

Problems and Solutions for Android Developers



Android Cookbook

O'REILLY®

Edited by Ian F. Darwin

Android Cookbook

Android Cookbook

Android Community Experts

Android Cookbook

by Android Community Experts

Copyright © 2011 Ian Darwin and Contributors. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://safari.oreilly.com>). For more information, contact our corporate/institutional sales department: (800) 998-9938 or corporate@oreilly.com.

Editor: Ian F. Darwin

Production Editor:

Copyeditor:

Proofreader:

Indexer:

Cover Designer:

Interior Designer:

Illustrators: and

November 2011: First Edition.

Revision History for the First Edition:

See <http://oreilly.com/catalog/errata.csp?isbn=9781449388416> for release details.

Android is a trademark of Google, Inc. for their open-source operating environment for mobile devices. Linux is a trademark of Linus Torvalds. Java is a trademark of Oracle America Corporation (formerly Sun Microsystems).

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. **!!FILL THIS IN!!** and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN: 978-1-449-38841-6

[?]

1318018084

Table of Contents

Preface	xiii
1. Getting Started	1
1.1 Introduction: Getting Started	1
1.2 Learning the Java Language	1
1.3 Hello World - Command Line	3
1.4 Hello World - Eclipse Version	5
1.5 Set Up an Android Virtual Device for Apps Testing	10
1.6 Set Up an IDE on Windows to Develop for Android	23
1.7 Android Lifecycle	32
1.8 Opening a Web Page, Phone Number or anything else with an Intent	33
1.9 Email Text From a View	35
1.10 Sending an email with attachments	38
1.11 Installing .apk files on the emulator	40
1.12 Installing apps onto an Android Emulator	40
1.13 Android Epoch HTML/Javascript Calendar	43
1.14 Sharing Java classes from another Eclipse Project	48
1.15 Referencing libraries to implement external functionality	49
1.16 Use SDK Samples to Help Avoid Head Scratching	50
1.17 Keeping the Android SDK Updated	55
1.18 Five Ways to Wire Up an Event Listener	61
1.19 Taking a Screenshot from the Emulator/Android Device	69
1.20 Program: A Simple CountdownTimer example	70
1.21 Program: Tipster, a tip calculator for the Android OS	73
2. Designing a successful Application	91
2.1 Introduction: Designing a Successful Android application	91
2.2 Keeping a Service running while other apps are on display	95
2.3 Starting a service after phone reboot	97
2.4 Exception Handling	98
2.5 Sending/Receive broadcast message	100

2.6	Android's Application Object as a "Singleton"	101
2.7	Keeping data when the user rotates the device	103
2.8	Creating a Responsive Application using Threads	105
2.9	Eating Too Much CPU Time In The UI Causes A Nasty Result	107
2.10	AsyncTask: Do background tasks and modify the GUI	113
2.11	Monitoring the Battery Level of your Android Device	114
2.12	Splash Screens in Android: Method 1	115
2.13	Splash Screens in Android: Method 2	117
2.14	Designing a Conference/*Camp/Hackathon App	123
2.15	Implementing Autocompletion in Android.	124
2.16	Using Google Analytics in Android Application	126
2.17	Using AsyncTask to do background processing	128
2.18	A Simple Torch Light	134
2.19	Adapting Android Phone Application to Tablet	136
2.20	First Run preferences	138
2.21	Formatting the time and date display	139
2.22	Controlling Input with KeyListeners	142
2.23	Android Application Data Backup	145
2.24	Making password fields	151
2.25	Working Without Tooltips: Use Hints Instead	152
3.	Testing	157
3.1	Introduction: Testing	157
3.2	How to TDD(test driven development) Android App	157
3.3	How to troubleshoot "The application has stopped unexpectedly. Please try again"	158
3.4	Debugging using Log.d and LogCat	161
3.5	Keep Your App Snappy With StrictMode	163
3.6	Barrel of Monkeys	164
3.7	Sending text messages and placing calls between AVDs	165
3.8	Activity LifeCycle Scenarios for Testing	166
4.	Content Providers	175
4.1	Introduction: Content Providers	175
4.2	Retrieving Data from a Content Provider	175
4.3	Writing a Content Provider	177
4.4	Android Remote Service	179
5.	Graphics	185
5.1	Introduction: Graphics	185
5.2	Getting Screenshots	185
5.3	Using a Custom Font	186
5.4	Draw a spinning cube with OpenGL ES	191

5.5	Adding control to the OpenGL spinning cube	196
5.6	Taking a Picture Using an Intent	199
5.7	Taking a Picture Using android.media.Camera	201
5.8	Using AndroidPlot to display charts and graphs in your Android application.	205
5.9	Use Inkscape to Create an Android Launcher Icon	207
5.10	Easy Launcher Icons from OpenClipArt.org using Paint.NET	216
5.11	Android HTML5 RGraph Charting	228
5.12	Simple Raster Animation	232
6.	Graphical User Interface	239
6.1	Introduction: GUI	239
6.2	User Interface Guidelines (placeholder)	240
6.3	SlidingDrawer Overlapping other UI components	240
6.4	Android 3.0 Photo Gallery	244
6.5	Building a UI using Fragments API of Android 3.0 in Android 2.2	246
6.6	Haptic Feedback	250
6.7	Handling Configuration Changes by Decoupling View from Model	254
6.8	Let Them See Stars: Using RatingBar	257
6.9	Invoke an action handler when a Button is pressed	260
6.10	Creating an Alert Dialog.	263
6.11	Customize the SlidingDrawer component to animate/transition from the top down.	264
6.12	Use a Timepicker widget	266
6.13	Formatting with Correct Plurals	268
6.14	Feed AutoCompleteTextView using a SQLite database query	272
6.15	Change The Enter Key to "Next" on the Soft Keyboard	273
6.16	How to Create a Simple Widget	277
6.17	Make a View Shake	280
6.18	Using CheckBoxes and RadioButtons	281
6.19	Creating a Notification in the Status Bar	286
6.20	Autocompletion with Icons/Images	288
6.21	Creating your own Custom Title Bar	295
6.22	iPhone-like wheel picker for selection	298
6.23	Simple Calendar	302
6.24	Formatting Numbers	310
6.25	Start a Second Screen from the First	314
6.26	Creating a Tabbed Dialog	322
6.27	Creating a Custom Dialog with buttons, images and text	326
6.28	Create a Custom Menu	328
6.29	Loading Screen in between two Activities	330
6.30	Implementing reactions on click of items in a Custom Menu.	333
6.31	Navigate different activities within a TabView	336

6.32	Drop-down Chooser via the Spinner Class	338
6.33	Effective UI design using Image Buttons	340
6.34	Pinch to zoom	343
6.35	Add a Border with Rounded Corners to a Layout	346
6.36	Creating a ProgressDialog in Android.	347
6.37	Creating a Submenu.	349
6.38	Processing key press events in an Activity.	351
6.39	Constrain EditText Values with Attributes and the TextWatcher Interface	352
6.40	Gesture Detection in Android	355
6.41	Customizing the Look of a Toast	362
6.42	Using SlidingDrawer to Overlap Other Components	363
7.	GUI: ListView	367
7.1	Introduction: ListView	367
7.2	Building list-based applications with ListView	367
7.3	'No data' View for Lists	372
7.4	Advanced ListView: populating a list with images and text	373
7.5	ListView with Icons/images	379
7.6	Sectioned Headers in ListViews	386
7.7	Making Lists Behave Nicely	392
7.8	Writing A Custom List Adapter	393
7.9	Orientation Changes : From ListView data values to Landscape Charting	396
8.	Multimedia	407
8.1	Introduction: Multimedia	407
8.2	Play a Youtube Video	407
8.3	Using Gallery with ImageSwitcher	408
8.4	Grabbing a video using MediaRecorder	411
8.5	Android Face Detection	414
8.6	Playing audio from a file	417
8.7	Playing Audio without Interaction	420
8.8	Using Speech to Text	421
8.9	Making the Device Speak with TTS	423
9.	Data Persistence	427
9.1	Listing a Directory	427
9.2	Default shared preferences consistency check	429
9.3	Advanced text search	431
9.4	How to push string-values using Intent.putExtra()	437
9.5	Retrieving data from a Sub-Activity back to your Main Activity	439
9.6	Getting total and free space on the SD card	442

9.7	Creating a SQLite database in an Android application.	442
9.8	Retrieving data from a SQLite database.	444
9.9	Inserting values into a SQLite database.	445
9.10	Work With Dates in SQLite	445
9.11	Parsing JSON using the Jackson Parser	448
9.12	Parsing an XML document using the DOM API	451
9.13	Parsing an XML document using an XmlPullParser	453
9.14	Accessing data from a file shipped with the App rather than in the filesystem	456
9.15	Adding a Contact	457
9.16	Reading Contact Data	461
9.17	Parsing JSON using JSONObject	463
10.	Telephone Applications	467
10.1	Introduction: Telephone Applications	467
10.2	Do something when the phone rings	467
10.3	Process outgoing calls	471
10.4	Dialing the phone	475
10.5	Sending single or multipart SMS messages	476
10.6	Receiving an SMS in an Android Application.	478
10.7	Using Emulator Controls to send SMS to the Emulator.	480
10.8	Android TelephonyManager.	480
11.	Networked Applications	491
11.1	Introduction: Networking	491
11.2	Using a RESTful Web Service	491
11.3	Extracting Information from Unstructured Text using Regular Expressions	494
11.4	Parsing RSS/ATOM feeds parsing with ROME	496
11.5	Using MD5 to Digest Free Text	500
11.6	Converting text into hyperlinks	502
11.7	Accessing a web page through your Android application	503
11.8	Customizing a WebView	505
12.	Gaming and Animation	507
12.1	Introduction: Gaming and Animation	507
12.2	Android Game Programming - Introduction to Flixel-Android	508
12.3	Introduction to Game Programming using AndEngine (Android-Engine)	510
13.	Social Networking	517
13.1	Facebook Integration	517
13.2	Social Networking Integration using Http	525

13.3	Loading a user's Twitter timeline (using JSON)	528
14.	Location and Map Applications	533
14.1	Introduction: Location-Aware Applications	533
14.2	Getting Location Information	533
14.3	Access GPS information anywhere in your application	535
14.4	Mocking GPS Coordinates On A Device	537
14.5	Geocoding and Reverse Geocoding	539
14.6	Getting ready for Google Maps development	540
14.7	Using Google Maps in your Android App	547
14.8	How to show your current location in a map	548
14.9	To Add Device's current location to Google Maps	549
14.10	Draw a location marker on a Google MapView	550
14.11	Drawing multiple location markers on a MapView	556
14.12	Creating Overlays for a Google MapView	560
14.13	Changing Views of a MapView.	561
14.14	Draw overlay icon without using Drawable	562
14.15	Location search on Google maps	567
14.16	MapView inside TabView	568
14.17	Handling longpress in a map	572
14.18	Using OpenStreetMap	575
14.19	Creating overlays in OpenStreetMaps	576
14.20	Using a scale on an OpenStreetMap	579
14.21	Handling touch events on an OpenStreetMap Overlay	582
14.22	Getting location updates with OpenStreetMaps	584
15.	Accelerometer	593
15.1	Using the accelerometer to detect shaking of the device	593
15.2	Introduction: Sensors	596
15.3	Checking for device facing up or facing down based on screen orientation using Accelerometer.	597
15.4	Finding the orientation of an Android device using Orientation sensor.	598
15.5	Checking for the Presence or Absence of a Sensor	600
15.6	Reading the Temperature Sensor	601
16.	Bluetooth	603
16.1	Introduction: Bluetooth	603
16.2	Connecting to Bluetooth enabled device	603
16.3	Enabling Bluetooth and making the device Discoverable.	606
16.4	Listening for Bluetooth Connection Requests.	607
16.5	Bluetooth Device discovery	609

17. System and Device Control	611
17.1 Phone network/connectivity information	611
17.2 Changing incoming call notification to Silent, Vibrate, or normal	612
17.3 Rebooting the Device	614
17.4 Running shell commands from your application	616
17.5 Copying text and getting text from the Clipboard	617
17.6 Making LED based notifications	619
17.7 Making the Device Vibrate.	620
17.8 Determining Whether a Given Application is Running	621
18. Other Programming Languages	623
18.1 Run external/native Linux command	623
18.2 Running Adobe Air/Flex on Android	624
18.3 Getting Started with "Scripting Layer for Android" (formerly Android Scripting Environment)	625
18.4 Running Native Code with JNI on the NDK	627
18.5 Introduction: Other Programming Languages	632
18.6 Intro to Flex 4.5 Android Programming	634
18.7 Sharing your scripts (ASE) using QR codes	636
18.8 Using native handset functionality from webview using Javascript	638
19. Internationalization	641
19.1 Introduction: Internationalization	641
19.2 Internationalizing Application Text	642
20. Packaging, deploying and selling	647
20.1 Signing Your Application	647
20.2 How to integrate Admob into your app	648
20.3 Distributing Your Application via the Android Market	652
20.4 Creating a Signing Certificate	654
20.5 Obfuscating and Optimizing with ProGuard	657
20.6 Provide a Link to other Published Apps in the Market	660
21. Other	663
21.1 Introduction: Everything Else	663
21.2 Sending messages between threads using activity thread queue and Handler class	663
21.3 Intercommunication amongst Applications	665
22. Contributors	667
22.1 Names	667

Preface

Preface

Ian Darwin

Android is "the open source revolution" applied to cellular telephony. At least, part of it. There are many other attempts to provide open source cell phones, ranging from the mostly-defunct *Openmoko Freerunner* through QT Embedded, Moblin, LiMo, Debian Mobile, Maemo to the *recently-open-sourced Symbian OS*. Not to mention the established non-open-source stalwarts: Blackberry OS, Apple's iPhone, and Microsoft Windows Mobile (these have developer toolkits, but their OS is not available as open source).

"Nobody's armchair is a good predictor of the future", though, as Mike O'Dell once said. Does Android have a place in the sun alongside these other players? We think it does. This book is here to help the Android developer community share the knowledge that will make it happen. Those who contribute knowledge here are helping make Android development easier for those who come after.

About Android

Android is a mobile technology platform that provides cell phones, tablets and other hand-held and mobile devices (even netbooks) with the power and portability of the Linux operating system and the reliability and portability of a standard high-level language and API. Android apps are written in the Java language, using tools such as Eclipse, compiled against the Android API, and translated into bytecode for the Dalvik VM.

Android is thus related by OS family to Openmoko, QT Embedded, MeeGo (the 2010 *merger of Nokia's Maemo and Intel's MobLin*), OPhone, LiMo and other Linux-based cell phone projects. Android is also related by programming language to Blackberry and JavaME phones, and to Java and the wider realm of Java Enterprise applications.

Android sales have continued to climb; there is a report from NPD that *first-quarter 2010 sales of all Android devices exceeded sales of the iPhone*, moving it into second

place (although still well behind the Blackberry platform). Surely it was due in part to major carrier Verizon's 2-for-1 sale, but that doesn't account for all of it...

Who This Book Is From

This book was written by several dozens of Android developers from the Android community at large. Development occurred in the open, on the web site Android-Cookbook.com, which I wrote to allow people to contribute, view, review and comment upon, the recipes that would make up this book. A complete list can be found in [Chapter 22](#). I am deeply grateful to all the contributors, who have helped moved this book from a dream to the reality that you have in your hands (or on-screen if you are reading the eBook format). Thank you all!

Who This Book Is For

We assume you know the basics of the Java language. If not, see “[Preface](#)” on [page xiii](#). We also assume you know the basics of the Java Standard Edition API (since this forms the basis of Android's runtime libraries) as well as the basics of Android. The terms "activity", "intent", and "content provider", while not necessarily being what you dream about at night, should at least be familiar to you.

What's in this Book?

[Chapter 1, *Getting Started*](#), takes you through the steps of setting up the Android development environment and building several simple applications of the well-known "Hello World" type pioneered by Brian Kernighan.

[Chapter 2, *Designing a successful Application*](#), covers some of the differences in mobile computing that will hit developers coming from desktop and enterprise software environments, and talks about how mobile design (in particular Android design) differs from those other environments.

Testing is often an afterthought for some developers, so we put this early on, in [Chapter 3, *Testing*](#). Not so you'll skip it, but so you'll read and heed. We talk about unit testing individual components as well as testing out your entire application in a well-controlled way.

In [Chapter 4, *Content Providers*](#), we show you how to make an application that can be used by other applications through something as simple but ubiquitous (in Android) as the URL.

[Chapter 5, *Graphics*](#), covers a range of topics related to graphics, including use of the graphical drawing and compositing facilities in Android as well as using desktop tools to develop graphical images, textures, icons, and so on that will be incorporated into your finished application.

Every mobile app needs a GUI, so [Chapter 6, *Graphical User Interface*](#), covers all the ins and outs of GUI development for Android. Examples are given both in XML and in hard-coded GUI development.

[Chapter 7, *GUI: ListView*](#), focuses on one of the most important Graphical User Interfaces in Android, the `ListView`.

Android is rich in multimedia capabilities. [Chapter 8, *Multimedia*](#), shows how.

[Chapter 9, *Data Persistence*](#), shows how to save data into files, databases and so on. And how to retrieve it later, of course.

Android started out as an operating system for mobile telephones. [Chapter 10, *Telephone Applications*](#), shows how to control and react to the telephone device that is in most mobile devices nowadays.

Mobile devices are, for the most part, always-on and always-connected. This has a major impact on how people use them and think about them. [Chapter 11, *Networked Applications*](#), shows the coding for traditional networked applications. This is followed by [Chapter 12, *Gaming and Animation*](#), and [Chapter 13, *Social Networking*](#).

The now-ubiquitous Global Positioning System has also had major implications on how mobile applications work. [Chapter 14, *Location and Map Applications*](#), discusses how to find your location, how to get map data from Google and OpenStreetMap, and how applications can be location-aware in ways that are just now being explored.

[Chapter 15, *Accelerometer*](#), talks about the sensors built into most Android devices and how to use them.

There may be a [Chapter 16, *Bluetooth*](#), if there's enough to say about it, going way beyond connecting your Bluetooth(TM) headset to your phone. This is followed by [Chapter 17, *System and Device Control*](#).

In [Chapter 18, *Other Programming Languages*](#), we explore the use of other programming languages to write all or part of your Android application. Examples include C, Perl, Python, Lisp, and other languages.

While this book is in English, and English remains the #1 language worldwide, it is far from the only one. And most end users would much rather have an application that has its text in their language and its icons in a form that is culturally correct for them. [Chapter 19, *Internationalization*](#), goes over the issues of language and culture and how it relates to Android.

