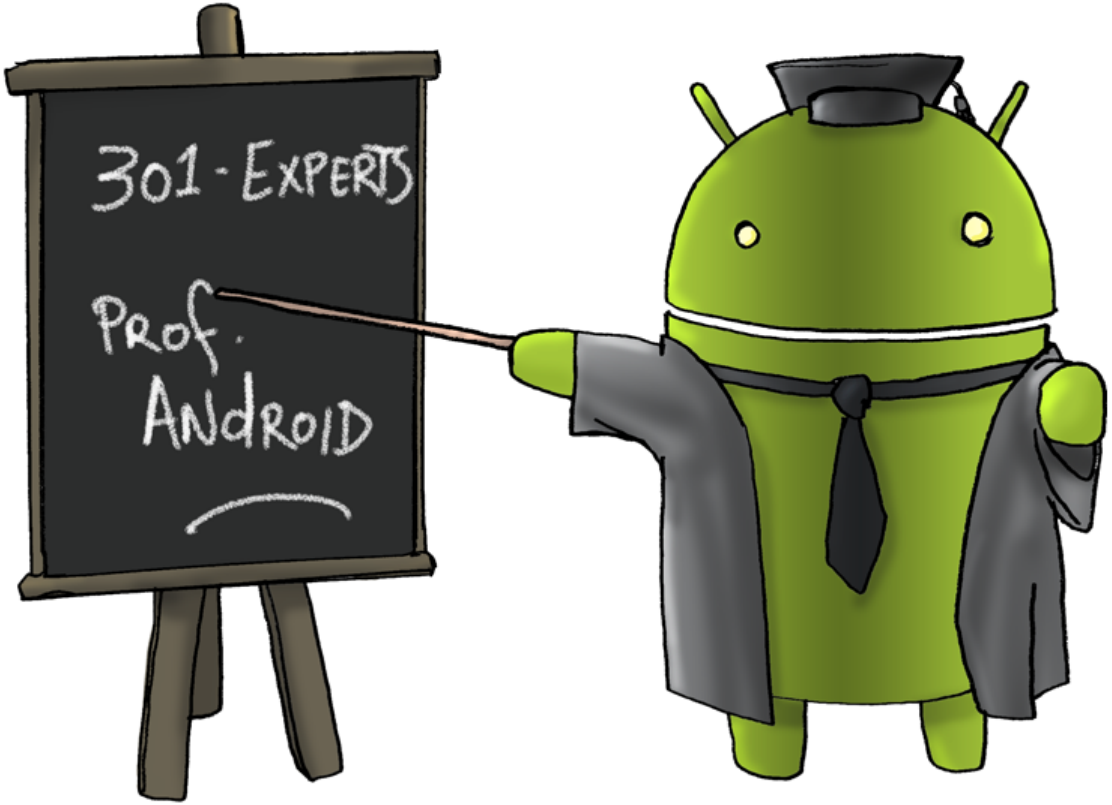
B = Don't know about featuring

C = Featured



C = Featured

All Apps in Android Market


301 - EXPERTS
Prof.
ANDROID

Create apps people want

- Gmail
- Google Maps
- News (CNN)
- Shopping (Ocado, Amazon, eBay)

Create apps people don’t know they want

- Shazam
- Google Goggles
- Google Body
- Angry Birds

Use and support the latest APIs and hardware

- Even if your Build Target / Target SDK is earlier.
- SDKs are backwards compatible.
- New APIs often optimize efficiency.
- Easy to detect the platform version at runtime.
- Use Interfaces or parallel Activities for backwards compatibility.

The parallel Activity pattern

```
private static boolean shinyNewAPIS =
android.os.Build.VERSION.SDK_INT >=
  android.os.Build.VERSION_CODES.HONEYCOMB;

@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  Intent startActivityIntent = null;
  if (!shinyNewAPIS)
    startActivityIntent = new Intent(this, legacyActivity.class);
  else
    startActivityIntent = new Intent(this, hcActivity.class);

  startActivity(startActivityIntent);
  finish();
}
```

The parallel Activity pattern

- Use the same layouts and Fragments.
- Encapsulate functionality within Fragments.
- Use Activity to handle animations and Action Bar.
- Use in combination with res folders (size / version / orientation)

```
res/layout-port
res/layout-land
res/layout-xlarge-port-v11
res/layout-xlarge-land
```

Use Interfaces for backwards compatibility

- Use the best hardware available.
- Use the most efficient software API.

