Busy Java Developer's Guide to Android:Basics

Ted Neward

Neward & Associates

http://www.tedneward.com | ted@tedneward.com

Credentials

Who is this guy?

- Principal, Architect, Consultant and Mentor

  ask me how I can help your project or your team
- Microsoft MVP (F#, C#, Architect)
- JSR 175, 277 Expert Group Member
- Author

  Professional F# 2.0 (w/Erickson, et al; Wrox, 2010)
  Effective Enterprise Java (Addison-Wesley, 2004)
  C# In a Nutshell (w/Drayton, et all; OReilly, 2003)
  SSCLI Essentials (w/Stutz, et al; OReilly, 2003)
  Server-Based Java Programming (Manning, 2000)
- Blog: http://blogs.tedneward.com
- Papers: http://www.tedneward.com/writings
- Twitter: @tedneward

Objectives

My job...

- ... is to show you enough Android to make you dangerous
- ... because I can't exhaustively cover the entire platform in just one conference session

To that end...

- ... I will show you the (barebones) tools
- ... I will show you some basics
- ... I will explain a few core concepts
- ... I will leave it up to you to research further

Getting Started

"Begin at the beginning"

--The White Rabbit

Concepts

- Activities
- Intents
- Services
- Content Providers

Features

- Storage (SD cards)
- Network
- Camera/Video
- OpenGL
- Sound/Music
- GPS
- Phone/SMS

Tools

Android tooling consists of:

- JDK 1.6.latest
- Android SDK
  - **Android SDK installer/updater**
  - **Android libraries & documentation (versioned)**
  - **Android emulator**
  - **ADB**
- an Android device (optional, sort of)

  mine is a Motorola Droid (for now...)
- IDE w/Android plugins (optional)
  - **Eclipse is the oldest; I don't particularly care for it**
  - **IDEA 10 rocks; Community Edition is free**
  - **Even NetBeans has an Android plugin**

Bits

- Android is not a JVM
  - Android is a mix of native and VM (Dalvik) code
  - Runs on top of a specialized build of Linux
  - Java source is compiled to JVM bytecode
  - Postprocessor converts JVM bytecode to DVM bytecode
  - All code and files are packaged up (APK files)
  - APK downloaded to phone, installed, and ready

Getting Dirty

"In theory, there's no difference
between theory and practice.
In practice, however..."

--Yogi Berra

Setting up

- Once the tools are installed…
  - put “android” on your command-line PATH
  - create a new AVD (Android Virtual Machine)

    android create avd --target 7 --name TestAVD
  - create a new Android project

    android create project --target 7 --package com.tedneward.tutorial --path ./HelloAndroid --activity HelloActivity

Getting Dirty

- Project contents
  - build.xml: Ant build script integrated with Android SDK
  - build.properties, local.properties: settings for build
  - AndroidManifest.xml: Android manifest file (more later)
  - assets/: static files (for packaging in the app)
  - bin/: generated binaries (including the APK file)
  - gen/: generated source files
  - libs/: Java jars used in your app
  - src/: your app source
  - res/: resources used in the app
  - tests/: unit-tests

Getting Dirty

- Resources (res/ directory)
  - res/drawable/ : graphics (PNG, JPEG, etc)
  - res/layout/ : UI XML layout files
  - res/menu/ : UI menu XML files
  - res/raw/ : "general-purpose" files (e.g., anything else)
  - res/values/ : general constants
  - res/xml/ : any other XML files included in the build

Getting Dirty

- AndroidManifest.xml: application manifest
  - uses-permission: what does the app do?
  - permissions: what new permissions does the app define?
  - instrumentation: what events should be hooked?
  - uses-library: which optional components does app use?
  - uses-sdk: which SDK was/is used?
  - application: what are the contents in the app?

    activity: defines an activity
    service: defines a service
    provider: defines a content provider
    receiver: defines a broadcast receiver

Core Basics

"How do it work?"

UI Basics

- Activity
  - somewhat akin to a web page or screen
  - public class that extends android.app.Activity
  - activities form a "stack of cards" as the user moves around

    "back" button on the handset removes the top card
    "going back" doesn't necessarily kill the activity, though
  - activities have event methods for override

    onCreate: activity "constructor"
    onDestroy: activity "finalizer"
    onStart/onStop/onPause/onResume/onRestart
    remember to call up the inheritance chain

UI Basics

- Controls
  - Label: text (readonly) label
  - Button, Checkbox, Radiobutton
  - Image
  - more (listboxes, etc)
- Control creation
  - either create controls “by hand” in Java
  - or using XML-based layout file
  - pros/cons either way

UI Basics

- Moving from one Activity to another
  - requires an "intent" to be passed to the Android OS for processing and evaluation
  - Intent = Action + Context
  - easiest Intent is the "launch the Activity" Intent

```
Intent next = new Intent(this, NextActivity.class);
startActivity(next);
```

Threading

- Good Thread hygiene is critical here
  - This is a phone—you don't own the machine!
  - Android isn't quite like Java, and has a slightly different "take" on the threads-and-UI position

    NO UI modifications from non-UI thread
    If the UI thread is inactive for more than 5 seconds, bye!
  - Android provides Handlers, which can be sent Messages that will then be processed on the Activity's UI thread
  - Java5 provides a few other constructs as well

Summary

- Android is a Java-based mobile device framework
  - ... but it's not Java
  - ... and it's definitely not Swing or SWT
  - ... but it does let you use the Java parts you love
  - ... so long as they fit a (relatively) narrow profile

Resources

- Busy Coder’s Guide to Android

  Mark Murphy, http://www.commonsware.com
- Android website (http://www.android.com)
- The remainder of this presentation series
  - Busy Android Dev's Guide to UI
  - Busy Android Dev's Guide to Persistence
  - Busy Android Dev's Guide to Communication
  - ... and more