Most Android developers hope that their applications will be used by other people. But this won't happen unless users can find your application. [Chapter 20, *Packaging, deploying and selling*](#), shows how to prepare your application for distribution via the Android Market, and to use that as well as other markets to get your application out to the people that will use it.

Finally, [Chapter 21, *Other*](#), covers a few miscellaneous topics that don't quite fit anywhere else.

Other Books You May Like

Java Books

T.B.A.

Android Books

T.B.A.

Programming and Design Books

T.B.A.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, and file extensions.

Constant width

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

Constant width bold

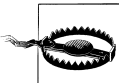
Shows commands or other text that should be typed literally by the user.

Constant width italic

Shows text that should be replaced with user-supplied values or by values determined by context.



This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.


Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Book Title* by Some Author. Copyright 2008 O'Reilly Media, Inc., 978-0-596-xxxx-x."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.

Safari® Books Online

 Safari Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at <http://my.safaribooksonline.com>.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at:

<http://www.oreilly.com/catalog/9781449388416>

To comment or ask technical questions about this book, send email to:

bookquestions@oreilly.com

For more information about our books, conferences, Resource Centers, and the O'Reilly Network, see our website at:

<http://www.oreilly.com>

Getting Started

1.1 Introduction: Getting Started

Ian Darwin

Discussion

The famous "Hello, World" pattern came about when Kernighan and Plaugher wanted to write a "Recipe" on how to get started in any new programming language and environment. This chapter is affectionately dedicated to these fine gentlemen, and to everyone who has ever struggled to **get started** in a new programming paradigm.

1.2 Learning the Java Language

Ian Darwin

Problem

Android apps are written in the Java(tm) programming language before they are converted into Android's own class file format, DEX. If you don't know how to program in Java you will find it hard to write Android apps.

Solution

There are lots of resources available to learn Java. Most of them will teach you what you need, but will also teach some API classes that are not available. **Avoid** any sections in any book that talk about topics in the left-hand column:

Table 1-1. Parts of Java API to Ignore

Java API	Android Equivalent
Swing, Applets	Android's GUI, see [[Introduction: GUI]]
application entry point main()	See [[Android Lifecycle]]

Java API	Android Equivalent
J2ME/JavaME	Most of android.* replaces JavaME API
Servlets/JSP, J2EE/JavaEE	Designed for server-side use

Discussion

Here are some books and resources:

- O'Reilly's *Java in a Nutshell* is a good introduction for programmers, particularly those immigrating from C/C++. This book has grown from an acorn to a coconut in size, to keep up with the growth of Java SE over its lifetime.
- *Head First Java* provides a great visual-learner-oriented introduction to the language. O'Reilly.
- *Thinking In Java* (4th edition) by Bruce Eckel, Prentice-Hall.
- *Learning Java* Formerly titled *Exploring Java*, O'Reilly.
- *Great Java videos* provides a visual introduction to the language.
- *Java: The Good Parts* From the book's web site: "What if you could condense Java down to its very best features and build better applications with that simpler version? In this book, veteran Sun Labs engineer Jim Waldo reveals which parts of Java are most useful, and why those features make Java among the best programming languages available..."
- *Java Cookbook* (disclosure: I wrote this book) is regarded as a good second book for Java developers. It has entire chapters on Strings, Regular Expressions, Numbers, Dates & Time, Structuring Data, I/O and Directories, Internationalization, Threading and Networking, all of which apply to Android. It has a number of chapters that are specific to Swing and to some EE-based technologies.

What's needed is for somebody to write a book on *Android for non-Java Programmers* that would include just exactly the right parts of standard Java language and API along with all the Android stuff. Available now in three volumes, ships in its own cool retro wooden case... :-).

See Also

This book's editor maintains a list of Java resources online at <http://www.darwinsys.com/java/>.

O'Reilly has many of the best Java books around; there's a complete list at <http://oreilly.com/pub/topic/java>.

1.3 Hello World - Command Line

Ian Darwin

Problem

You want to create a new Android project without using the Eclipse ADT plug-in.

Solution

Use the Android Development Kit tool `android` with the `create project` argument and some additional arguments to configure your project.

Discussion

In addition to being the name of the platform, Android is also the name of a command-line tool for creating, updating and managing projects. You can either navigate into the `android-sdk-xxx` directory, or you can set your `PATH` variable to include its tools sub-directory.

Then, to create a new project, give the command "`android create project`" with some arguments. Here is an example run under MS-DOS:

Example 1-1.

```
C:\Documents and Settings\Ian\My Documents>PATH=%PATH%;"C:\Documents and Settings\Ian\My Documents\android-s
C:\Documents and Settings\Ian\My Documents>android create project --target 1 --package com.example.foo --nam
Created project directory: C:\Documents and Settings\Ian\My Documents\MyAndroid
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\src\com\example\foo
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\src\com\example\foo\FooActivity.java
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\res
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\bin
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\libs
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\res\values
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\res\values\strings.xml
Created directory C:\Documents and Settings\Ian\My Documents\MyAndroid\res\layout
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\res\layout\main.xml
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\AndroidManifest.xml
Added file C:\Documents and Settings\Ian\My Documents\MyAndroid\build.xml
```

```
C:\Documents and Settings\Ian\My Documents>
```

The list of arguments for the `create project` code follows:

Table 1-2. List of Create Project Arguments

Name	Meaning	Example
<code>--activity</code>	Name of your "main class" and default name for the generated .apk file	<code>--target HelloActivity</code>
<code>--name</code>	Name of the project and the generated .apk file	<code>--name MyProject</code>
<code>--package</code>	Name of Java package for your classes	<code>--package com.example.hello</code>

Name	Meaning	Example
--path	Path to create the project in (does not create a subdirectory under this)	--path /home/ian/workspace/MyProject (see above for Windows example)
--target	Level of the Android platform to target	--target 1

If it cannot complete the requested operation the `android` command presents a voluminous "command usage" message listing all the operations it can do and the arguments for them. If successful, the `android create project` command creates the following files and directories.

Table 1-3. Artifacts Created by Create Project

Name	Meaning
AndroidManifest.xml	Config file that tells Android about your project
bin	generated binaries (compiled class files)
build.properties	Editable properties file
build.xml	Standard Ant build control file
default.properties	
gen	Generated stuff
libs	Libraries, of course
res	important resource files (strings.xml, layouts, etc.)
src	source code for your application
src/packageName/ActivityName.java	source of "main" starting activity
test	copies of most of the above

It is normal and recommended Android practice to create your user interface in XML using the layout file created under `res/layout`, but it is certainly possible to write all the code in Java. To keep this example self-contained, we'll do it the "wrong" way for now. Use your favorite text editor to replace the contents of the file `HelloWorld.java` with the following contents:

Example 1-2.

```
public class Hello extends Activity {

    /**
     * This method gets invoked when the activity is instantiated in
     * response to e.g., you clicked on the app's Icon in the Home Screen.
     */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Create a TextView for the current Activity
        TextView view = new TextView(this);
        // Make it say something
    }
}
```

```
        view.setText("Hello World");
        // Put this newly-created view into the Activity
        // sort of like JFrame.setContentPane()
        setContentView(tv);
    }
}
```

Assuming you have the Apache Software Foundation [Ant Build Tool](#) installed, you can now (in a command-line window) change directory into the project directory (`...MyDocuments\MyAndroid` in the above example) and issue the command:

```
ant debug
```

This will create an archive file named e.g., `MyAndroid.apk` (apk standing for Android Package) in the `bin` directory.

Assuming you have either the Emulator running, or your device plugged in and recognized via USB, you can then do

```
adb -e install -r bin/MyAndroid.apk
```

The `-e` flag is for the emulator; use `-d` for a real device.

If you are handy with shell scripts or batch files, you'll want to create one called, say, `download`, to avoid typing the `adb` invocation on every build cycle.

You will probably find it most convenient to create an icon for your app on the home screen of the device or emulator; this icon will survive multiple "install -r" cycles so it's the easiest way to test running your application.

See Also

[Recipe 1.4](#). The blog *"a little madness" has a more detailed formulation*. The *official Android reference site has a page on developing without Eclipse*.

1.4 Hello World - Eclipse Version

Ian Darwin

Problem

You want to use Eclipse to develop your Android application.

Solution

Install [Eclipse](#), the [Android SDK](#) and the [ADT plug-in](#). Create your project and start writing your app. Build it, and test it under the Emulator, from within Eclipse.

Discussion

Once you have these items installed, you are ready to begin:

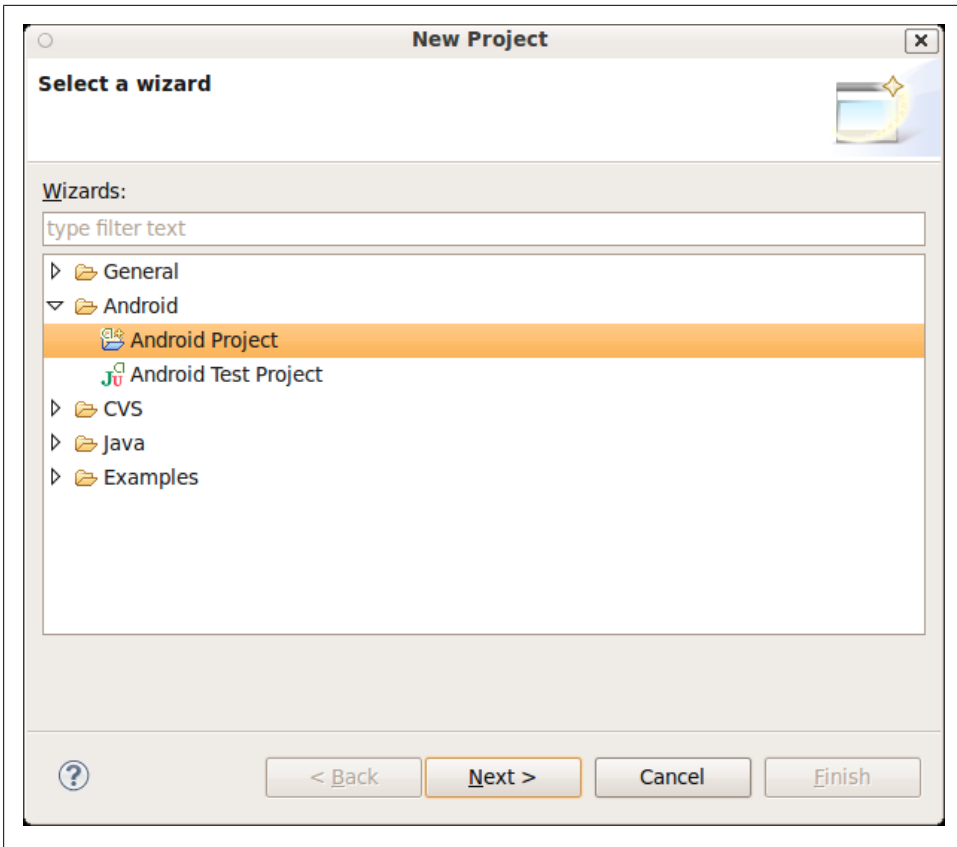


Figure 1-1.

- [Eclipse IDE](#)
- The [Android SDK](#)
- The [ADT plug-in](#)

To get started, create a new project from the **File->New** menu.

Click Next. Give your new project a name, and select an SDK version to target. 1.5 gives you almost all the devices in use today; 2.1 or 2.2 gives you the latest features. You decide.

This figure shows the project structure expanded in the Project Panel at the right. It also shows the extent to which you can use Eclipse Auto-completion within Android - I added the 'gravity' attribute for the label, and Eclipse is offering a full list of possible attribute values. I chose center-horizontal, so the label should be centered when we get the application running.

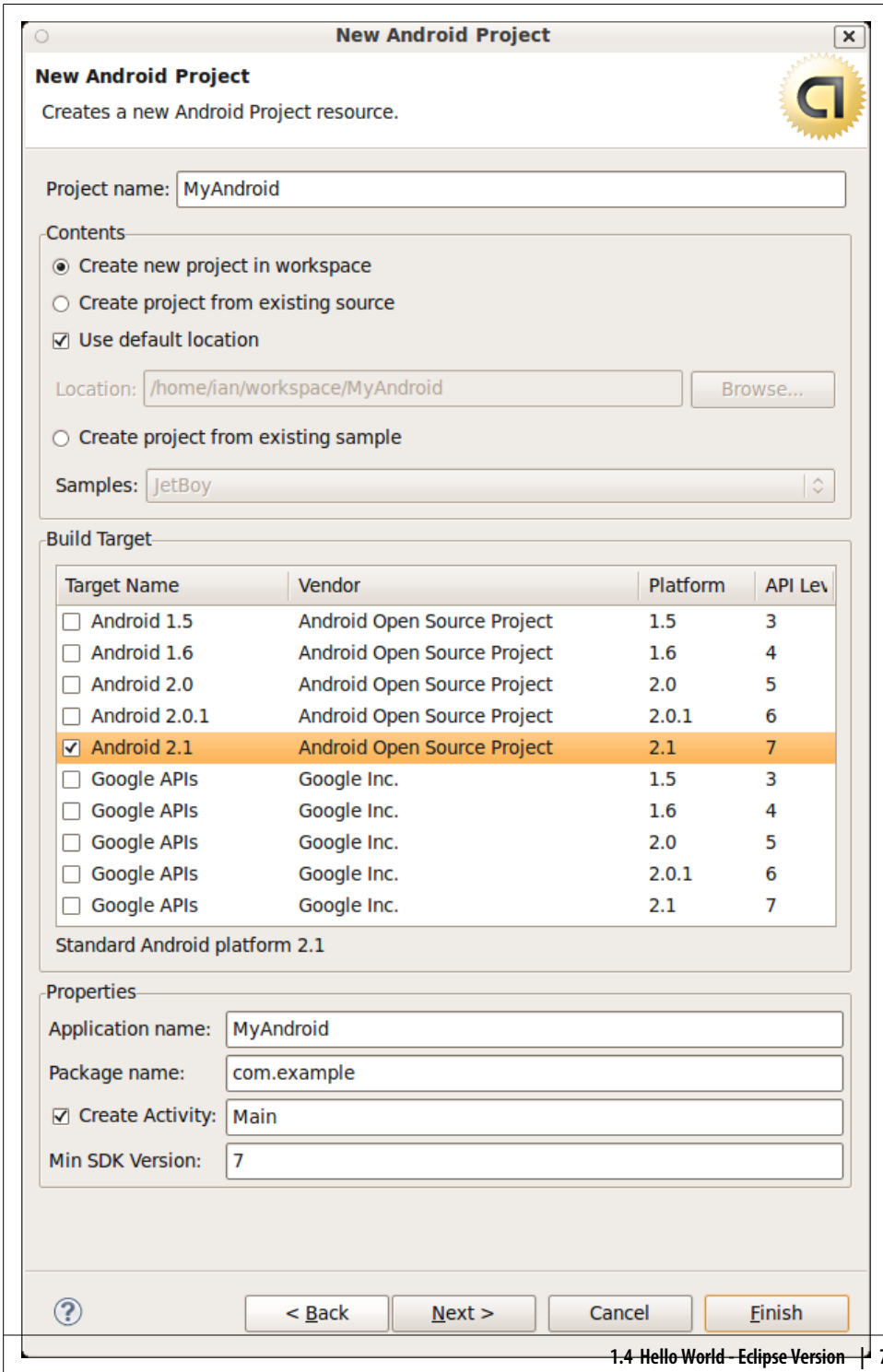


Figure 1-2.

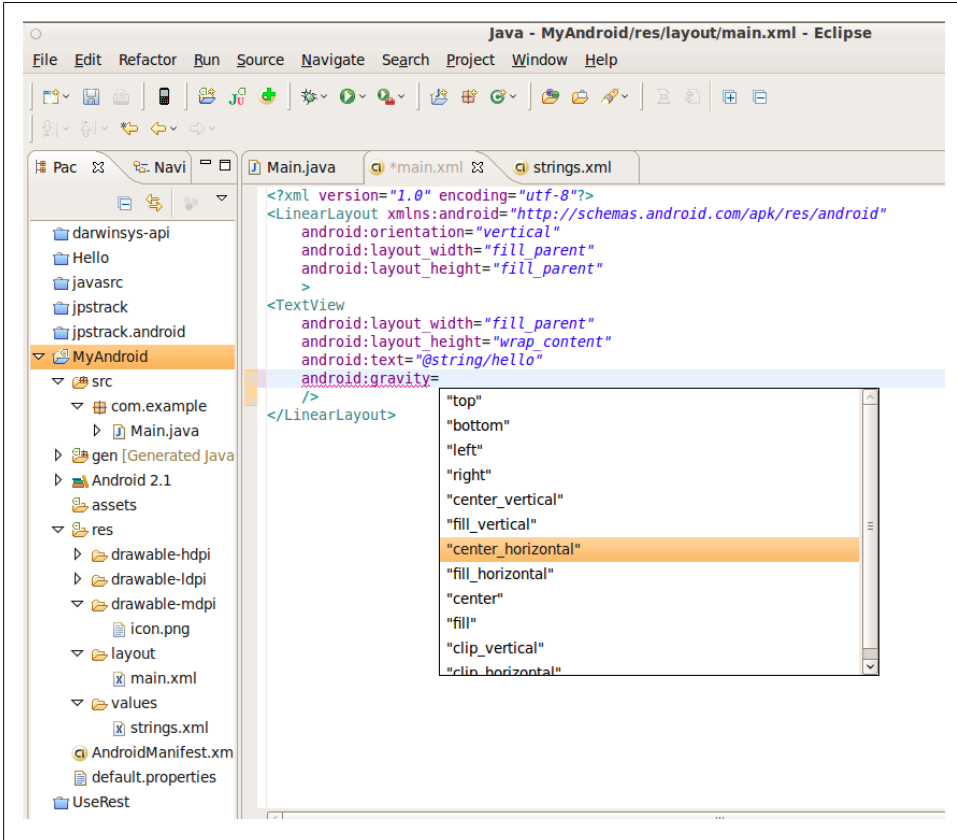


Figure 1-3.