Use Interfaces for backwards compatibility

```
private static boolean newSensorAPIsSupported =
  Build.VERSION.SDK_INT >= Build.VERSION_CODES.CUPCAKE;

boolean gyroExists =
  getPackageManager().hasSystemFeature(
  PackageManager.FEATURE_SENSOR_GYROSCOPE);

IOrientationSensorListener myListener;
if (gyroExists)
  myListener = new GyroOrientationSensorListener();
else if (newSensorAPIsSupported)
  myListener = new AccOrientationSensorListener();
else
  myListener = new AccOldOrientationSensorListener();

myListener.setOrientationChangeListener(myOCListener);
```

Get real user feedback before launching

- 170+ devices in 100+ countries.
- Use Android Market for private (or public) BETAs.
- Use analytics to find usage patterns and isolate bugs.

```
try {
 // TODO Things that might throw exceptions.
}
catch (Exception ex) {
  string extText = "/exception_type_description";
  MyApplication.getInstance().tracker().trackPageView(exText);
}
```

Get real user feedback before launching

```
private static final boolean isA =
  UUID.randomUUID().getLeastSignificantBits() % 2 == 0;

@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  if (isA) {
    setContentView(R.layout.mainA);
    MyApp.getInstance().tracker().trackPageView("/AUser");
  } else {
    setContentView(R.layout.mainB);
    MyApp.getInstance().tracker().trackPageView("/BUser");
  }
 ...
}
```

Use Android Market for beta testing

- Private betas that require login or passcode.
- Public betas with disguised or obscured Market listing.
- Choose to update listing or create a new one at launch.
- Point beta users to official app at launch.
- Be very clear that it is a beta. Provide venues for feedback.

Rookie Mistake

Not protecting your package name

Protect your package name and keystore

- Upload (don't publish) your package before distributing it to anyone.

Protect your package name and keystore

- Upload (don't publish) your package before distributing it to anyone.
- If you lose your keystore your app becomes an orphan.

Protect your package name and keystore

- Upload (don't publish) your package before distributing it to anyone.
- If you lose your keystore your app becomes an orphan.
- Don't use your personal Gmail for your Android Market Publisher account.

Rookie Mistake

Assuming the natural orientation is portrait

Natural sensor orientation

- Don't assume that the natural orientation is portrait.
- The natural orientation doesn't change when the device moves.
- Always use `Display.getRotation()` to get the screen orientation.

```
int x = AXIS_X;
int y = AXIS_Y;

case (Display.getRotation()):
  Surface.ROTATION_0: break;
  Surface.ROTATION_90:  x = AXIS_Y; y = AXIS_MINUS_X; break;
  Surface.ROTATION_180: y = AXIS_MINUS_Y; break;
  Surface.ROTATION_270: x = AXIS_MINUS_Y; y = AXIS_X; break;
  default: break;
}
SensorManager.remapCoordinateSystem(inR, x_axis, y_axis, outR);
```

Rookie Mistake

Detecting devices rather than installations

Why not detect devices?

- `TelephonyManager.getDeviceId()` returns null if the device doesn't have telephony support.
- MAC address may not work if Bluetooth or WiFi is missing (or off).
- Don't change when the device is wiped. New user. Same DeviceId.
- `Settings.Secure.ANDROID_ID` is reset on wipe but is unreliable pre Android 2.2.

