Android for Java Developers
OSCON 2010

Marko Gargenta
Marakana
About Marko Gargenta

Developed Android Bootcamp for Marakana.

Trained over 1,000 developers on Android.

Clients include Qualcomm, Sony-Ericsson, Motorola, Texas Instruments, Cisco, Sharp, DoD.

Author of upcoming *Learning Android* by O’Reilly.

Spoke at OSCON, ACM, IEEE, SDC. Organizes SFAndroid.org
Agenda

- The Stack
- Android SDK
- Hello World!
- Main Building Blocks
- Android User Interface
- Operating System Features
- Debugging
- Summary
ANDROID STACK
Android runs on Linux.

Linux provides as well as:
- Hardware abstraction layer
- Memory management
- Process management
- Networking

Users never see Linux sub system

The adb shell command opens Linux shell
Native Libraries

**Bionic**, a super fast and small license-friendly libc library optimized for embedded use

**Surface Manager** for composing window manager with off-screen buffering

**2D and 3D graphics** hardware support or software simulation

**Media codecs** offer support for major audio/video codecs

**SQLite** database

**WebKit** library for fast HTML rendering
Dalvik

Dalvik VM is Google’s implementation of Java VM

Optimized for mobile devices

Key Dalvik differences:
- Register-based versus stack-based VM
- Dalvik runs .dex files
- More efficient and compact implementation
- Different set of Java libraries than SDK
Application Framework

The rich set of system services wrapped in an intuitive Java API.

This ecosystem that developers can easily tap into is what makes writing apps for Android easy.

Location, web, telephony, WiFi, Bluetooth, notifications, media, camera, just to name a few.
Applications

Dalvik Executable + Resources = APK
Must be signed (but debug key is okay for development)
Many markets with different policies
Android and Java

Android Java = Java SE – AWT/Swing + Android API
Android SDK - What’s in the box

<table>
<thead>
<tr>
<th>Category</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SDK</strong></td>
<td>adb, hprof-conv</td>
</tr>
<tr>
<td><strong>Tools</strong></td>
<td>android, Jet</td>
</tr>
<tr>
<td><strong>Docs</strong></td>
<td>apkbuilder, layoutopt</td>
</tr>
<tr>
<td><strong>Platforms</strong></td>
<td>ddms, lib, mksdcard</td>
</tr>
<tr>
<td><strong>Data</strong></td>
<td>dmtracedump</td>
</tr>
<tr>
<td><strong>Skins</strong></td>
<td>sqlite3</td>
</tr>
<tr>
<td><strong>Images</strong></td>
<td>draw9patch, emulator</td>
</tr>
<tr>
<td><strong>Samples</strong></td>
<td>hierarchyviewer, traceview</td>
</tr>
<tr>
<td><strong>Add-ons</strong></td>
<td>zipalign</td>
</tr>
<tr>
<td><strong>Google</strong></td>
<td></td>
</tr>
</tbody>
</table>
HELLO WORLD!

HELLO WORLD!
LET'S HAVE A PICNIC!!
Create New Project

Use the Eclipse tool to create a new Android project.

Here are some key constructs:

<table>
<thead>
<tr>
<th>Project</th>
<th>Eclipse construct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>minimum to run</td>
</tr>
<tr>
<td>App name</td>
<td>whatever</td>
</tr>
<tr>
<td>Package</td>
<td>Java package</td>
</tr>
<tr>
<td>Activity</td>
<td>Java class</td>
</tr>
</tbody>
</table>
The Manifest File

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.marakana"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".HelloAndroid"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="5" />
</manifest>
```
The Layout Resource

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
>
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
</LinearLayout>
```
package com.marakana;

import android.app.Activity;
import android.os.Bundle;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
Running on Emulator

Hello, Android!!!
Hello World! Look ma, it works!
MAIN BUILDING BLOCKS
Activities

Activity is to an application what a web page is to a website. Sort of.
Activity Lifecycle

Activities have a well-defined lifecycle. The Android OS manages your activity by changing its state. You fill in the blanks.
Intents represent an events or actions.

They are to Android apps what hyperlinks are to websites. Sort of.

Intents can be implicit or explicit.
Services

Services are code that runs in the background. They can be started and stopped. Services doesn’t have UI.
Service Lifecycle

Service also has a lifecycle, but it’s much simpler than activity’s.

An activity typically starts and stops a service to do some work for it in the background, such as play music, check for new tweets, etc.
Content Providers

Content Providers share content with applications across application boundaries. Examples of built-in Content Providers are: Contacts, MediaStore, Settings and more.
Broadcast Receivers

An Intent-based publish-subscribe mechanism. Great for listening system events such as SMS messages.
**MyTwitter – A Real World App**

- **MyTwitter Activity**
- **Preference Activity**
- **Timeline Receiver**
- **Timeline Activity**
- **Updater Service**
- **Prefs XML**
- **Timeline DB**
- **Timeline Adapter**
- **Boot Receiver**