In fact, if you set gravity to `center_vertical` on the `LinearLayout` and set it to `center_horizontal` on the `TextView`, the text will be centered both vertically and horizontally. Here's the version of the layout file `main.xml` which achieves this:

Example 1-3.

```
<?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"
    android:gravity="center_vertical"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        android:gravity="center_horizontal"
```

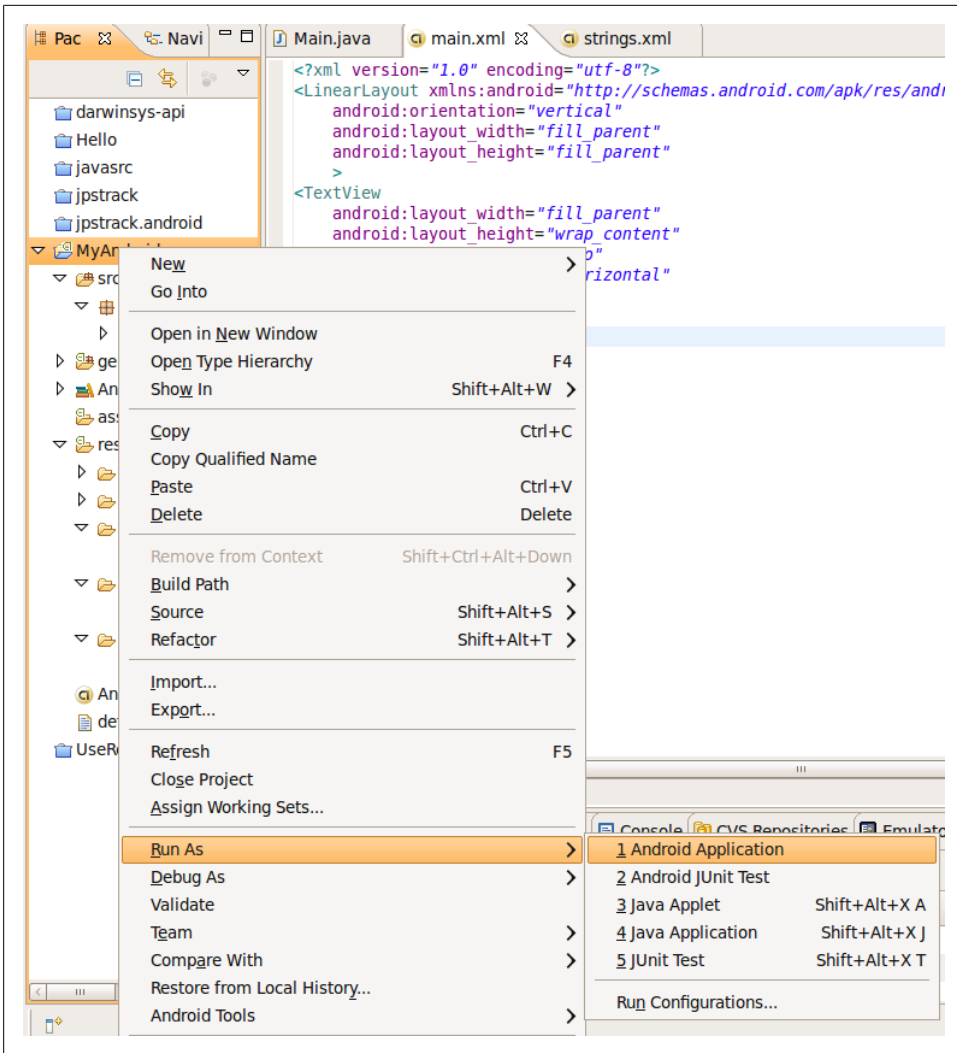


Figure 1-4.

```

/>
</LinearLayout>

```

As always, Eclipse generates a compiled version whenever you save a source file. Also, in an Android project, it also runs an Ant Build to create the compiled, packaged APK that is ready to run. So you only need to run it. Right click on the project itself, and do Run As -> Android Project.

This will start the Android Emulator if it's not already running. The emulator will start with the word `Android` in typewriter text, then switch to the fancier Android Font with

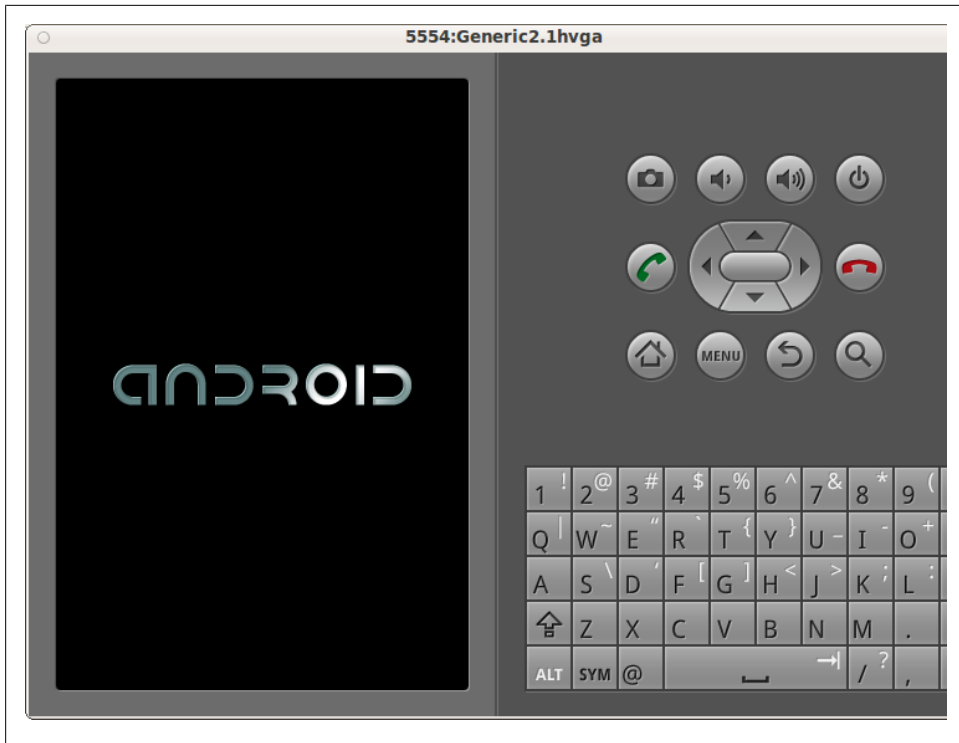


Figure 1-5.

a moving white patch over blue lettering - remember the Microsoft Windows'95 start-up?

After a little longer, your application should start up (and here we'll only show the screen shot of the application itself, since the rest of the Emulator view is redundant).

See Also

[Recipe 1.3](#)

1.5 Set Up an Android Virtual Device for Apps Testing

Daniel Fowler

Problem

Successful Apps must run on a wide range of Android devices and versions.

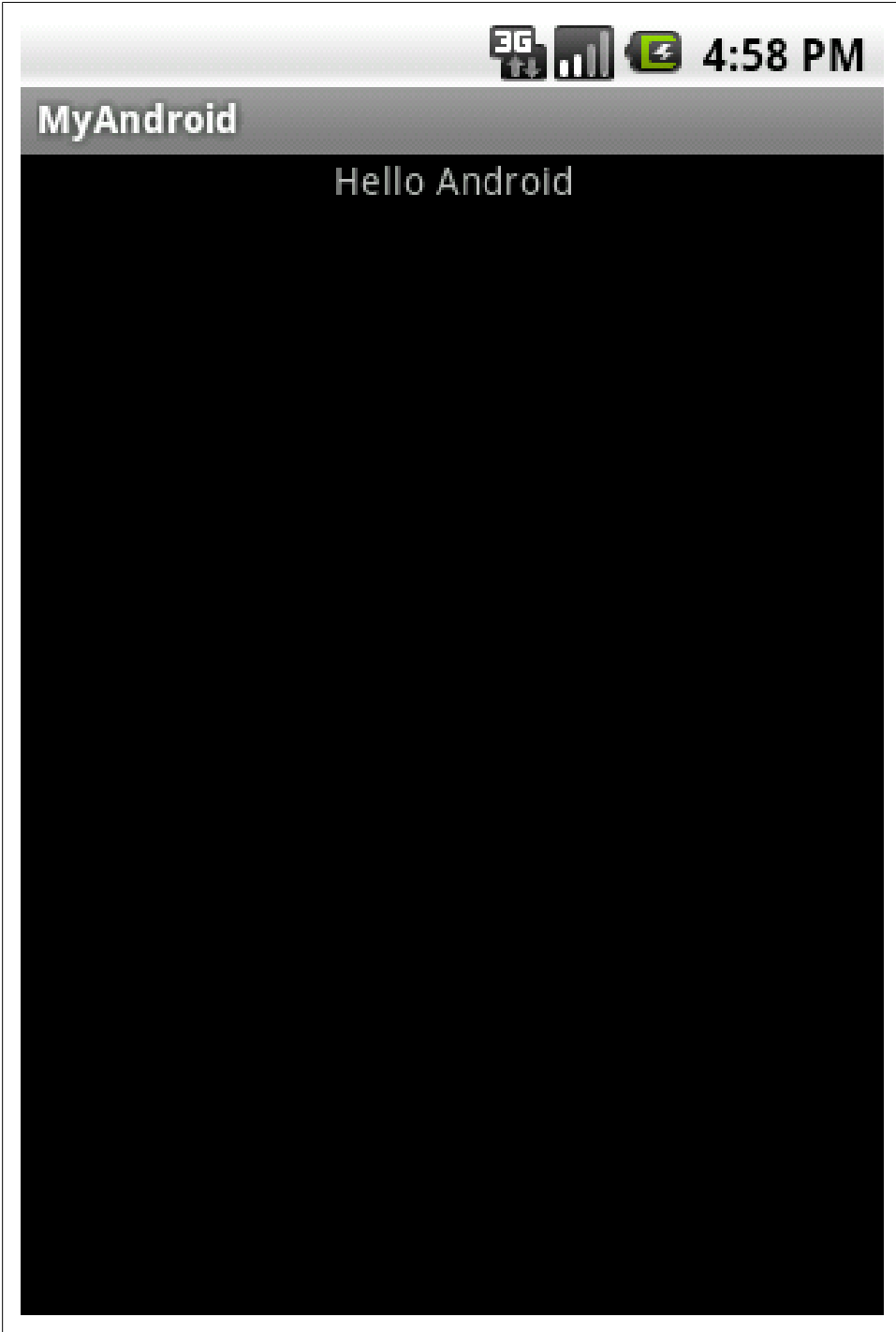


Figure 1-6.

Solution

Use the Android SDK's device emulation toolkit to configure combinations of devices and operating systems. Testing on various combinations reduces issues related to hardware differences in devices.

Discussion

Android devices are manufactured to cover a wide market, from low cost to high specification and high value. Android has also been in the marketplace for more than a couple of years. For these reasons there are a wide range of devices with a wide range of hardware options and operating system versions being used. A successful Application (App) will be one that can run on such a range of devices. An App developer will only be able to test on a very small range of physical devices. Fortunately a developer's confidence in their App can be boosted by using an Android Virtual Device (AVD).

A compiled App can be tested on a physical device or a virtual device. An AVD is an emulation of an Android platform on a host machine, usually the development machine. AVDs simplify testing for these reasons:

- Multiple AVD configurations can be created to test an App on different versions of Android.
- Different (emulated) hardware configurations can be used, for example GPS or no GPS.
- An AVD is automatically launched and your compiled App is installed on to it when the 'Run' button is pressed in Eclipse.
- You can test your App on many more combinations of Android version and hardware versions than physical devices you possess.
- Testing on AVDs greatly reduces the amount of testing required on physical devices.
- AVDs can be used alongside a physical device.
- You don't need to handicap your physical device to induce error conditions, e.g. testing on a device with no SD card, just set up an AVD with no SD card.
- An AVD can simulate network events without the costs involved in using a physical device, e.g. simulate phone calls or send an SMS between two AVDs.
- Simulate GPS data from an AVD from different physical locations without moving from your desk.
- When App users report bugs you can try and mimic their hardware configurations using AVDs.

Please note that on older development machines and when emulating larger Android devices the performance of an AVD will be less than that of a physical device.

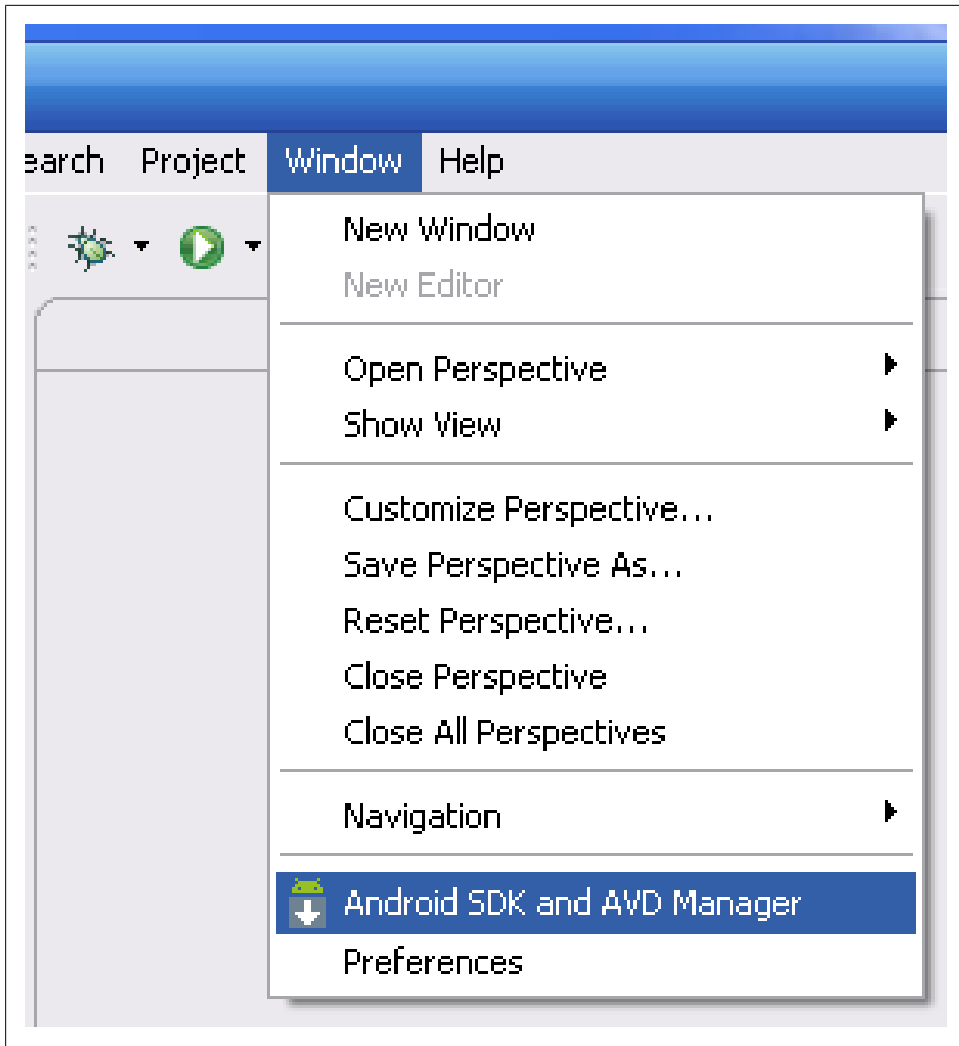


Figure 1-7.

An AVD can be configured using the 'SDK Manager' program (opened directly from the file system or from within Eclipse). It is also possible to create AVDs from the command line.

To create an AVD with the 'SDK Manager' load the program. When using Eclipse select 'Window' from the menu bar and then select 'Android SDK and AVD Manager'.

The program can also be started directly from the file system. For example in Windows open 'C:\Program Files\Android\android-sdk\SDK Manager.exe'. If started directly from the file system 'SDK Manager' will check for SDK updates, in which case press

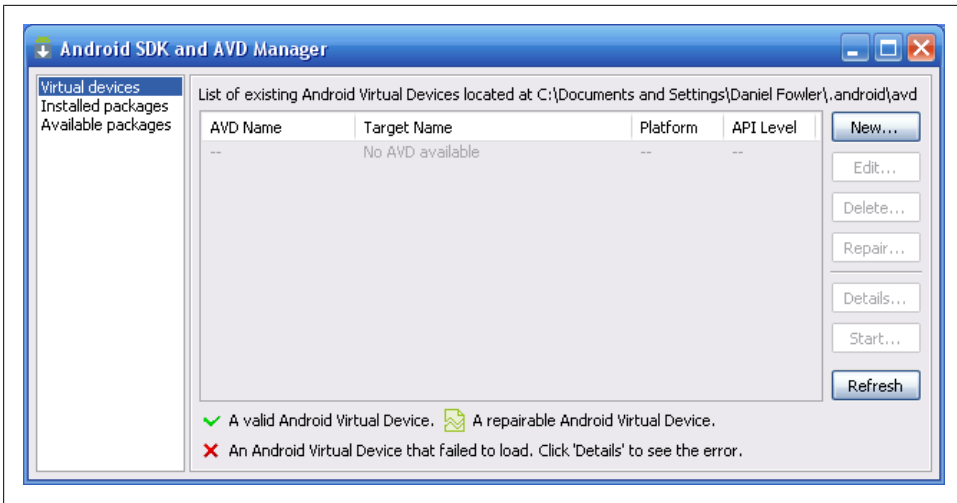


Figure 1-8.

'Cancel' to go to the main window, titled 'Android SDK and AVD Manager'. If opened from Eclipse the main Window will show without the check for updates to the SDK.

The left hand column of the main window will list 'Virtual Devices', 'Installed packages' and 'Available packages'. 'Virtual Devices' should already be selected, if not select 'Virtual Devices', any existing defined AVDs will be listed in the right hand table. If the Android SDK has just been installed no AVDs may be listed.

To create an AVD select the 'New' button. The 'Create new Android Virtual Device (AVD)' window will load.

The following fields are used to define an AVD:

Name

Give a name to the new Android device that is to be emulated. Make the name descriptive, for example if emulating a device with a version 2.1 operating system and medium resolution screen (HVGA) a name such as Android-v2.1-HVGA is better than AndroidDevice.

Target

This is the version of the Android operating system than will be running on the emulated device, as an example for a device running version 2.1 this will be set to "Android 2.1-update1 - API Level 7".