Detecting unique installations

```
private static String uniqueID = null;
private static final String PREF_UNIQUE_ID = "PREF_UNIQUE_ID";

public synchronized static String id(Context context) {
  if (uniqueID == null) {
    SharedPreferences sp = context.getSharedPreferences(
      PREF_UNIQUE_ID, Context.MODE_PRIVATE);
    uniqueID = sp.getString(PREF_UNIQUE_ID, null);
    if (uniqueID == null) {
      uniqueID = UUID.randomUUID().toString();
      Editor editor = sp.edit();
      editor.putString(PREF_UNIQUE_ID, uniqueID);
      editor.commit();
    }
  }
  return uniqueID;
}
```

Rookie Mistake

Not reading the Android Developer Blog

http://android-developers.blogspot.com

The Five Deadly Sins

- Sloth
- Gluttony
- Hostility
- Arrogance
- Discrimination

The Five Glorious Virtues

- Beauty
- Generosity
- Ubiquity
- Generosity
- Epicness


Fresh
Google I/O 11

Being fresh

- Means never having to wait.
- Means always knowing where you are.
- Means always being up to date.

When is the best time to update your app's data?

- Immediately before they look at it:
  - Provided they still have battery,
  - and that they have connectivity and bandwidth.

Use the Passive Location Provider

The Passive Location Provider

- Receives location updates iff another application requests them.
- Requires `ACCESS_FINE_LOCATION` permission.
- `Location.getProvider` reveals the underlying Location Provider.

```
String passiveProvider = LocationManager.PASSIVE_PROVIDER;
locationManager.requestLocationUpdates(passiveProvider,
                                       minTime, minDistance,
                                       myLocationListener);
```

Using Intents to monitor location changes

- Specify a Pending Intent to broadcast when the location changes.
- Location is stored as an extra: `KEY_LOCATION_CHANGED`.
- Useful for multiple Activities / Services that track location.

```
final int resultCode = 0;
final String locAction = "com.ioApp.LOCATION_UPDATE_RECEIVED";
int flags = PendingIntent.FLAG_UPDATE_CURRENT;

Intent intent = new Intent(locAction);
PendingIntent pi = PendingIntent.getBroadcast(this,
  resultCode, intent, flags);

locationManager.requestLocationUpdates(provider, minTime, minDistance, pi);
```

Using Intents to monitor location changes

- Specify a Pending Intent to broadcast when the location changes.
- Location is stored as an extra: `KEY_LOCATION_CHANGED`.
- Useful for multiple Activities / Services that track location.

```
BroadcastReceiver locReceiver = new BroadcastReceiver() {
  @Override
  public void onReceive(Context context, Intent intent) {
    String key = LocationManager.KEY_LOCATION_CHANGED;
    Location location = (Location)intent.getExtras().get(key);
     // [... Do something with the new location ...]
  }
};
IntentFilter locIntentFilter = new IntentFilter(locAction);
registerReceiver(locReceiver, locIntentFilter);
```

Use Intents to update location when your app isn’t running

Using Intents to passively detect location changes

- Your app updates even if it's not running.
- No incremental battery cost.
- Start a Service to refresh the data without updating a UI.

```
<receiver android:name=".locReceiver" android:enabled="true">
  <intent-filter>
    <action android:name="com.ioApp.LOCATION_UPDATE_RECEIVED"/>
  </intent-filter>
</receiver>
```

Check the last known location

```
List<String> providers = lm.getProviders(criteria, false);
for (String provider: providers) {
  Location location = lm.getLastKnownLocation(provider);
  location.getAccuracy();
  location.getTime();
  // Is this the best previously known location?
}
```

Monitor inactive providers for a better option

```
locationManager.getBestProvider(criteria, false);

public void onProviderEnabled(String provider){
  // Switch providers!
}
```

Make your Alarms variable and your Services and Receivers conditional

Is it worth waking up for?

- Set wake alarm for minimum update frequency.
- Set non-waking alarm for optimal update frequency.
- Reset the minimum trigger on each update.

```
int wake  = AlarmManager.ELAPSED_REALTIME_WAKEUP;
int sleep = AlarmManager.ELAPSED_REALTIME;
long minInt  = AlarmManager.INTERVAL_HALF_DAY;
long bestInt = AlarmManager.INTERVAL_HALF_HOUR;
long trigger = SystemClock.elapsedRealtime() + bestInt;

alarms.setInexactRepeating(wake, trigger, minInt, alarmIntent);
alarms.setInexactRepeating(sleep, trigger, bestInt, alarmIntent);
```

Is it worth waking up for?