**Twitter.com**
- Updates Status via web service
- Pull timeline updates via web service
- Start at boot

**Preference Activity**
- Read/write preferences
- Read Prefs
- Notify of new status

**Timeline Receiver**
- Update list
- Pull timeline from DB

**Updater Service**
- Insert updates in DB
- Notify of new status
- Pull timeline updates via web service

**Timeline Activity**
- Update ListView
- Read Prefs
- Read Prefs

**Boot Receiver**
- Start at boot
ANDROID USER INTERFACE
Two UI Approaches

<table>
<thead>
<tr>
<th>Procedural</th>
<th>Declarative</th>
</tr>
</thead>
<tbody>
<tr>
<td>You write Java code</td>
<td>You write XML code</td>
</tr>
<tr>
<td>Similar to Swing or AWT</td>
<td>Similar to HTML of a web page</td>
</tr>
</tbody>
</table>

You can mix and match both styles. Declarative is preferred: easier and more tools.
XML-Based User Interface

Use WYSIWYG tools to build powerful XML-based UI. Easily customize it from Java. Separate concerns.
# Dips and Sps

<table>
<thead>
<tr>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>px</strong> (pixel)</td>
<td>Dots on the screen</td>
</tr>
<tr>
<td><strong>in</strong> (inches)</td>
<td>Size as measured by a ruler</td>
</tr>
<tr>
<td><strong>mm</strong> (millimeters)</td>
<td>Size as measured by a ruler</td>
</tr>
<tr>
<td><strong>pt</strong> (points)</td>
<td>1/72 of an inch</td>
</tr>
<tr>
<td><strong>dp</strong> (density-independent pixel)</td>
<td>Abstract unit. On screen with 160dpi, 1dp=1px</td>
</tr>
<tr>
<td><strong>dip</strong></td>
<td>Synonym for dp and often used by Google</td>
</tr>
<tr>
<td><strong>sp</strong></td>
<td>Similar to dp but also scaled by users font size preference</td>
</tr>
</tbody>
</table>
Views and Layouts

Layouts contain other Views, or other Layouts.
Common UI Components

Android UI includes many common modern UI widgets, such as Buttons, Tabs, Progress Bars, Date and Time Pickers, etc.
Selection Components

Some UI widgets may be linked to zillion pieces of data. Examples are ListView and Spinners (pull-downs).
Adapters

To make sure they run smoothly, Android uses Adapters to connect them to their data sources. A typical data source is an Array or a Database.
Complex Components

Certain high-level components are simply available just like Views. Adding a Map or a Video to your application is almost like adding a Button or a piece of text.
Menus and Dialogs

If you want to choose another menu resource, go back and re-run this activity.
Graphics & Animation

Android has rich support for 2D graphics. You can draw & animate from XML. You can use OpenGL for 3D graphics.
Multimedia

**AudioPlayer** lets you simply specify the audio resource and play it.

**VideoView** is a View that you can drop anywhere in your activity, point to a video file and play it.

**XML:**

```xml
<VideoView
    android:id="@+id/video"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_gravity="center" />
```

**Java:**

```java
player = (VideoView) findViewById(R.id.video);
player.setVideoPath("/sdcard/samplevideo.3gp");
player.start();
```
OPERATING SYSTEM FEATURES
Each Android application runs inside its own Linux process.

Additionally, each application has its own sandbox file system with its own set of preferences and its own database.

Other applications cannot access any of its data, unless it is explicitly shared.
The file system has three main mount points. One for system, one for the apps, and one for whatever.

Each app has its own sandbox easily accessible to it. No one else can access its data. The sandbox is in /data/data/com.marakana/

SDCard is expected to always be there. It’s a good place for large files, such as movies and music. Everyone can access it.
Cloud to Device Push

Announcing: Cloud-to-Device Messaging API

Big deal for many pull-based apps. Will make devices use less battery.
Preferences

Your app can support complex preferences quite easily.

You define your preferences in an XML file and the corresponding UI and data storage is done for free.
SQLite Database

Android ships with SQLite3

SQLite is

- Zero configuration
- Serverless
- Single database file
- Cross-Platform
- Compact
- Public Domain

Database engine.

May you do good and not evil
May you find forgiveness for yourself and forgive others
May you share freely, never taking more than you give.
DEBUGGING
ANDROID APPS
The universal, most versatile way to track what is going on in your app.

Can be viewed via command line or Eclipse.

Logs can be generated both from SDK Java code, or low-level C code via Bionic libc extension.
Debugger

Your standard debugger is included in SDK, with all the usual bells & whistles.
TraceView helps you profile your application and find bottlenecks. It shows execution of various calls through the entire stack. You can zoom into specific calls.
Hierarchy Viewer

Hierarchy Viewer helps you analyze your User Interface.

Base UI tends to be the most “expensive” part of your application, this tool is very useful.
Summary

Android is open and complete system for mobile development. It is based on Java and augmented with XML.

Android is being adopted very quickly both by users, carriers, and manufacturers.

It takes about 3-5 days of intensive training to learn Android application development for someone who has basic Java (or similar) experience.

Marko Gargenta, Marakana.com
marko@marakana.com
+1-415-647-7000

Licensed under Creative Commons License (cc-by-nc-nd) – non-commercial. Please Share!