SD Card

Here you specify the size of the devices emulated Secure Digital (SD) card, or select an existing SD card image (allowing the ability to share SD card data amongst different AVD emulations). To specify a new SD card enter the size in MiBs for the card. Re-

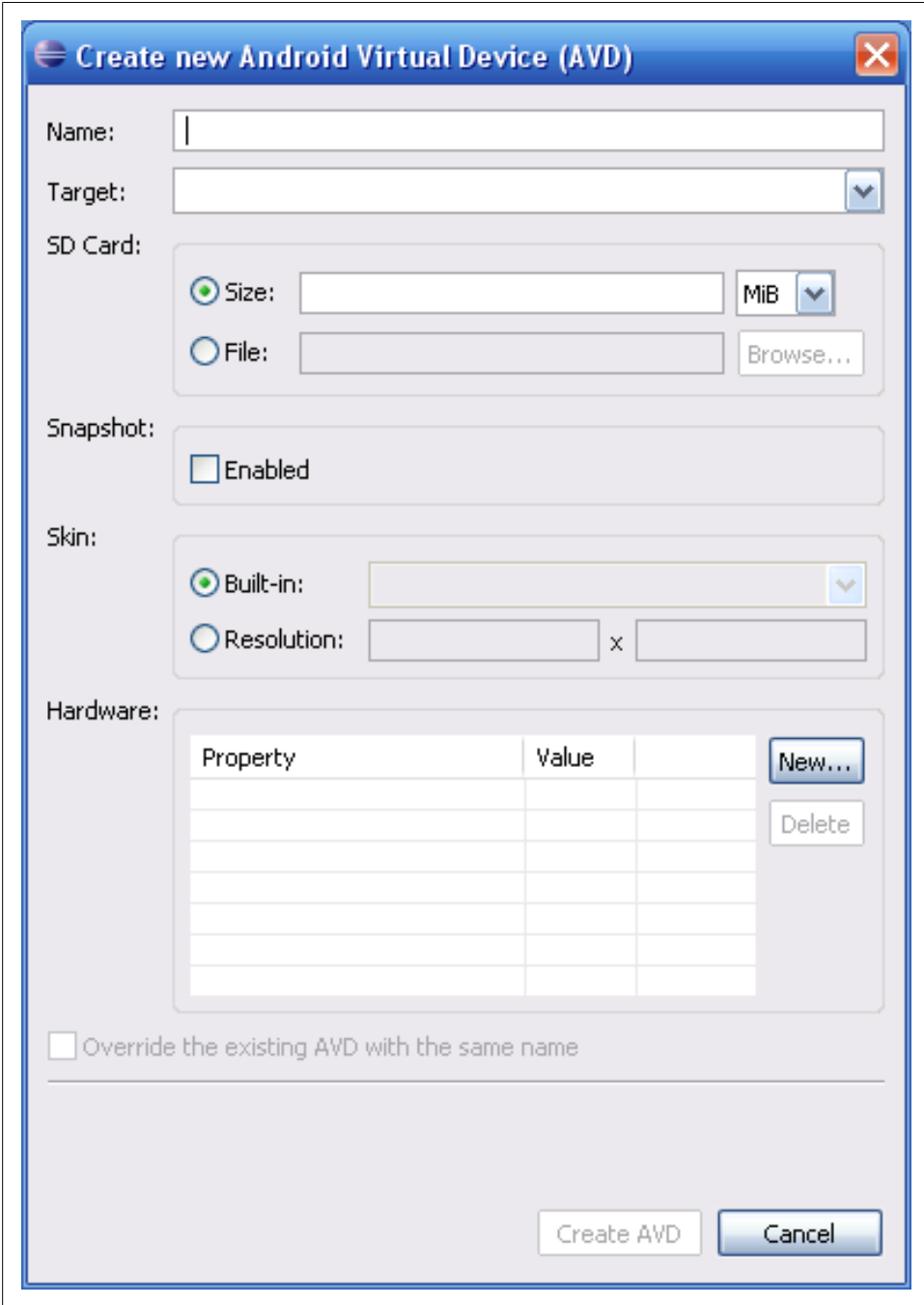


Figure 1-9.

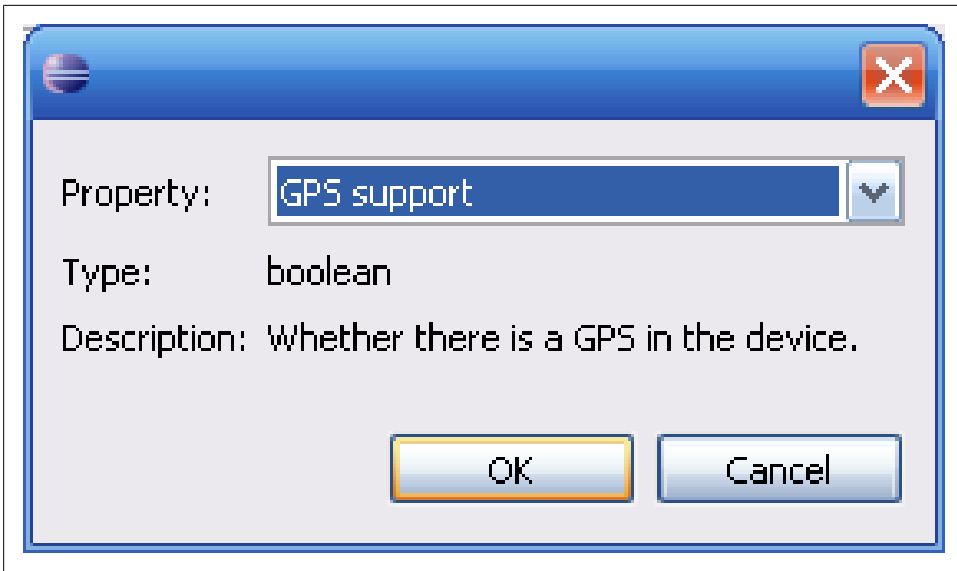


Figure 1-10.

member that the bigger the number the bigger the file created on the host computer system to mimic the SD card. Alternatively select the 'File' option and browse to an existing SD card image (on a Windows machine the 'sdcard.img' files will be found in the sub-folders of the 'avd' directory under the '.android' directory in the logged on users folder).

Snapshot

Check the 'Enabled' box if you want the runtime state of the emulated device to persist between sessions, useful if a long running series of tests are being performed and when the AVD is closed you do not want to have to start the tests from the beginning. It also speeds up the start up time of an AVD.

Skin

Here you select the screen size for the device, a list of common screen sizes is presented e.g. HVGA, QVGA etc. The list will vary depending on the operating system version. Alternatively a custom resolution can be entered.

Hardware

The table under the hardware option allows the AVD to be configured with or without certain hardware features. To change features first add them to the table using the 'New' button (a couple of features will be added and default automatically based on the 'Target' selected). A dialog will open to allow the selection of a hardware property.

For example select 'GPS support' then 'OK'. Select 'yes' next to 'GPS support in the table and change it to 'no'. The AVD will not support GPS.

The AVD supported properties are:

Table 1-4. AVD Supported Properties

Description
Camera support
Boolean
yes or no
This determines if the AVD supports the detection of a camera.
Max VM application heap size
integer
megabyte value
The maximum size of the heap an App might allocate before being shut down by the system.
Abstracted LCD density
integer
120/160/240/320
Approximate density (dots per inch) of the AVD screen, 120 is low density, 160 for standard or normal density, 240 is high density and 320 is extra high density.
Cache partition size
integer megabytes
xxxMB
This sets the size of cache used by the browser.
SD Card support
Boolean
yes or no
Support for a SD card.
Cache partition support
Boolean
yes or no
This determines whether a browser uses a cache.
Keyboard support
Boolean
yes or no
This controls emulation of a physical keyboard (as opposed to an on screen one).
Audio playback support
Boolean
yes or no
Support for audio play back.

Description

Keyboard lid support

Boolean

yes or no

Can the emulated keyboard be opened and closed.

Audio recording support

Boolean

yes or no

Support for recording audio.

DPad support

Boolean

yes or no

This indicates emulation of a directional pad.

Maximum vertical camera pixels

integer

pixels height

This determines the height of photos taken with the camera.

Accelerometer

Boolean

yes or no

Can a tilt and movement device be detected.

GPS support

Boolean

yes or no

Can Global Positioning System data be provided.

Device ram size

integer

megabytes

This determines size of the AVD's memory.

Touch screen support

Boolean

yes or no

This determines if the AVD supports operation via the screen.

Proximity support

Boolean

yes or no

Description

Support for a proximity sensor.

Battery support

Boolean

yes or no

Support for simulated battery power.

GSM modem support

Boolean

yes or no

This determines emulation of telephony abilities.

Trackball support

Boolean

yes or no

Support for a trackball.

Maximum horizontal camera pixels

integer

pixels width

This determines the width of photos taken with the camera.

When the required fields have been defined the 'Create AVD' button is pressed to generate the AVD. The AVD will now be listed on the 'Android SDK and AVD Manager' window.

The AVD is ready to be launched using the 'Start...' button. It is also ready to be selected in a project configuration to test an App under development. When the 'Start...' button is pressed the 'Launch Options' window is shown.

The options at launch are:

Scale the display to real size

On larger computer monitors you will not normally need to change the AVD scale. The dpi of the Android screens is greater than the standard dpi on computer monitors; therefore the AVD screen will appear larger than the physical device. If necessary this can be scaled back to save screen space. Use this option to get the AVD to display at an approximate real size on the computer monitor. The values need to be set so that the AVD screen and keyboard is not too small to be used.

Wipe user data

When the AVD is started the user data file is reset, any user data generated from previous runs of the AVD is lost.

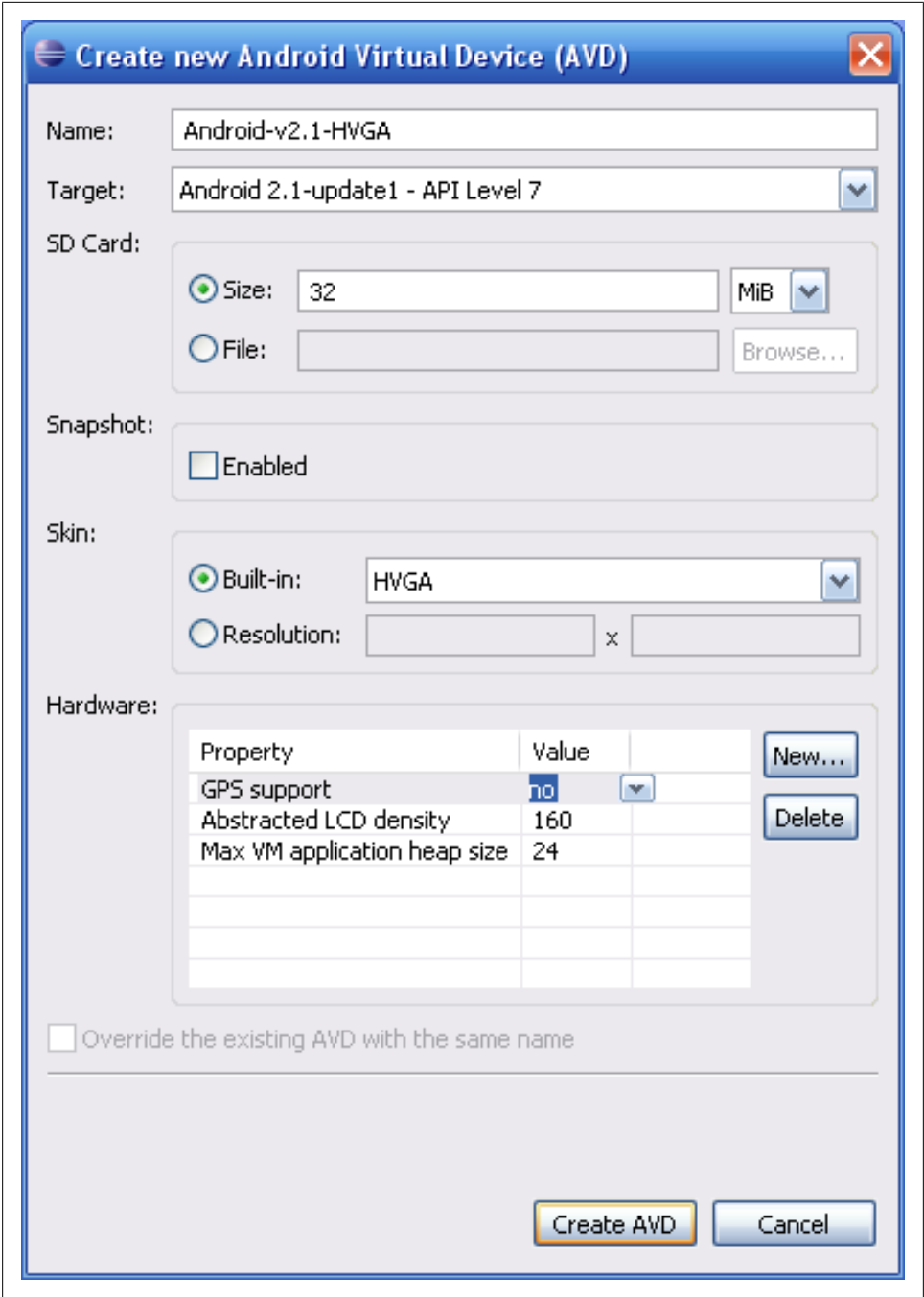


Figure 1-11.

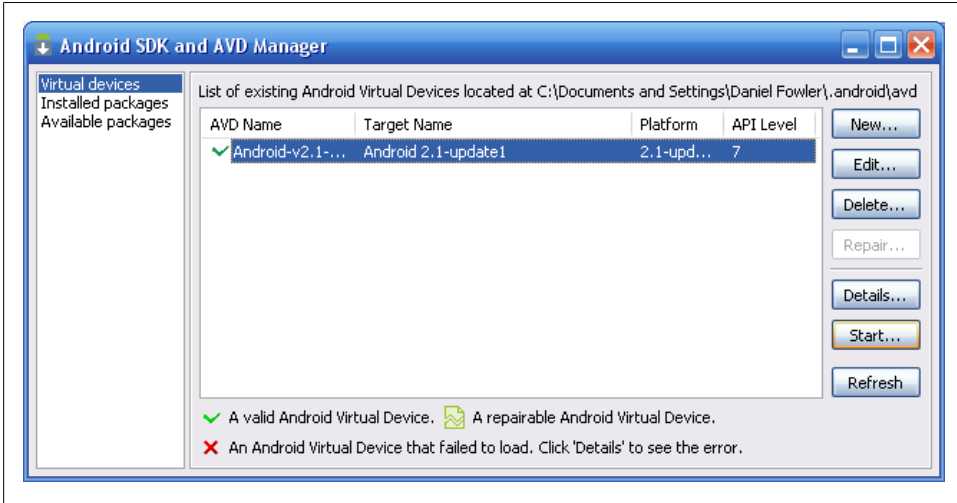


Figure 1-12.

Launch from snapshot

If 'Snapshot' has been 'Enabled' for an AVD then after it has been first launched subsequent launches are quicker. The AVD is loaded from a snapshot and the Android operating system does not need to start up again. Although when the AVD is closed the shutdown is longer because the snapshot has to be written to disk.

Save to snapshot

When the AVD is closed the current state is saved for quicker launching next time; although close down is slower as the snapshot is written to disk. Once you have a snapshot you can uncheck this option so closing an AVD is quick as well, though any changes since the last snapshot will be lost.

Use the 'Launch' button to start the AVD. Once loaded it can be used like any other Android device and driven from the keyboard and mouse of the host computer.

Error Message on Windows when Launching

When trying to launch an AVD on a Windows installation an error with the description beginning *invalid command-line parameter* may occur.

To fix this problem change the path to the Android SDK directory so that it does not contain any spaces. The default installation path for the SDK is in *C:\Program Files\Android*. The space in *Program Files* needs to be removed. To do this and maintain a valid directory name *Program Files* needs to be converted to its Microsoft DOS format (also referred to as 8.3 format). This is usually the first six letters in upper case followed by a tilde and the number 1, i.e. *PROGRA~1*. If other directories start with *Program* followed by a space then the number may need to be increased. To see the DOS format for the *Program Files* directory on your machine open a *Command Prompt* (via *Start->All Programs->Accessories*). Change to root

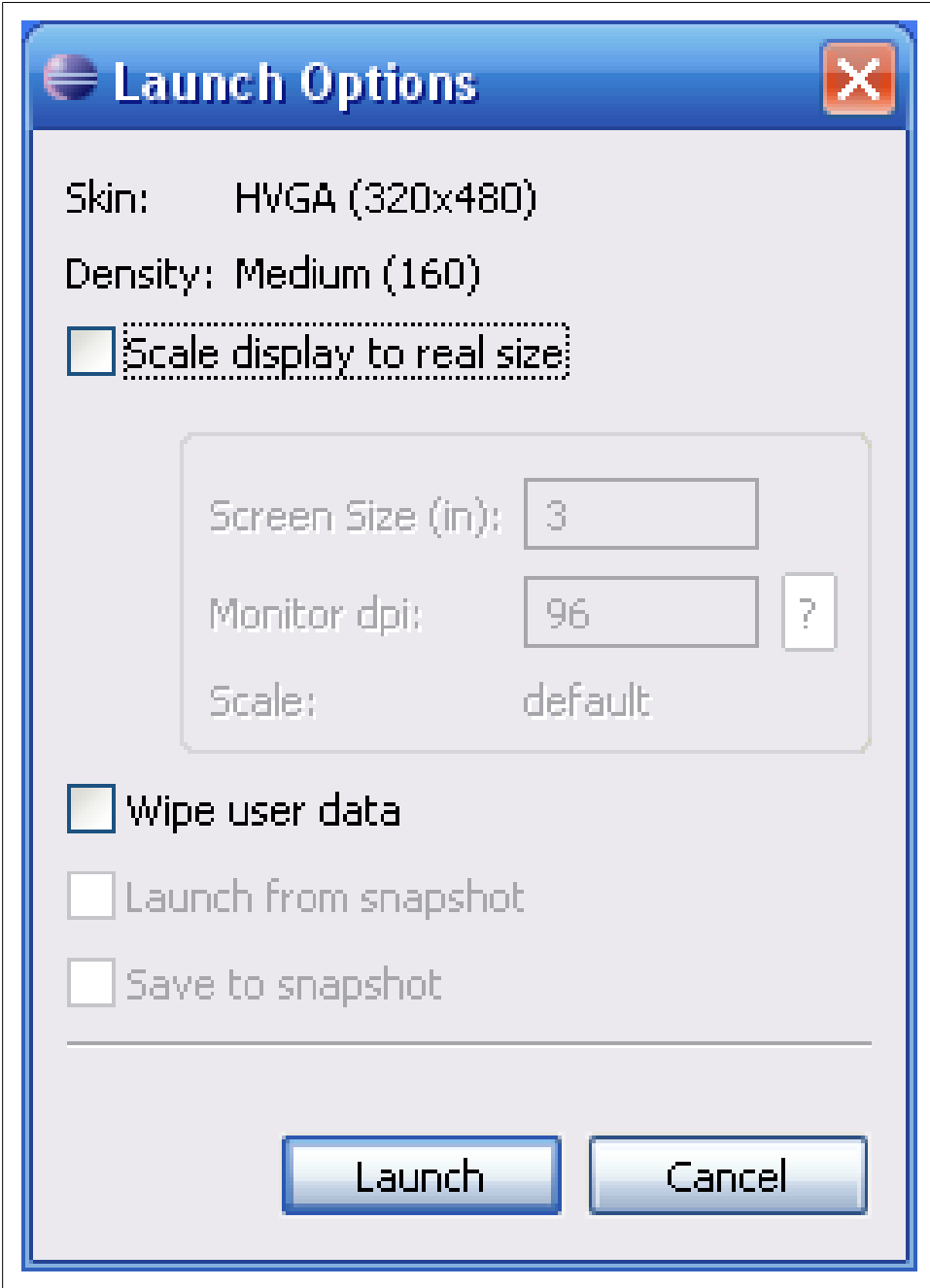


Figure 1-13.