- Set wake alarm for minimum update frequency.
- Set non-waking alarm for optimal update frequency.
- Reset the minimum trigger on each update.

```
int wake  = AlarmManager.ELAPSED_REALTIME_WAKEUP;
int sleep = AlarmManager.ELAPSED_REALTIME;
long minInt  = AlarmManager.INTERVAL_HALF_DAY;
long bestInt = AlarmManager.INTERVAL_HALF_HOUR;
long trigger = SystemClock.elapsedRealtime() + bestInt;

alarms.setInexactRepeating(wake, trigger, minInt, alarmIntent);
alarms.setInexactRepeating(sleep, trigger, bestInt, alarmIntent);
```

Is it worth waking up for?

- Set wake alarm for minimum update frequency.
- Set non-waking alarm for optimal update frequency.
- Reset the minimum trigger on each update.

```
int wake   = AlarmManager.ELAPSED_REALTIME_WAKEUP;
int sleep = AlarmManager.ELAPSED_REALTIME;
long minInt   = AlarmManager.INTERVAL_HALF_DAY;
long bestInt = AlarmManager.INTERVAL_HALF_HOUR;
long trigger = SystemClock.elapsedRealtime() + bestInt;

alarms.setInexactRepeating(wake, trigger, minInt, alarmIntent);
alarms.setInexactRepeating(sleep, trigger, bestInt, alarmIntent);
```

Monitor device state to vary refresh rate

- Update without connectivity?
- More updates when on WiFi?
- More updates when charging?
- Suspend updates when battery is low?
- More updates when docked?
- Suspend updates while in car dock?

Monitor connectivity

- Monitor connectivity.
- Monitor type of connection.

```
ConnectivityManager cm = (ConnectivityManager)context.
  getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo activeNW = cm.getActiveNetworkInfo();
boolean isConnected = activeNW.isConnectedOrConnecting();
boolean isMobile =
  activeNW.getType() == ConnectivityManager.TYPE_MOBILE;
```

Monitor power and battery

- Sticky broadcast Intent.
- Monitor charging status and level.
- Monitor power source (USB or power).

```
IntentFilter bF = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
Intent bat = context.registerReceiver(null, bF);
int bstat = bat.getIntExtra(BatteryManager.EXTRA_STATUS, -1);
boolean power = bstat == BatteryManager.BATTERY_STATUS_CHARGING ||
                bstat == BatteryManager.BATTERY_STATUS_FULL;
```

Monitor the docking state

- Sticky broadast Intent.
- Monitor dock status and dock type.

```
IntentFilter dFilter = new IntentFilter(Intent.ACTION_DOCK_EVENT);
Intent dock = context.registerReceiver(null, dFilter);
int dState = battery.getIntExtra(BatteryManager.EXTRA_STATUS, -1);
boolean isDocked = dockState != Intent.EXTRA_DOCK_STATE_UNDOCKED;
```

Monitor device state to vary refresh rate

- Update without connectivity?
- More updates when on WiFi?
- More updates when charging?
- Suspend updates when battery is low?
- More updates when docked?
- Suspend updates while in car dock?

Monitor state-change broadcasts

```
<receiver android:name=".UpdateRateMonitorReceiver">
 <intent-filter>
  <action
  android:name="android.intent.action.ACTION_DOCK_EVENT"/>
  <action
  android:name="android.intent.action.ACTION_BATTERY_LOW"/>
  <action
  android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
  <action
  android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
  <action
  android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
 </intent-filter>
</receiver>
```

Monitor state-change broadcasts

```
<receiver android:name=".UpdateRateMonitorReceiver">
 <intent-filter>
  <action
  android:name="android.intent.action.ACTION_DOCK_EVENT"/>
  <action
  android:name="android.intent.action.ACTION_BATTERY_LOW"/>
  <action
  android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
  <action
  android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
  <action
  android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
 </intent-filter>
</receiver>
```