Figure 1-14.

(type `cd\` and press Enter) and run `dir/x`, the directories DOS name will be displayed next to its full name.

In Eclipse use the *Windows->Preferences* menu option and select *Android*, in the *SDK Location* field change *Program Files* to its DOS version.

See Also

<http://d.android.com/guide/developing/devices/emulator.html>

1.6 Set Up an IDE on Windows to Develop for Android

Daniel Fowler

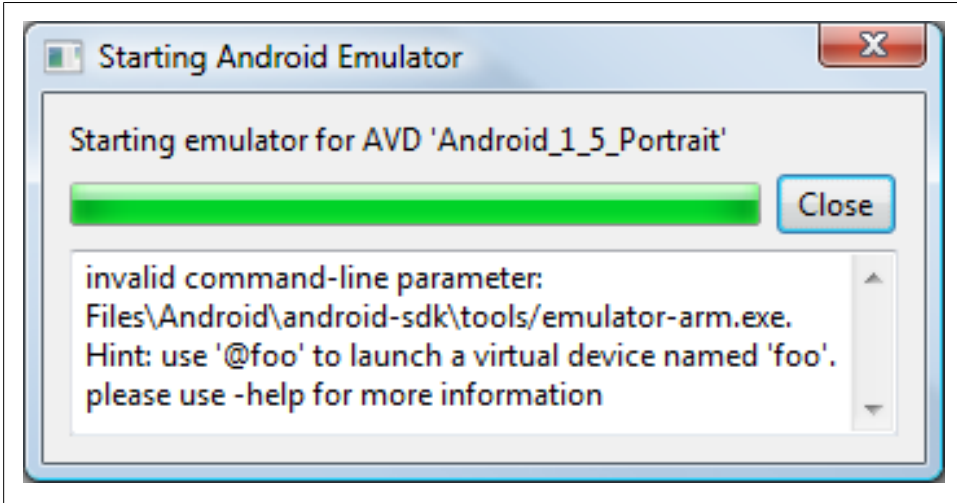


Figure 1-15.

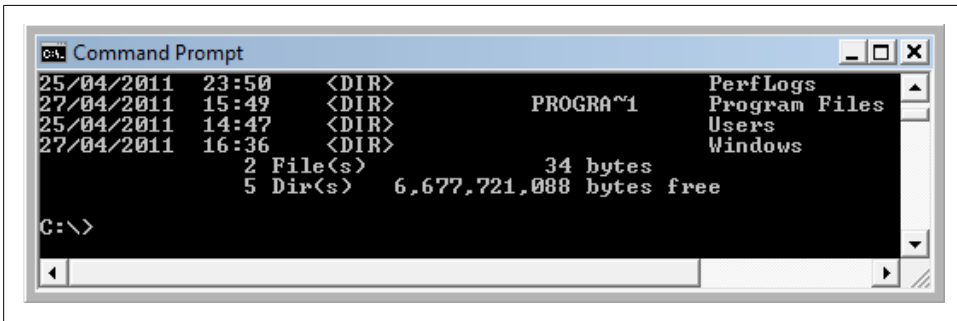


Figure 1-16.

Problem

Some owners of Android mobile phones, tablets and other devices may want to try and develop their own Android applications. They may use a Windows PC, a concise guide to setting up an IDE for that platform is useful.

Solution

The use of the Eclipse IDE is recommended when developing Android Apps. Configuring Eclipse on Windows is not a single shot install, several stages need to be completed. A single concise guide to setting up the IDE on Windows is helpful, particularly for those with limited software development experience.

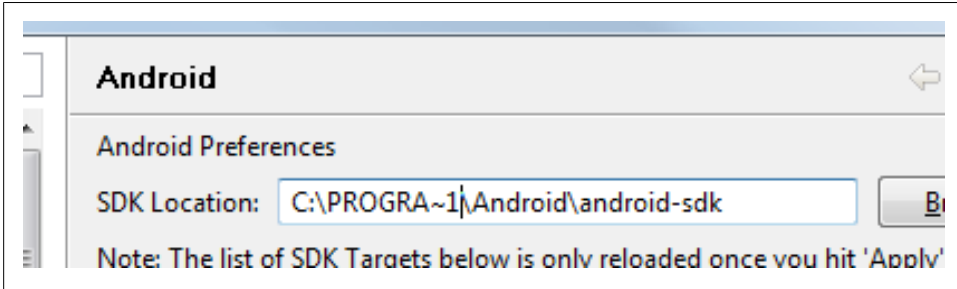


Figure 1-17.

Discussion

To develop applications for Android it is recommended that Eclipse is used. Eclipse provides an Integrated Development Environment (IDE) for Java. An Android Development Tools plug-in is available to enhance Eclipse. The ADT plug-in uses the Android Software Development Kit which provides essential tools to develop Android software. To set-up a development system you will need to download and install:

1. Java Standard Edition Development Kit
2. Eclipse for Java Development
3. Android Software Development Kit
4. Android Development Tools Plug-in (from within Eclipse)

These stages in more detail for a PC running Windows (tested on 32 bit XP and Vista):

1. Install JDK (Java Development Kit)

Go to the Java download page at:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Select the 'Java' icon to access the JDK downloads:

The list of JDK downloads will be shown. Click the 'Accept License Agreement' radio button, otherwise you cannot download the JDK.

Download and run the file *jdk-7-windows-i586.exe* (or *jdk-7-windows-x64.exe* for 64 bit Windows). You may need to select the location of the download site. Accept any security warnings that appear but only if you are downloading from the official Java download web page.

When the download has completed and is run you will need to go through the install screens clicking *Next* until the JDK installer has finished. You should not need to change any options presented. When the JDK installer has completed click the *Finish* button. A product registration web page may load, this can be closed or you can choose to register your installation.



Figure 1-18.

2. Install Eclipse for Java Development

Got to the Eclipse Downloads web page at:

<http://www.eclipse.org/downloads/>

Windows needs to be selected in the *Packages* dropdown, select the relevant *Eclipse IDE for Java Developers* download link.

Download and open the zip file. In the file there will be an *eclipse* directory containing several files and sub-directories. Copy the eclipse directory and all its contents as it comes. The usual place to copy the files to is either the root of the C drive or under C: \Program Files, you may need to select continue when Windows asks permission for the copy.

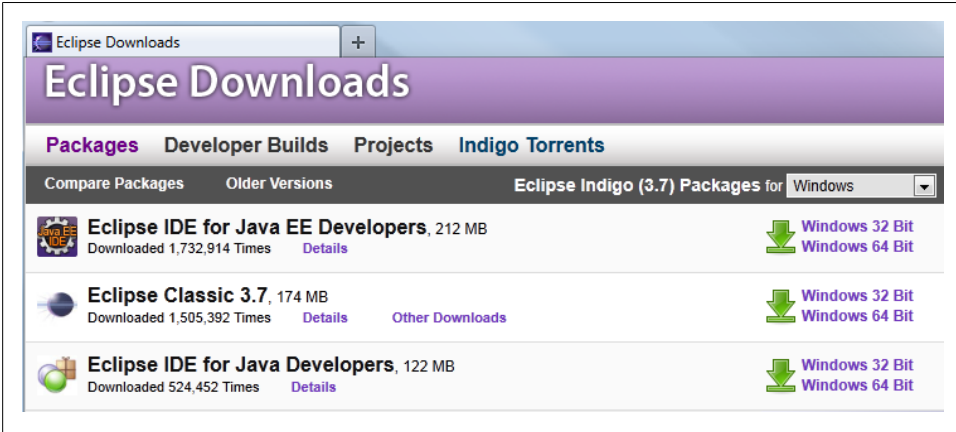


Figure 1-19.

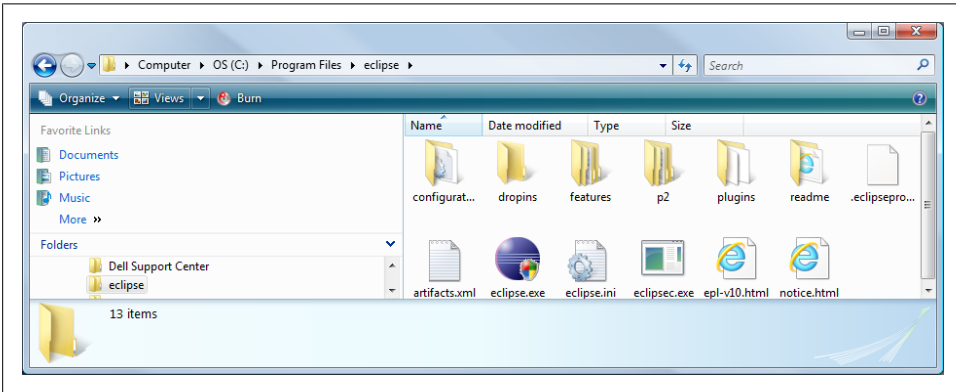


Figure 1-20.

Make a shortcut to *eclipse.exe*.

Run Eclipse so that it sets up a workspace and to check that both Java and Eclipse installed correctly. When running Eclipse a security warning may be displayed, select *Run* to continue. Accept the default workspace location or use a different directory.

3. Install Android SDK (Software Development Kit)

Go to the Android Software Development Kit download page at:

<http://developer.android.com/sdk/index.html>

Choose the Windows EXE package (*installer_r12-windows.exe*) and select *Run*. Accept the security warning only if you are downloading from the official Android SDK web site. The Android SDK Tools installer will show some screens, select the *Next* button



Figure 1-21.

on each screen, you should not need to change any options. You may see a *Java SE Development Kit (JDK) not found* screen. Although the JDK has been installed this is a

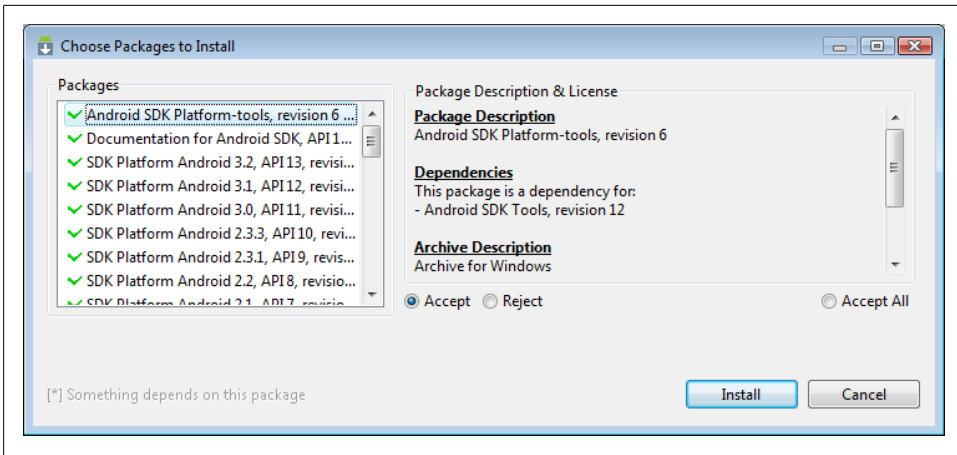


Figure 1-22.

bug in the Android SDK installer. The work around is to select the *Back* button and then the *Next* button again.

When the *Install* button is pressed a progress screen will briefly display while the Android files are copied, usually to `C:\Program Files\Android\android-sdk` unless the install location was changed. Some users have experienced minor issues with the SDK installed under `C:\Program Files` because of the space in the directory name. There is a work around described below, alternatively install to `C:\Android\android-sdk`. Click the final *Next* button and the *Finish* button at the end of the installation. If you left the *Start SDK Manager* checkbox ticked then the SDK Manager will run. Otherwise select *SDK Manager* from the *Android SDK Tools* program group (Start->All Programs->Android SDK Tools->SDK Manager).

When the SDK Manager runs a progress dialog will be shown while the Android packages available to download are checked. Then a list of all available packages are shown with many pre-selected for download. You should not need to change the initial selection. Click *Install* and the selected packages will download and be configured for use. This may take a few minutes.

You may see a message box titled *ADB Restart*, if this is still the first run of SDK Manager you can select *No*. Select *Close* on the completed progress screen. Close SDK Manager by clicking the *X* button in the top corner of the window.

4. Android Development Tools (ADT) Plug-in

Installing the ADT Plug-in is done via Eclipse. To install the ADT Plug-in Eclipse must be run from the Administrator account. Use the shortcut created earlier or *eclipse.exe* from the eclipse folder. In either case bring up the context menu (usually right-click) and select *Run as administrator*, accept any security warnings. When Eclipse has loaded open the *Help* menu item and select *Install New Software...*

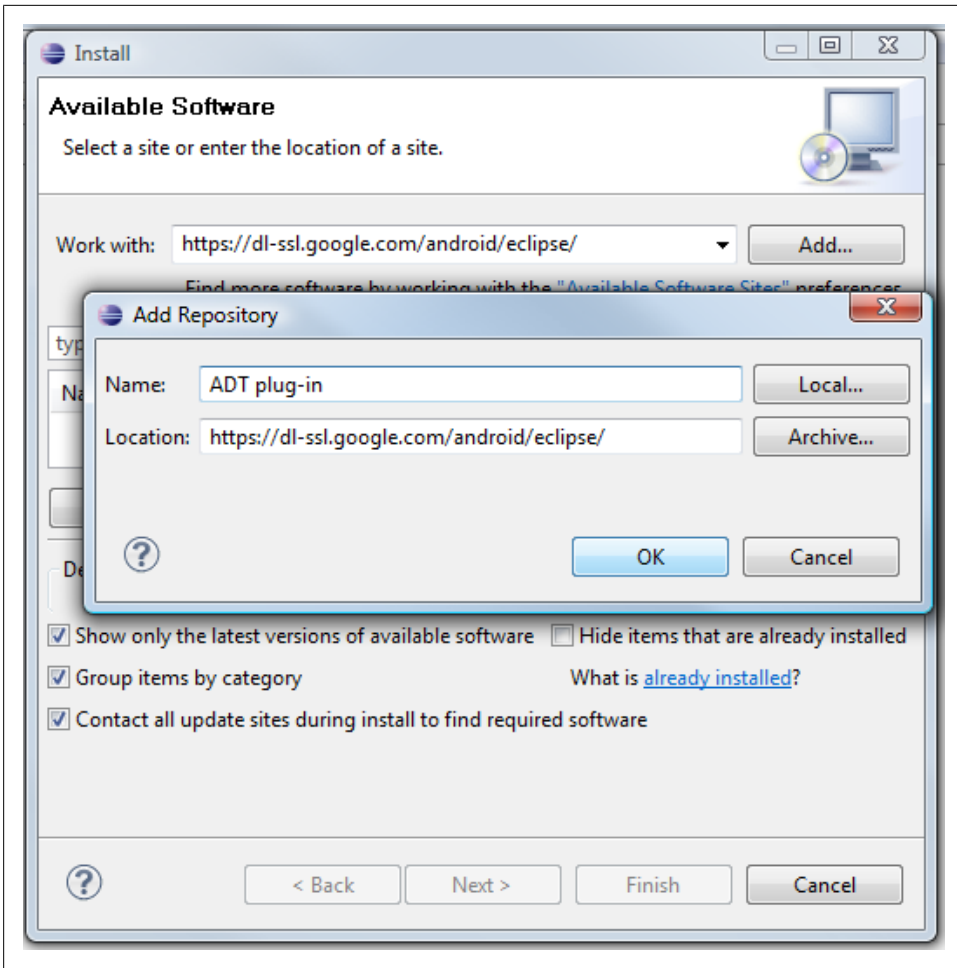


Figure 1-23.

On the *Install* screen enter the following address into the *Work with* box:

```
https://dl-ssl.google.com/android/eclipse/
```

Click the *Add* button. An *Add Repository* screen appears, in the *Name* box type something meaningful, such as *ADT plug-in* (the above address will be displayed in the *Location* box below).

Click the *OK* button. The screen will update after briefly showing *Pending* in the *Name* column of the table.

Check the box next to *Developer Tools*. Then select the *Next* button at the bottom of the screen.

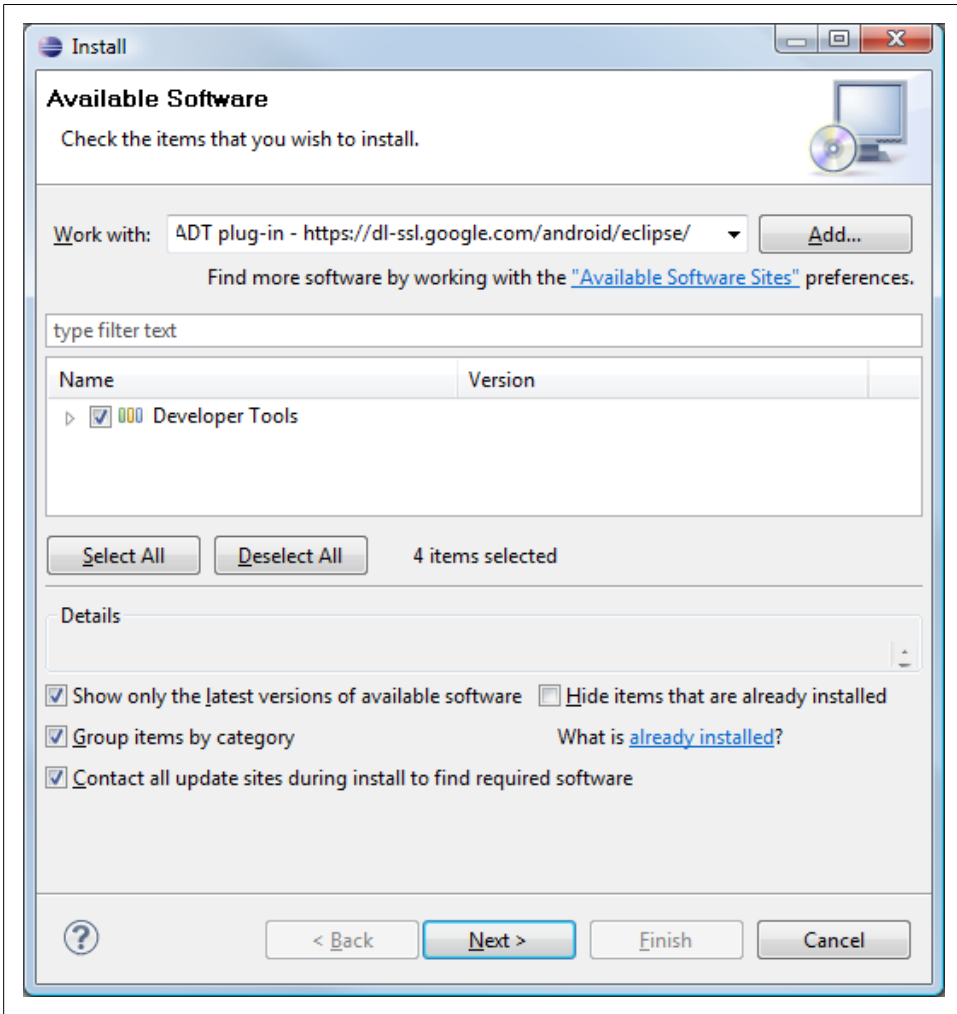


Figure 1-24.

A list of the items to be installed will be displayed. Select *Next* again.

A screen displays the licenses, ensure that each license has been accepted (select *I accept the terms of the license agreements* radio button). Then click the *Finish* button. A security warning will need to be accepted to complete the installation, select *OK* to this warning (the address entered above is a secure address).

Eclipse will ask you for a restart. Select the *Restart Now* button and Eclipse will close and reload.

Use the *Window* menu option in Eclipse and select *Preferences*. On the *Preferences* dialog select *Android*. A Google Android SDK usage monitoring question may appear. If you are happy with usage monitoring select the *Proceed* button, otherwise remove the tick on the check box and then select *Proceed*.

In the *SDK Location* box enter the location selected in step three (or use the *Browse* button to choose). If that location is `C:\Program Files\Android\android-sdk` use the Windows 8.3 format for the *Program Files* directory, i.e. set it to `C:\PROGRA~1\An-droid\android-sdk`. Select the *Apply* button and the list below the SDK Location will update. Select *OK* to close the screen.

Eclipse is now configured to build and debug Android Apps. Use the recipe [Recipe 1.5](#) to configure an Android Emulator; then try the [Recipe 1.4](#) recipe as a sanity check.

See Also

[Recipe 1.5](#)

[Recipe 1.4](#)

1.7 Android Lifecycle

Ian Darwin

Problem

Android apps do not have a "main" method; you need to learn how they get started and how they stop or get stopped.

Solution

The class `android.Activity` provides a number of well-defined life-cycle methods that are called when an application is started, suspended, restarted, etc., as well as a method you can call to mark an Activity as finished.

Discussion

Your Android application runs in its own Unix process, so in general it cannot directly affect any other running application. The Dalvik VM interfaces with the operating system to call you when your application starts, when the user switches to another application, and so on. There is a well-defined lifecycle for Android applications.

An Android application has three states it can be in:

1. Active - the app is visible to the user and is running;
2. Paused - the app is partly obscured and has lost the input focus.
3. Stopped - the app is completely hidden from view

Your app will be transitioned among these states by Android calling the following methods on the current Activity at the appropriate time:

Example 1-4.

```
void onCreate(Bundle savedInstanceState)
void onStart()
void onResume()
void onRestart()
void onPause()
void onStop()
void onDestroy()
```

For an application's first Activity, `onCreate()` is how you know that the application has been started. This is where you normally do constructor-like work such as setting up the "main window" with `setContentView()`, add listeners to buttons to do work (including starting additional Activities), and so on. This is the one method that even the simplest Android app needs.

You can see the effects of the various lifecycle methods by creating a dummy project in Eclipse and overriding all the methods with log "debug" statements.

1.8 Opening a Web Page, Phone Number or anything else with an Intent

Ian Darwin

Problem

The `Intent` mechanism is fundamental to Android; it allows one application to have some entity processed by another application without knowing or caring what that application is.

Solution

Invoke the `Intent` constructor; invoke `startActivity` on the constructed `Intent`.

Discussion

The `Intent` constructor takes two arguments, the action to take and the entity to act on. Think of the first as the verb and the second as the object of the verb. The most common action is `Intent.ACTION_VIEW`, for which the String representation is `android.intent.action.VIEW`. The second will typically be a URL or as Android likes it less precisely (more generally) a URI. URIs can be created using the static `parse()` method in the `URI` class. Assuming that the String variable `data` contains the location we want to view, the code to create an `Intent` for it might be something like the following:

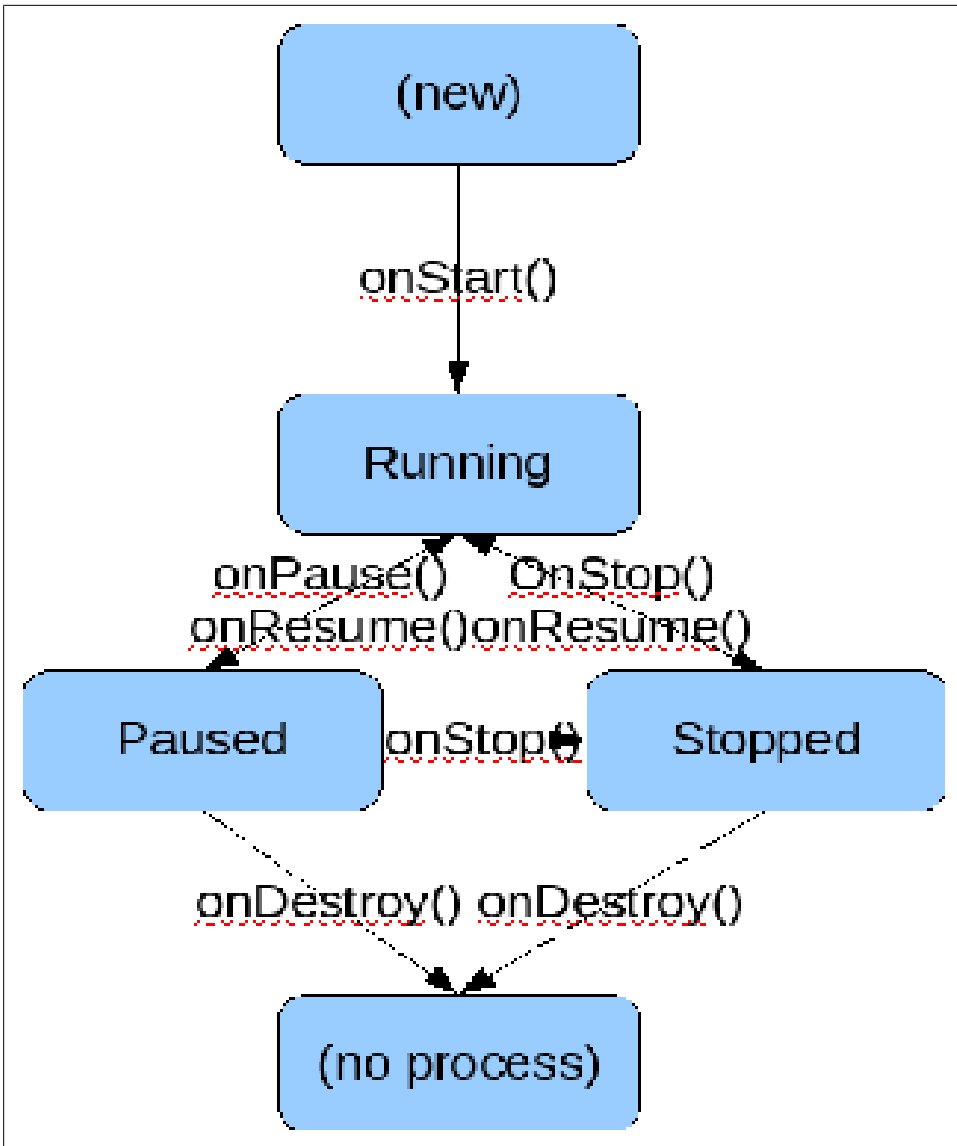


Figure 1-25.

Example 1-5.

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(data));
```

That's all! The beauty of Android is shown here - we don't know or care if `data` contains a web page URL with `http:`, a phone number with `tel:`, or even something we've never seen. As long as there is an application registered to process this type of intent, Android will find it for us, after we invoke it. How do we invoke the Intent? Remember that

Android will start a new Activity to run the intent. Assuming the code is in an Activity, just call the inherited `startIntent` method, e.g.,

Example 1-6.

```
startActivity(intent);
```

If all goes well, the user will see the web browser, phone dialer, maps application, or whatever.

XXX Discuss other actions such as `ACTION_OPEN`

However, if things fail, the user will not see anything. Why not? We basically told Android that we don't care whether the intent succeeds or fails. To get feedback, we have to call `startActivityForResult`:

Example 1-7.

```
startActivityForResult(intent, requestCode);
```

The `requestCode` is an arbitrary number used to keep track of multiple Intent requests; you should generally pick a unique number for each Intent you start, and keep track of these numbers to track the results later (if you only have one Intent whose results you care about, just use the number '1').

Just making this change will have no effect, however, unless we also override an important method in `Activity`, that is:

Example 1-8.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    // do something with the results...
}
```

It may be obvious, but is important to note, that you cannot know the result of an Intent until the entire application that was processing it is finished, which may be an arbitrary time later. However, the `onActivityResult` will eventually be called,

XXX Cover `resultCode`

XXX cover use of the passed intent - refer to recipes on passing Extra Data

Source Download URL

The source code for this example may be downloaded from this URL: <http://projects.darwinsys.com/IntentsDemo-src.zip>

1.9 Email Text From a View

Wagied Davids

Problem

Send an Email containing text or images from a View. The data to be emailed is passed as a parameter using an Intent.

Solution

File: AndroidManifest.xml

Example 1-9.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.examples"
  android:versionCode="1"
  android:versionName="1.0">
  <application
    android:icon="@drawable/icon"
    android:label="@string/app_name">
    <activity
      android:name=".Main"
      android:label="@string/app_name">
      <intent-filter>
        <action
          android:name="android.intent.action.MAIN" />
        <category
          android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>

    <!-- Required Permission -->
    <uses-permission
      android:name="android.permission.INTERNET" />
    <uses-permission
      android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
      android:name="android.permission.ACCESS_COARSE_LOCATION"></uses-permission>
    <uses-permission
      android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
  </application>
</manifest>
```

File: main.xml

Example 1-10.

```
<?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">

  <Button
```

```

        android:id="@+id/emailButton"
        android:text="Email Text!"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
</Button>

<TextView
    android:id="@+id/text_to_email"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/my_text" />

</LinearLayout>

```

File: strings.xml

Example 1-11.

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string
        name="hello">Hello World, Main!</string>
    <string
        name="app_name">EmailAndroid</string>
    <string
        name="my_text">
        "Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem
        Ipsum has been the industry's standard dummy text ever since the 1500s, when
        an unknown printer took a galley of type and scrambled it to make a type
        specimen book. It has survived not only five centuries, but also the leap into
        electronic typesetting, remaining essentially unchanged. It was popularised in
        the 1960s with the release of Letraset sheets containing Lorem Ipsum passages,
        and more recently with desktop publishing software like Aldus PageMaker
        including versions of Lorem Ipsum."
    </string>
</resources>

```

File: Main.java

Example 1-12.

```

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class Main extends Activity implements OnClickListener
{
    private static final String tag = "Main";
    private Button emailButton;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)

```

```

    {
        super.onCreate(savedInstanceState);

        // Set the View Layer
        setContentView(R.layout.main);

        // Get referenc to Email Button
        this.emailButton = (Button) this.findViewById(R.id.emailButton);

        // Sets the Event Listener onClick
        this.emailButton.setOnClickListener(this);
    }

    @Override
    public void onClick(View view)
    {
        if (view == this.emailButton)
        {
            Intent emailIntent = new Intent(android.content.Intent.ACTION_SEND);
            emailIntent.setType("text/html");
            emailIntent.putExtra(android.content.Intent.EXTRA_TITLE, "My Title");
            emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, "My Subject");

            // Obtain refenerenc to String and pass it to Intent
            emailIntent.putExtra(android.content.Intent.EXTRA_TEXT, getString(R.string.my_text));
            startActivity(emailIntent);
        }
    }
}

```

Discussion

1. Modify AndroidManifest.xml to allow for internet connection allowing email to be sent.
2. Create the the visual presentation layer with Email Button which the user clicks.
3. Attach a OnClickListener to allow the email to be sent when the user clicks the Email button.

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.filefactory.com/file/b43debh/n/EmailAndroid.zip>

1.10 Sending an email with attachments

Marco Dinacci

Problem

You want to send an e-mail with attachments.

Solution

We're going to create an Intent, add extended data to specify the file we want to include and start a new activity to allow the user to send the e-mail.

Discussion

The easiest way to send an e-mail is to create an Intent of type ACTION_SEND.

Example 1-13.

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_SUBJECT, "Test single attachment");
intent.putExtra(Intent.EXTRA_EMAIL, new String[]{recipient_address});
intent.putExtra(Intent.EXTRA_TEXT, "Mail with an attachment");
```

To attach a single file, we add some extended data to our intent:

Example 1-14.

```
intent.putExtra(Intent.EXTRA_STREAM, Uri.fromFile(new File("/path/to/file")));
intent.setType("text/plain");
```

The MIME type can always be set as `text/plain` but you may want to be more specific so applications parsing your message will work properly. For instance if you're including a JPEG image you should write `image/jpeg`.

To send an e-mail with multiple attachment the procedure is slightly different:

Example 1-15.

```
Intent intent = new Intent(Intent.ACTION_SEND_MULTIPLE);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_SUBJECT, "Test multiple attachments");
intent.putExtra(Intent.EXTRA_TEXT, "Mail with multiple attachments");
intent.putExtra(Intent.EXTRA_EMAIL, new String[]{recipient_address});
```

```
ArrayList<Uri> uris = new ArrayList<Uri>();
uris.add(Uri.fromFile(new File("/path/to/first/file")));
uris.add(Uri.fromFile(new File("/path/to/second/file")));
```

```
intent.putParcelableArrayListExtra(Intent.EXTRA_STREAM, uris);
```

First, we need to use `Intent.ACTION_SEND_MULTIPLE`, which is available since Android 1.6. Second, we need to create an `ArrayList` with the URIs of the files we want to attach to the mail and call `putParcelableArrayListExtra`.

If sending different type of files you may want to use `multipart/mixed` as MIME type.

Finally, in both cases, you can start a new Activity with the following code:

Example 1-16.

```
startActivity(Intent.createChooser(intent, "Send mail"));
```

`Intent.createChooser` is optional but will allow the user to select his favourite application to send the e-mail.

1.11 Installing .apk files on the emulator

Rachee Singh

Problem

Many free Android applications provide their .apk files. Installing them on the emulator is necessary to check out the application.

Solution

Use of command-line to install the .apk on the running emulator(or the connected Android phone).

Discussion

To install the .apk, follow the following steps:

1. Find the location on your machine where you have installed Android SDK. In the Android SDK directory, go to tools directory.
2. Look for an executable 'adb' in the tools directory. If it is present then that is the location of adb, otherwise, there must be a .txt file named 'adb has moved'. The contents of the file would point you to the location of adb. This file states that adb is present in the 'platform-tools' directory within the Android SDK installation directory.

'tools' directory, with adb has moved.txt:

3. Once you have located the adb, open the terminal in that location(for Linux) for Windows, cd to that location on the command prompt.
4. Use the command: `./adb install location of the apk you want to install` for Linux. For Windows: `adb install location of the apk you want to install`
5. This should start the installation on the current device running (It could be an Emulator that is running or an Android device that is connected).
6. After the installation finishes, in the Menu of the Android device/Emulator you would see the icon of the application you just installed.

1.12 Installing apps onto an Android Emulator

David Dawes

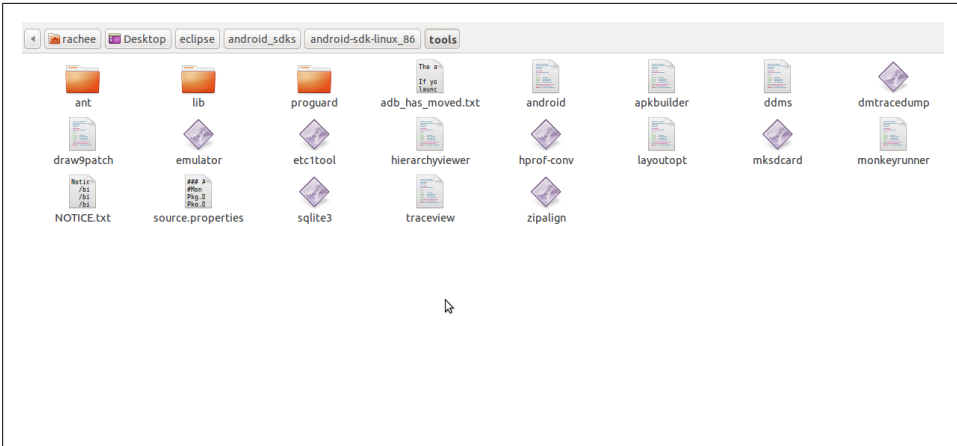


Figure 1-26.

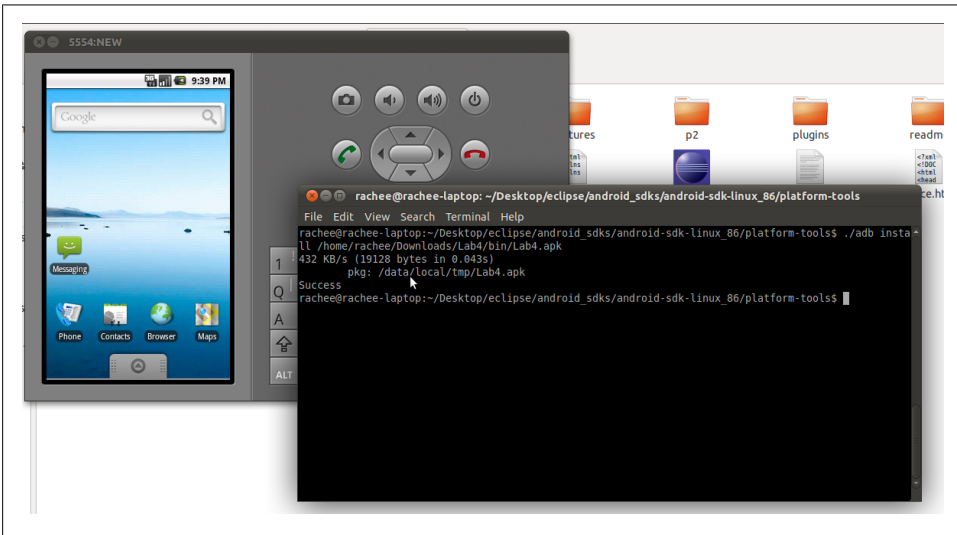


Figure 1-27.

Problem

App stores are a huge element of the attraction of modern smartphones. Unfortunately, if you're developing on an emulated version 2.x Android, you can't currently use Google's Android App store.

Solution

SlideMe (<http://slideme.org/>) offers an alternative app store that works with version 2.x emulated Androids. This allows you to install other apps (perhaps you want to integrate

with other apps), and also to test the experience of publishing and downloading your own apps on your emulated Android. SlideMe also reaches many Android users who are locked out of the Google Android App Store - like anybody from most of the world who doesn't happen to live in the right country. It also allows publishing from outside of the even smaller list of countries supported by Google's Android App Store.

Discussion

There are multiple links showing how to install the Android App Store onto an emulated Android so that you can use it to download apps to your emulated Android. Unfortunately the only options I can find are for versions 1.6 or below of the Android SDK platform.

One alternative I was able to find and use for my 2.1 SDK is [Slide Me, an alternative app store](#) - it may not have as many apps as Google's Android App Store, but it does work with my emulated Android.

Go to the [web site](#) using your emulated Android, browse or search through apps, and click on a free one. After a pause to download, open the download (the little arrow on the top left) and launch the .apk you've downloaded to install the app. I installed the SlideMe app, after reviewing the license (typical stuff: it's licensed to me (I don't own it) exclusively and non-transferably, and only if I accept ALL terms and conditions. They get to include ads, if I want to buy apps there are further considerations, I have to abide by all laws, no porn, no messing with SlideMe, no posting viruses, if it messes anything up it's my problem, not theirs, SlideMe accepts no liability of any sort and you indemnify them against that (nasty, but a mostly standard clause), etc.)

Once the SlideMe app is installed you can go through the catalog and install more apps without using the browser. This is much easier, since the presentation is designed for the Android. Chose a category, scroll through it, and chose an app to install. I have had some stability problems using it on my emulator - it freezes on occasion - but I was able to install some simple free apps like "Grocery list."

I noticed in the Android Invasion discussion forum on linkedin.com that many Android users are disappointed to find that many cell phone providers do NOT include the app store in their Android cell phone offering, and unless you're comfortable rooting and flashing your Android phone there's no way to get it. Most consumers are NOT comfortable rooting and flashing their phones and for them SlideMe offers an alternative way to find free and inexpensive apps for their phones.

SlideMe also allows you to publish your apps onto their app store, See (BROKEN XREF TO RECIPE -1 'Publishing Your App on SlideMe').

See Also

[The SlideMe Application Store.](#)

[SlideMe Upload for Developers.](#)

1.13 Android Epoch HTML/Javascript Calendar

Wagied Davids

Problem

Require a custom calendar using Javascript. Also shows how to interact between Javascript and Java.

Solution

Use a WebView component to load an HTML file containing the Epoch calendar javascript component. Steps involved:

- Download Epoch DHTML/Javascript calendar <http://www.javascriptkit.com/script/script2/epoch/index.shtml>.
- Create an assets directory under your Android Project folder eg. TestCalendar/assets/
- Code your main HTML file for referencing Epoch calendar
- Create an Android Activity for launching the Epoch calendar.

Note that files placed in Android assets directory are referenced like this: `file:///android_asset/` (NOTE: triple leading slash and singular spelled asset)

Discussion

The solution - to make use of the WebView component for loading an HTML file containing the Epoch calendar javascript component. To enable interaction between the Javascript based view layer and the Java-based logic layer, a Java--Javascript bridge interface is required, MyJavaScriptInterface inner class. The `onDayClick()` function shows how to call a javascript function from an Android activity eg. `webView.loadUrl("javascript: popup();");`

File: `calendarview.html`

Example 1-17.