Monitor state-change broadcasts

```
<receiver android:name=".UpdateRateMonitorReceiver">
 <intent-filter>
  <action
  android:name="android.intent.action.ACTION_DOCK_EVENT"/>
  <action
  android:name="android.intent.action.ACTION_BATTERY_LOW"/>
  <action
  android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
  <action
  android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
  <action
  android:name="android.net.conn.CONNECTIVITY_CHANGE"/>
 </intent-filter>
</receiver>
```

Toggle your Manifest Receivers

- Enable and disable receivers for specific state changes.
- Use a Receiver as a passive alarm.

```
ComponentName receiver = new ComponentName(this, myReceiver.class);

PackageManager pm = getPackageManager();

pm.setComponentEnabledSetting(receiver,
  PackageManager.COMPONENT_ENABLED_STATE_ENABLED,
  PackageManager.DONT_KILL_APP);
```

Make your Services intelligent

- Everything should be done asynchronously.
- Service should die as quickly as possible.
- Control restart behaviour based on time since last success.

Make your Services intelligent

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
  startAsynchServiceWorker();
  return forceRefresh() ? START_STICKY : START_NOT_STICKY;
}

private boolean forceRefresh() {
  int failCount = sp.getInt(FAIL_COUNT, 0);
  long lastSuccess = sp.getLong(LAST_SUCCESS, 0);
  Editor editor = sp.edit();
  editor.putInt("FAIL_COUNT", failCount+1);
  editor.commit();

  return ((System.currentTimeMillis()-lastSuccess > maxSuccessLatency) ||
          (failCount > maxFailCount));
}
```


Psychic

Being Psychic

- Means knowing things without being told.
- Leverage what the phone already knows.
- Remember whatever you're told.
- Use Intents to share and consume data anonymously.

Being Psychic

- Means knowing things without being told.
- Leverage what the phone already knows.
- Remember whatever you're told.
- Use Intents to share and consume data anonymously.
- Be conscious of user privacy.

Use the Account Manager to suggest login usernames and email addresses

Use the Account Manager to simplify logins

- Use the Account Manager to suggest your email address.
- Extract and suggest usernames.

```
ArrayList<String> accountNames = new ArrayList<String>();

Account[] accounts = AccountManager.get(context).getAccounts();
for (Account account : accounts) {
  accountNames.add(account.name);
}

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

Never forget

- Sync with web preferences for logged in users.
- Save preferences for anonymous users using UUID.
- Preferences should follow users across devices and resets.
- Let users delete their preferences!

Backup Shared Preferences in the cloud using the Backup Manager

The Android Backup Service

- Persist preferences in cloud storage.
- Restore preferences when the application is re-installed.
- Lets you anonymously track users across installations and devices.

Backup Shared Preferences

```
public class MyPrefsBackupAgent extends BackupAgentHelper {
  static final String PREFS = "user_preferences";
  static final String PREFS_BACKUP_KEY = "prefs";

  @Override
  public void onCreate() {
    SharedPreferencesBackupHelper helper =
      new SharedPreferencesBackupHelper(this, PREFS);
    addHelper(PREFS_BACKUP_KEY, helper);
  }
}
```

Backup Shared Preferences

```
<application android:label="MyApplication"
             android:backupAgent="MyPrefsBackupAgent">
    ...
    <meta-data android:name="com.google.android.backup.api_key"
               android:value="MyAPIKey" />
</application>
```

Persisting unique installations

```
private static String uniqueID = null;
private static final String PREF_UNIQUE_ID = "PREF_UNIQUE_ID";

public synchronized static String id(Context context) {
  if (uniqueID == null) {
    SharedPreferences sp = context.getSharedPreferences(
      PREF_UNIQUE_ID, Context.MODE_PRIVATE);
    uniqueID = sp.getString(PREF_UNIQUE_ID, null);
    if (uniqueID == null) {
      uniqueID = UUID.randomUUID().toString();
      Editor editor = sp.edit();
      editor.putString(PREF_UNIQUE_ID, uniqueID);
      editor.commit();
    }
  }
  return uniqueID;
}
```

Intercept links using an Intent Receiver

Capture links using Intent Filters

- Use Intents to capture links.
- Offer the best way to consume your content.
- Deep link directly to equivalent content.

```
<activity android:name=".MyActivity">
  <intent-filter>
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
    <data android:scheme="http"
          android:host="mysite.com"
          android:pathPrefix="/news/articles/"/>
  </intent-filter>
</activity>
```


Adaptive

Being Adaptive

- Don’t be noticed.
- Be boringly predictable.
- Behave as expected.
- Optimize for different user experiences.

Specify the Edit Text input type to always show the right type of keyboard

Display the right keyboard

- Specify the input type of each EditText control
- Use the `android:inputType` attribute
- Four classes of keyboard:
  - Plain text
  - Decimal number
  - Phone number
  - Date or time

Display the right keyboard

```
<EditText android:id="@+id/editInput"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:inputType="phone"
  android:hint="@string/compose_hint"
/>
```

Use specialized keyboards for text entry

- URIs
- Email addresses
- People's names
- Postal addresses
- Passwords

Use specialized keyboards for text entry

```
<EditText android:id="@+id/editEmailInput"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:inputType="textEmailAddress"
  android:hint="@string/compose_email_hint"
/>
<EditText android:id="@+id/editUriInput"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:inputType="textUri"
  android:hint="@string/compose_uri_hint"
/>
```

Combine variations to simplify text input

- Capitalize each character
- Capitalize each word
- Capitalize each sentence
- Apply auto correction
- Support multi-line input
- Disable suggestions

Combine variations to simplify text input

```
<EditText android:id="@+id/editEmailInput"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:inputType="textShortMessage|
                     textAutoCorrect|
                     textCapSentences|
                     textMultiLine"
  android:maxLines="4"
  android:maxLength="2000"
  android:hint="@string/compose_email_hint"
/>
```

Explicitly set the keyboard action

Customize the IME keyboard behaviour

- Always specify the most appropriate actions:
  - *Go*
  - *Search*
  - *Send*
  - Navigate (*Next* or *Previous*)
  - *Done*
  - *None*
- The default action is *next* if the next field is focussable.
- The default action is *done* if not.
- Enter key will execute the specified (or default) action.
- Use the *No Enter Action* flag to disable action on enter key press.

Combine variations to simplify text input

```
<EditText android:id="@+id/editEmailInput"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:inputType="textShortMessage|
                     textAutoCorrect|
                     textCapSentences"
  android:imeOptions="actionSend|flagNoEnterAction"
  android:hint="@string/compose_email_hint"
/>
```

Combine variations to simplify text input

```
EditText.OnEditorActionListener myActionListener =
  new EditText.OnEditorActionListener() {
    @Override
      public boolean onEditorAction(EditText v,
                                    int actionId, KeyEvent event){
        if (actionId == EditorInfo.IME_ACTION_SEND) {
          // todo Transmit the message
          return true;
        }
        return false;
      }
  };

editText.setOnEditorActionListener(myActionListener);
```

Handle volume and playback buttons while managing the audio focus

Predictable audio behaviour

- Take control of the volume controls.
- Request focus for the media button controls.
- Request (and dismiss) audio focus.
- Gracefully handle interruptions (duck!)

Take control of the volume controls

```
setVolumeControlStream(AudioManager.STREAM_MUSIC);
```

Handle media button control presses

```
<receiver android:name=".MyReceiver">
  <intent-filter>
    <action android:name="android.intent.action.MEDIA_BUTTON" />
  </intent-filter>
</receiver>

public class RemoteControlReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    if (Intent.ACTION_MEDIA_BUTTON.equals(intent.getAction())) {
      KeyEvent event =
      (KeyEvent)intent.getParcelableExtra(Intent.EXTRA_KEY_EVENT);
      if (KeyCode.KEYCODE_MEDIA_PLAY.equals(event.getKeyCode())) {
        // Handle key press.
      }
    }
  }}
```

Handle media button control presses

```
// Start listening for button presses
am.registerMediaButtonEventReceiver(myReceiver);

// Stop listening for button presses
am.unregisterMediaButtonEventReceiver(myReceiver);
```