```
<html>
  <head>
    <title>My Epoch DHTML Javascript Calendar</title>
    <style type="text/css">
      dateheader {
        -background-color: #3399FF;
        -webkit-border-radius: 10px;
        -moz-border-radius: 10px;
        -border-radius: 10px;
        -padding: 5px;
      }
    </style>
```

```

<style type="text/css">
html {height:100%;}
body {height:100%; margin:0; padding:0;}
#bg {position:fixed; top:0; left:0; width:100%; height:100%;}
#content {position:relative; z-index:1;}
</style>
<!--[if IE 6]>
<style type="text/css">
html {overflow-y:hidden;}
body {overflow-y:auto;}
#page-background {position:absolute; z-index:-1;}
#content {position:static;padding:10px;}
</style>
<![endif]-->

<link rel="stylesheet" type="text/css" href="epoch_v106/epoch_styles.css" />
<script type="text/javascript" src="epoch_v106/epoch_classes.js"></script>

<script type="text/javascript">
/*You can also place this code in a separate file and link to it like epoch_classes.js*/
var my_cal;

window.onload = function () {
    my_cal = new Epoch('epoch_basic', 'flat', document.getElementById('basic_container'));
};

function popup()
{
    var weekday=new Array("Sun", "Mon", "Tue", "Wed", "Thur", "Fri", "Sat");
    var monthname=new Array("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec");
    var date = my_cal.selectedDates.length > 0 ? my_cal.selectedDates[0] : null;
    if ( date != null )
    {
        var day = date.getDate();
        var dayOfWeek= date.getDay();
        var month = date.getMonth();
        var yy = date.getYear();
        var year = (yy < 1000) ? yy + 1900 : yy;

        /* Set the User selected date in HTML form*/
        var dateStr= weekday[dayOfWeek] + ", " + day + " " + monthname[month] + " " + year;
        document.getElementById("selected_date").value= dateStr;

        /* IMPORTANT: Call Android Javascript->Java bridge setting a Java-field variable
        window.android.setSelectedDate( date );
        window.android.setCalendarButton( date );
    }
}
</script>
</head>
<body>
<div id="bg"></div>
<div id="content">
<div class="dateheader" align="center">

```

```

        <form name="form_selected_date">
            <span style="color:white">Selected day:</span>
            <input id="selected_date" name="selected_date" type="text" readonly="true">
        </form>
    </div>
    <div id="basic_container" onClick="popup()"></div>
</div>
</body>
</head>>

```

File: CalendarView.java

Example 1-18.

```

import java.util.Date;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.webkit.JsResult;
import android.webkit.WebChromeClient;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.Toast;

import com.pfizer.android.R;
import com.pfizer.android.utils.DateUtils;
import com.pfizer.android.view.screens.journal.CreateEntryScreen;

public class CalendarViewActivity extends Activity
{
    private static final String tag = "CalendarViewActivity";
    private ImageView calendarToJournalButton;
    private Button calendarDateButton;
    private WebView webview;
    private Date selectedCalDate;

    private final Handler jsHandler = new Handler();

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        Log.d(tag, "Creating View ...");
        super.onCreate(savedInstanceState);

        // Set the View Layer
        Log.d(tag, "Setting-up the View Layer");
        setContentView(R.layout.calendar_view);
    }
}

```

```

// Go to CreateJournalEntry
calendarToJournalButton = (ImageView) this.findViewById(R.id.calendarToJournalButton);
calendarToJournalButton.setOnClickListener(new OnClickListener()
{
    @Override
    public void onClick(View v)
    {
        Log.d(tag, "Re-directing -> CreateEntryScreen ...");
        Intent intent = intent = new Intent(getApplicationContext(), CreateEntryScreen.class);
        startActivity(intent);
    }
});

// User-Selected Calendar Date
calendarDateButton = (Button) this.findViewById(R.id.calendarDateButton);

// Get access to the WebView holder
webView = (WebView) this.findViewById(R.id.webview);

// Get the settings
WebSettings settings = webView.getSettings();

// Enable Javascript
settings.setJavaScriptEnabled(true);

// Enable ZoomControls visibility
settings.setSupportZoom(true);

// Add Javascript Interface
webView.addJavaScriptInterface(new MyJavaScriptInterface(), "android");

// Set the Chrome Client
webView.setWebChromeClient(new MyWebChromeClient());

// Load the URL of the HTML file
webView.loadUrl("file:///android_asset/calendarview.html");

}

public void setCalendarButton(Date selectedCalDate)
{
    Log.d(tag, jsHandler.obtainMessage().toString());
    calendarDateButton.setText(DateUtils.convertDateToSectionHeaderFormat(selectedCalDate.getTime()));
}

/**
 *
 * @param selectedCalDate
 */
public void setSelectedCalDate(Date selectedCalDate)
{
    this.selectedCalDate = selectedCalDate;
}

```

```

/**
 *
 * @return
 */
public Date getSelectedCalDate()
{
    return selectedCalDate;
}

/**
 * JAVA->JAVASCRIPT INTERFACE
 *
 * @author wagied
 */
final class MyJavaScriptInterface
{
    private Date jsSelectedDate;
    MyJavaScriptInterface()
    {
        // EMPTY;
    }

    public void onDayClick()
    {
        jsHandler.post(new Runnable()
        {
            public void run()
            {
                // Java telling Javascript to do things
                webView.loadUrl("javascript: popup();");
            }
        });
    }

    /**
     * NOTE: THIS FUNCTION IS BEING SET IN JAVASCRIPT User-selected Date in
     * WebView
     *
     * @param dateStr
     */
    public void setSelectedDate(String dateStr)
    {
        Toast.makeText(getApplicationContext(), dateStr, Toast.LENGTH_SHORT).show();
        Log.d(tag, "User Selected Date: Javascript -> Java : " + dateStr);

        // Set the User Selected Calendar date
        setJsSelectedDate(new Date(Date.parse(dateStr)));
        Log.d(tag, "java.util.Date Object: " + Date.parse(dateStr).toString());
    }
    private void setJsSelectedDate(Date userSelectedDate)
    {
        jsSelectedDate = userSelectedDate;
    }
    public Date getJsSelectedDate()

```

```

        {
            return jsSelectedDate;
        }
    }

/**
 * Alert pop-up for debugging purposes
 *
 * @author w david01
 */
final class MyWebChromeClient extends WebChromeClient
{
    @Override
    public boolean onJsAlert(WebView view, String url, String message, JsResult result)
    {
        Log.d(tag, message);
        result.confirm();
        return true;
    }