Requesting audio focus for playback

```
// Request audio focus for playback
int result = am.requestAudioFocus(afChangeListener,
                                  AudioManager.STREAM_MUSIC,
                                  AudioManager.AUDIOFOCUS_GAIN);

if (result == AudioManager.AUDIOFOCUS_REQUEST_GRANTED) {
  // Start playback.
}

// Abandon audio focus when playback complete
am.abandonAudioFocus(afChangeListener);
```

Requesting audio focus for playback

```
// Request audio focus for playback
int result = am.requestAudioFocus(afChangeListener,
                                  AudioManager.STREAM_MUSIC,
                                  AudioManager.AUDIOFOCUS_GAIN);

if (result == AudioManager.AUDIOFOCUS_REQUEST_GRANTED) {
  // Start playback.
}

// Abandon audio focus when playback complete
am.abandonAudioFocus(afChangeListener);
```

Losing audio focus

```
OnAudioFocusChangeListener afChangeListener =
  new OnAudioFocusChangeListener() {
  public void onAudioFocusChange(int focusChange) {
    if (focusChange ==
        AudioManager.AUDIOFOCUS_LOSS_TRANSIENT_CAN_DUCK) {
      // Lower our volume
    } else if (focusChange == AudioManager.AUDIOFOCUS_LOSS) {
      // Pause playback
    }
  }
};
```

Provide translations for your app title and description

Smooth

Being Smooth

- Means never having to wait.
- Be fast, responsive, and consistent.
- Optimize for efficiency.
- Animate everything.
- Move everything off the application thread.

Make everything asynchronous.
No exceptions.

Always be asynchronous

- Make everything asynchronous
  - `Handler`
  - `AsyncTask`
  - `IntentService`
  - `AsyncQueryHandler`
  - `Loader` and `CursorLoader`

Always be asynchronous

- Make everything asynchronous
  - `Handler`
  - `AsyncTask`
  - `IntentService`
  - `AsyncQueryHandler`
  - `Loader` and `CursorLoader`

The Cursor Loader

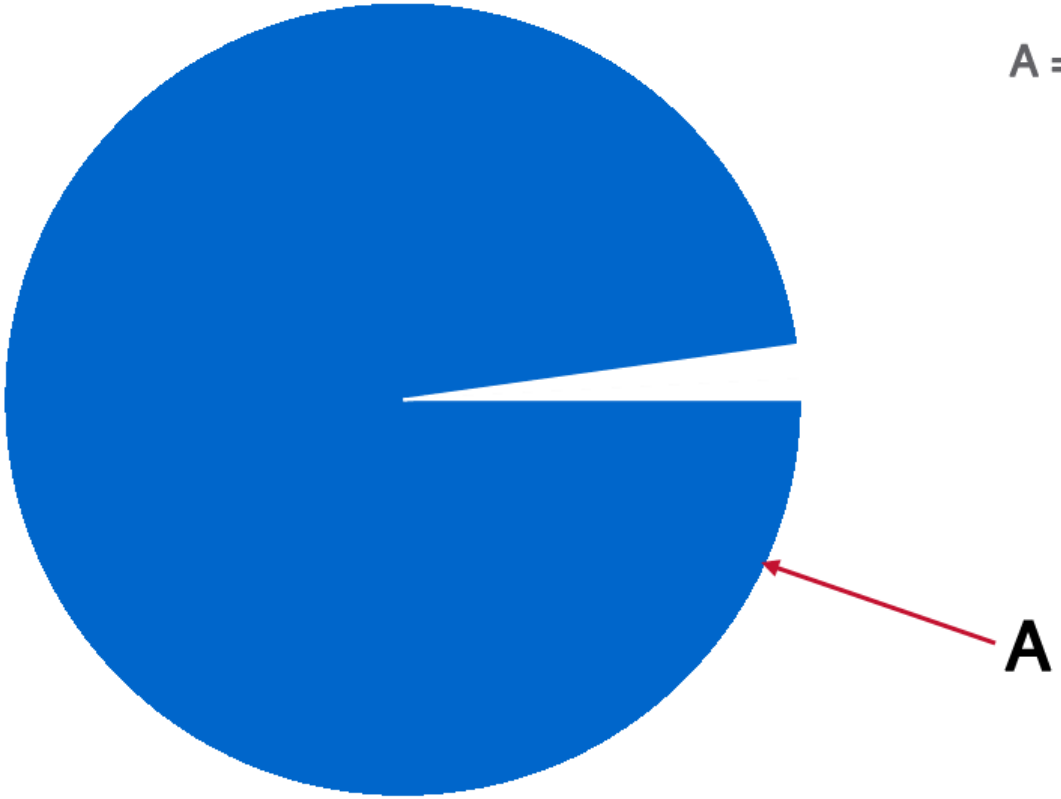
```
// Within onCreate...
getLoaderManager().initLoader(0, null, this);

// Callbacks
public Loader<Cursor> onCreateLoader(int id, Bundle args) {
  Uri baseUri = MyContentProvider.CONTENT_URI;
  return new CursorLoader(getActivity(), baseUri, null, null, null, null);
}

public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
  mAdapter.swapCursor(data);
}

public void onLoaderReset(Loader<Cursor> loader) {
  mAdapter.swapCursor(null);
}
```

Use Strict Mode

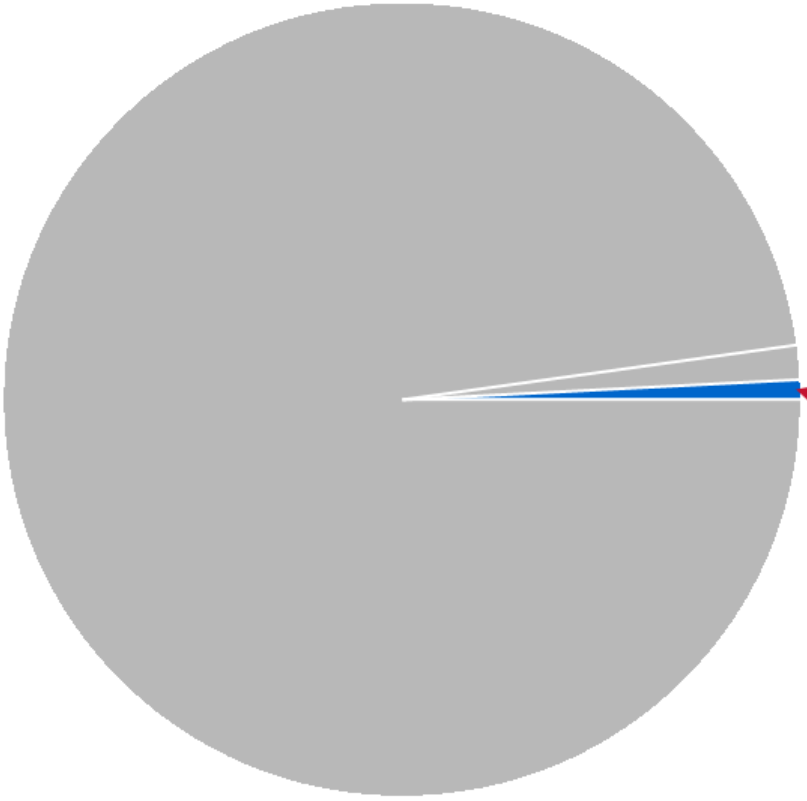
- Turn it **off** for release builds.

```
public void onCreate() {
  if (DEVELOPER_MODE) {
    StrictMode.setThreadPolicy(
      new StrictMode.ThreadPolicy.Builder()
        .detectDiskReads()
        .detectDiskWrites()
        .detectNetwork()
        .penaltyFlashScreen()
        .build());
  }
  super.onCreate();
}
```

A = Want to be featured



All Apps in Android Market



C = Featured

All Apps in Android Market


FRESH


PSYCHIC


FRESH

ADAPTIVE

PSYCHIC

FRESH

ADAPTIVE

PSYCHIC

FRESH

SMOOTH

Want More?

- Twitter: @retomeier
- Stack Overflow: “android”
- http://d.android.com
- Video of this session at Google I/O 2011 http://goo.gl/H5nFe