    @Override
    public void onDestroy()
    {
        Log.d(tag, "Destroying View!");
        super.onDestroy();
    }
}

```

For debugging purposes, a `MyWebChromeClient` is created and the `onJsAlert()` method is overridden.

1.14 Sharing Java classes from another Eclipse Project

Ian Darwin

Problem

You want to use a class from another project, but don't want to copy-and-paste

Solution

Add the project as a "referenced project", and Eclipse (and DEX) will do the work.

Discussion

You often need to re-use classes from another project. In JPSTrack GPS tracking program, the Android version borrows classes like the file I/O module from the Java SE version. You surely do not want to copy and paste classes willy-nilly from one project into another, because this makes maintenance improbable.

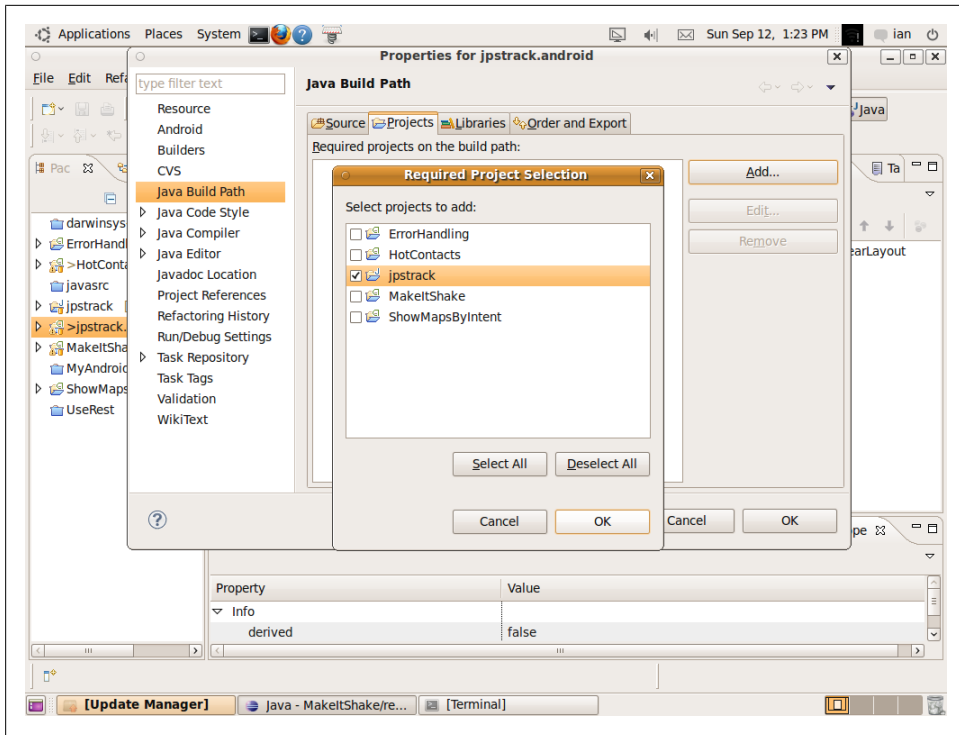


Figure 1-28.

All you really have to do is declare the project containing the needed classes (the Java SE version in this case) as a referenced project on the build path. Select Project->Properties->Build Path, select Project, and click "Add". In the screenshot I am adding the SE project "jpstrack" as a dependency on the Android version project "jpstrack.android".

Since you are probably keeping both projects under source control (if these are programs you ever ship, you should!), remember to tag both projects when you release the Android project - one of the points in favor of source control is that you need to be able to re-create exactly what you shipped.

1.15 Referencing libraries to implement external functionality

Rachee Singh

Problem

You need to reference an external library in your source code

Solution

Obtain the JAR file for the library that you require and add it to your project

Discussion

As an example, you might require to use AndroidPlot, a library to plot charts and graphs in your application or use OpenStreetMaps. For this your application needs to reference these libraries. In Eclipse this can be done in a few simple steps:

- Download the JAR file corresponding to the library you wish to use.
- After creating your Android project in Eclipse, right click on the project name and select 'Properties' option in the menu.
- From the list on the left side, select 'Java Build Path' and click on the 'Libraries' tab.
- Click on 'Add External JARs' button.
- Provide the location where you downloaded the JAR file for the library you wish to use.

Now you will see a 'Referenced Libraries' directory appearing in your project. Within this directory the JARs that you added will appear.

An alternate approach is to create a 'lib' folder in your project, physically copy the Jar files there, and add them individually as above, but using "Add Jars". This way keeps everything in one place (especially if your project is shared by a version control system to others who might even use a different operating system and be unable to locate the external jars in the same place). However it does raise the burden of responsibility for license issues on the included jar files.

In either case, if you also build with Ant, be sure to update your *build.xml* file.

Whichever way you do it, it's pretty easy to add libraries to your project.

1.16 Use SDK Samples to Help Avoid Head Scratching

Daniel Fowler

Problem

Sometimes it is a struggle to code up some functionality, especially when the documentation is sketchy or does not provide any examples.

Solution

Looking at existing working code will help. The SDK has sample programs that can be picked apart to see how they work.

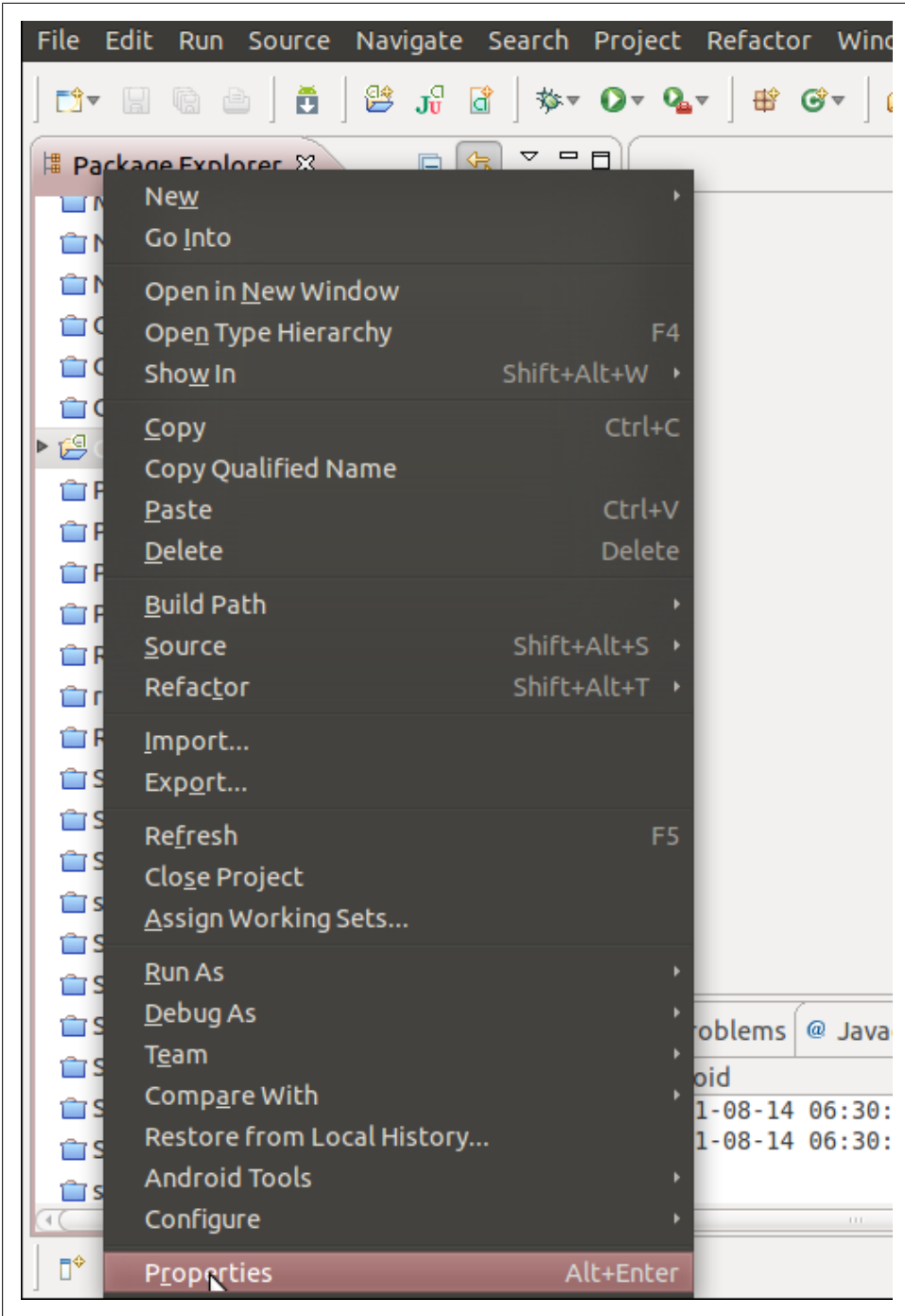


Figure 1-29.

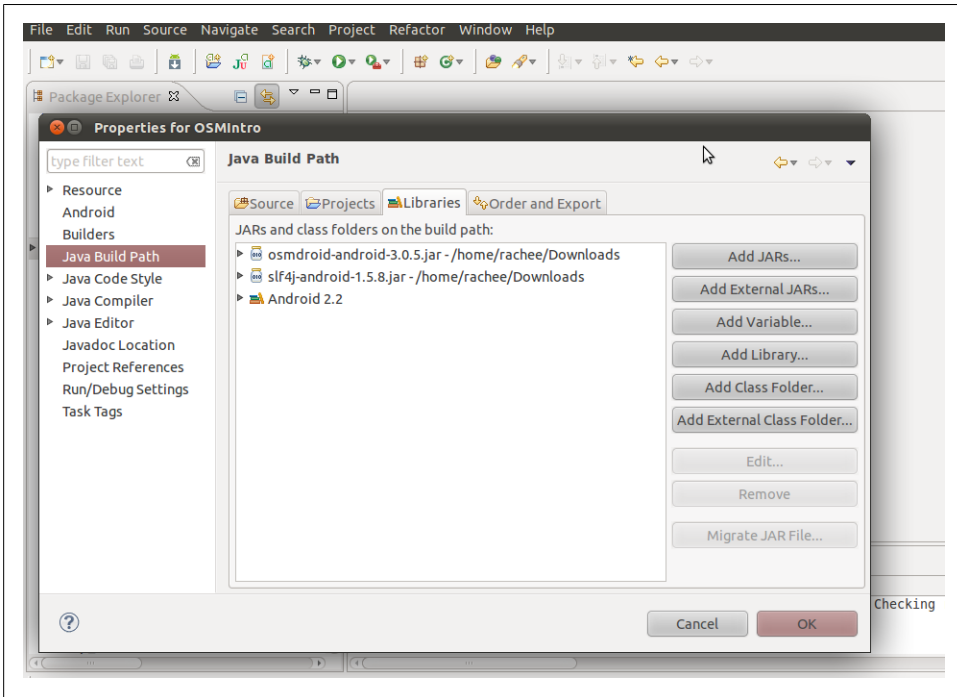


Figure 1-30.

Discussion

The Android SDK comes with several sample applications that can be useful when trying to code up some functionality. Looking through the sample code can be insightful. Once the Android SDK is installed there are several samples available:

- Accelerometer Play
- Accessibility Service
- API Demos
- Backup and Restore
- Bluetooth Chat
- Business Card
- Contact Manager
- Cube Live Wallpaper
- Home
- Honeycomb Gallery
- JetBoy

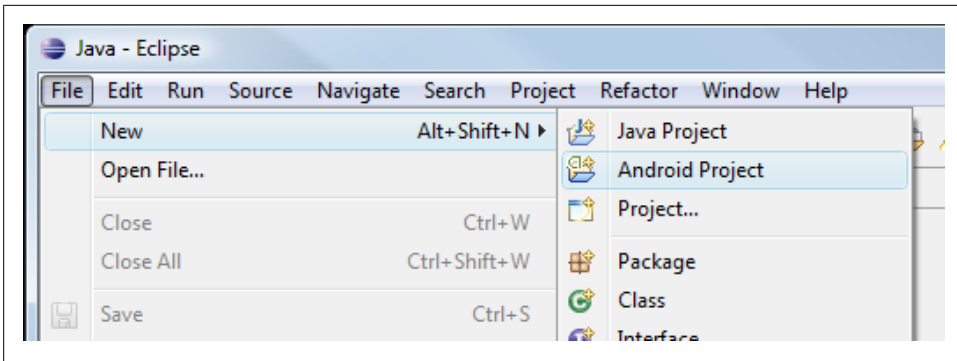


Figure 1-31.

- Lunar Lander
- Multiple Resolutions
- Near Field Communication
- Note Pad
- RenderScript
- Sample Sync Adapter
- Searchable Dictionary
- Session Initiation Protocol
- Snake
- Soft Keyboard
- Spinner
- SpinnerTest
- StackView Widget
- TicTacToeLib
- TicTacToeMain
- USB
- Wiktionary
- Wiktionary (Simplified)
- Weather List Widget
- XML Adapters

To open a sample project from Eclipse open the 'File' menu and then select 'Android Project'.

On the 'New Android Project' dialog select the 'Create project from existing sample' option. Select the API level/platform version required from the list of available 'Build Targets'. Select the required sample from the 'Samples' dropdown. If the drop down is

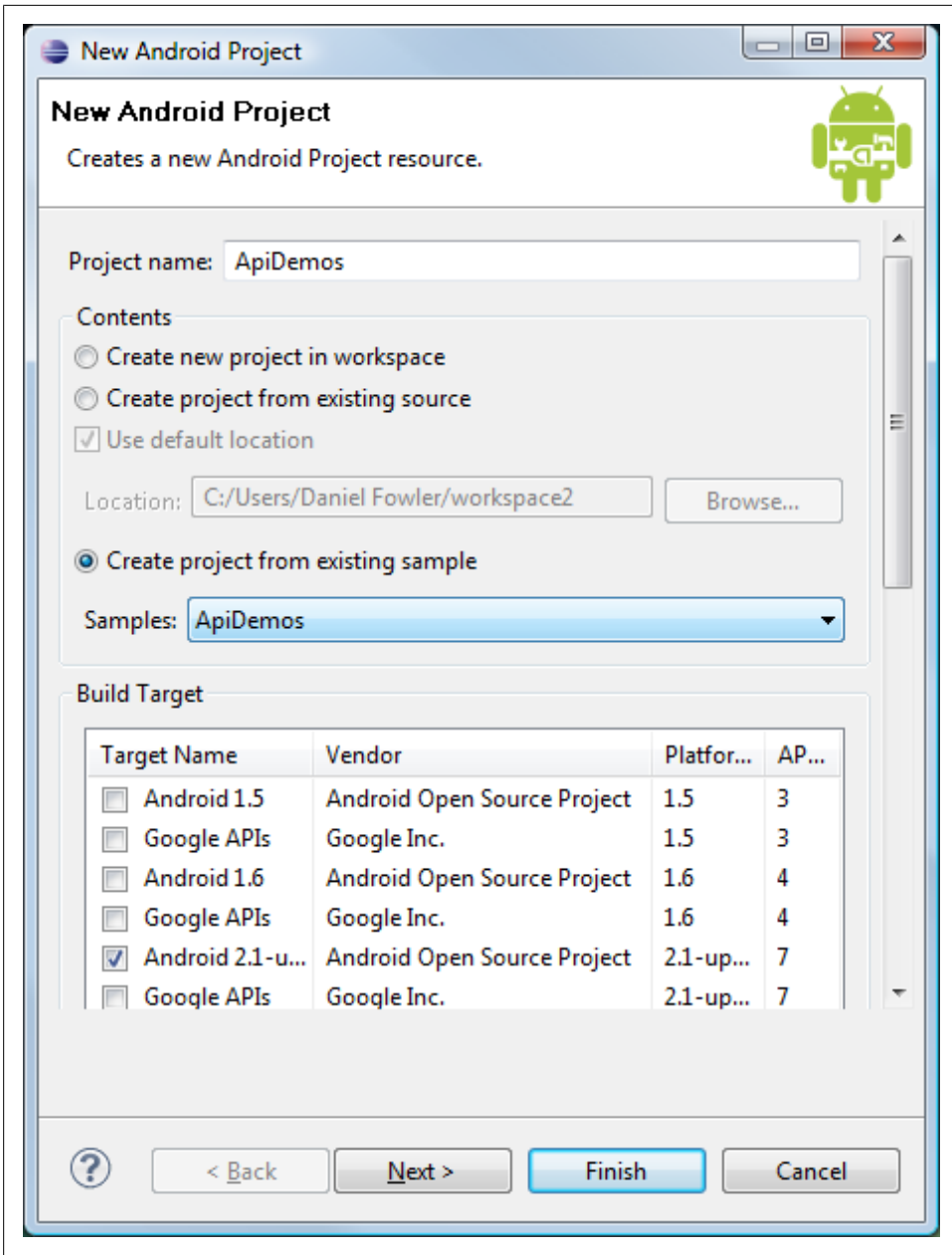


Figure 1-32.

not enabled this is because samples for the selected API level have not been installed or are not available. The SDK Manager can be used to install the samples, or if not available for that API level select a different API level.

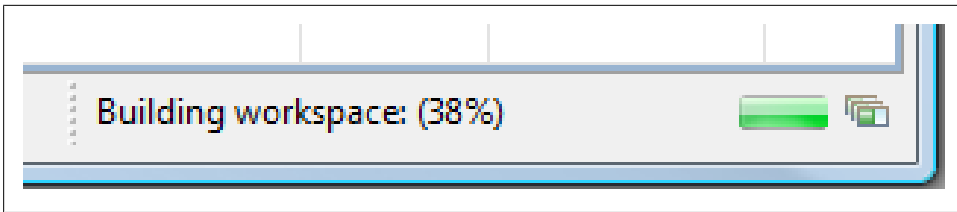


Figure 1-33.

Click 'Finish', the sample will load into the workspace and build (progress shown on the status bar).

After a short time the sample will be ready to run and the source code can be browsed to see how it is all done.

If the samples have been moved from the SDK samples directory use the 'Create project from existing source' option on the 'New Android Project' dialog to open the sample.

When the sample is first run select 'Android Application' in the 'Run As' dialog that may appear. It may also be necessary to configure an appropriate AVD to run the sample (see [Recipe 1.5](#)).

Additional sources of information include:

- The Android Developers web site at <http://developer.android.com/index.html>
- This Cookbook of course.
- Doing a web search.

Still head scratching, ask someone a question, try:

- Stack Overflow at <http://www.stackoverflow.com> using the Tag "android".
- Internet Relay Chat (IRC) channel #android-dev on freenode.

1.17 Keeping the Android SDK Updated

Daniel Fowler

Problem

The SDK must be kept updated to allow App developers to work with the latest APIs on the evolving Android platform.

Solution

Use the *Android SDK and AVD Manager* to update the existing installed SDK packages and to install new SDK packages. This includes third party packages for device specific functionality.

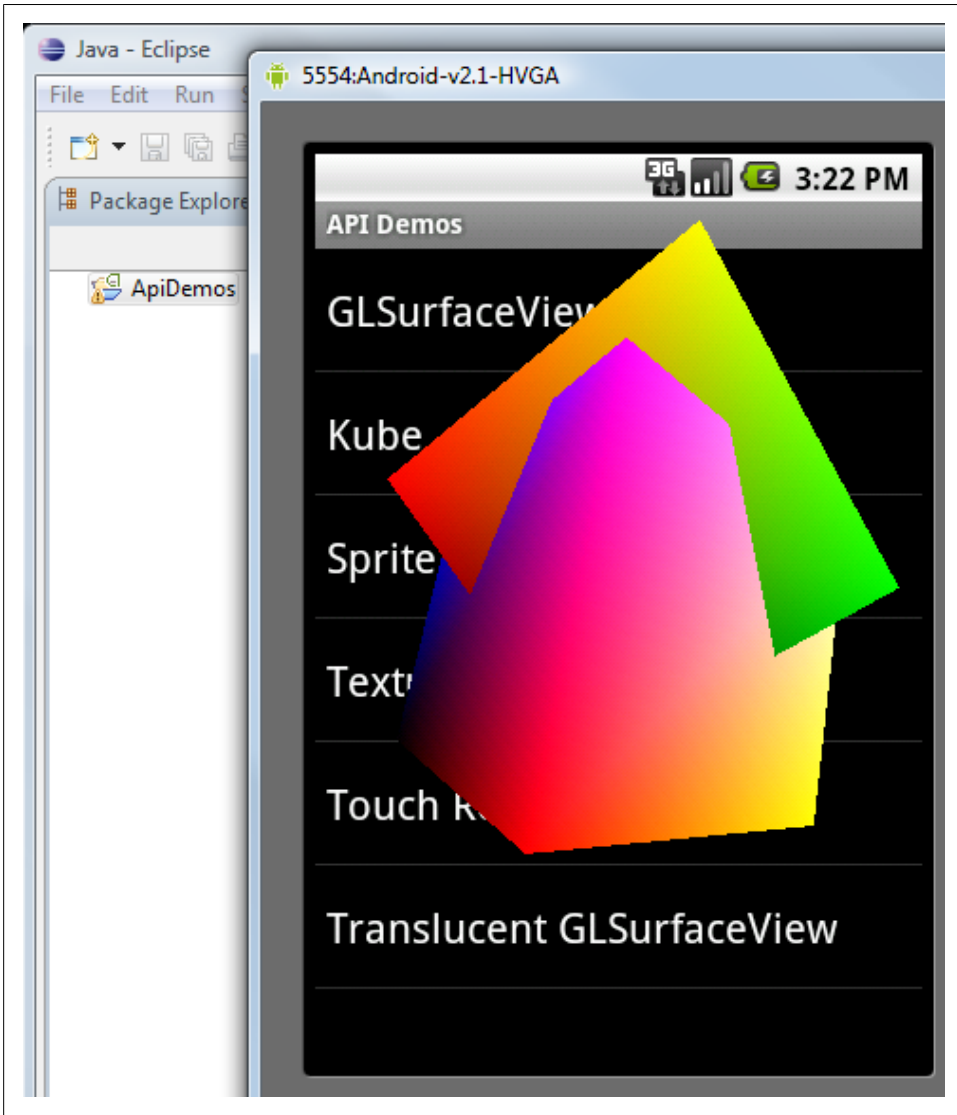


Figure 1-34.

Discussion

The Android Operating System (OS) is constantly evolving and therefore so is the Android Software Development Kit (SDK). The ongoing development of Android is driven by:

- Google's research and development.

- Phone manufacturers developing new and improved handsets.
- Addressing security issues and possible exploits.
- The need to support new devices (e.g. support for tablets devices was added with version 3.0).
- Support for new hardware interfaces (e.g. adding support for Near Field Communication in version 2.3).
- Fixing bugs.
- Improvements in functionality (e.g. a new JavaScript engine).
- Changes in the underlying Linux kernel.
- Deprecation of redundant programming interfaces.
- New uses (e.g. Google TV).
- The wider Android development community.

Installation of the Android SDK has been covered elsewhere (see [Recipe 1.6](#) or <http://developer.android.com/sdk/installing.html>). After the SDK is installed on the development machine, and the programming environment is running smoothly, once in a while developers will need to check for updates to the SDK.

The SDK can be kept up to date by running the *SDK Manager* program. (On a Windows machine run *SDK Manager.exe* in the folder *C:\Program Files\Android\android-sdk*, or use the **Start** button, then **All Programs**, **Android SDK Tools** and click **SDK Manager**). The SDK Manager automatically scans for updates and new packages.

If the SDK Manager is opened from within Eclipse (using the **Window** menu and selecting **Android SDK and AVD Manager**) then the scan has to be manually started. Select **Installed packages** in the left hand column, then click the **Update All...** button.

Available updates and new packages will show in a list. If a package has licence terms that require accepting they are shown with a question mark. Highlight each package with a question mark to read the licence terms. The package can be accepted or rejected using the radio buttons. Rejected packages are marked with a red cross.

Alternatively click on **Accept All** to accept everything that is available. All packages and updates ready to download and install will be shown with a green tick.

Click the **Install** button to begin the download and installation.

Once the downloads and package installations are complete click the **Close** button.

If the Android SDK and AVD Manager program (*SDK Manager.exe*) has itself been updated there will be a message asking to restart the program.

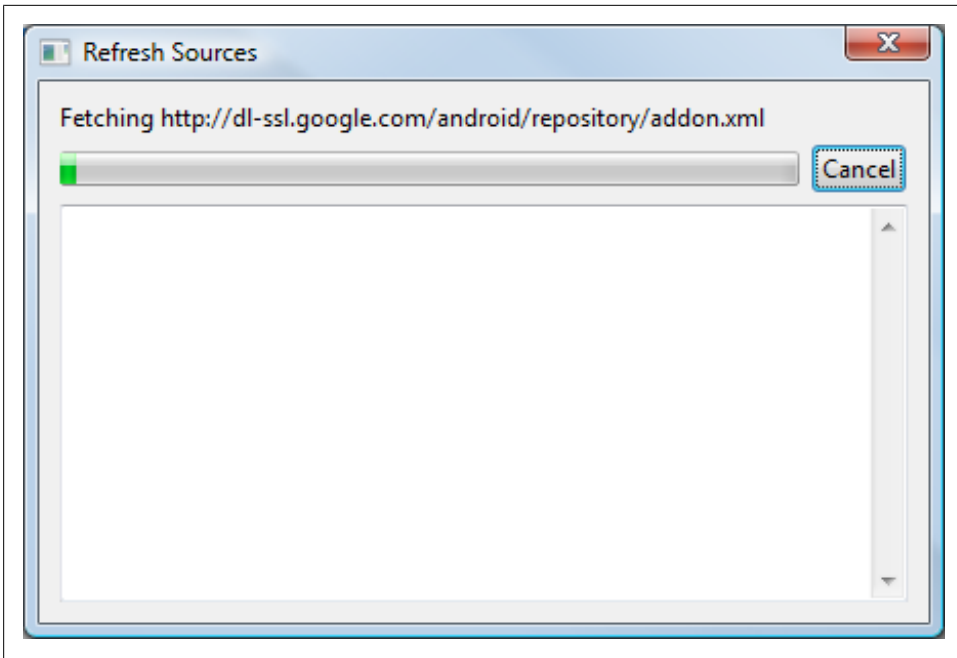


Figure 1-35.

When the SDK Manager program does the scan for updates and new packages there may be none available; in this case an empty list is displayed which can be closed using the **Cancel** button.

The SDK Manager is also used to download additional packages that are not part of the standard platform. This is used by device manufacturers to provide support for their own hardware. For example LG Electronics provide a 3D device and to support 3D capability in applications an additional package is provided.

In the **Android SDK and AVD Manager** dialog click on **Available packages**. Expand and tick the required package in the right hand list then click **Install Selected** to get the third party package. If a third party package is not listed the URL to a *respository.xml* file, provided by the package publisher, will need to be entered via the **Add Add-On Site...** button.

On a Windows machine the default location for the SDK is under the *Program Files \Android\android-sdk* directory. This is a restricted directory and can cause the SDK installation to fail. A message dialog with the title "SDK Manager: failed to install" can appear.

To overcome this error there are several items to check.

- Unplug any Android devices (this may prevent adb.exe from closing).

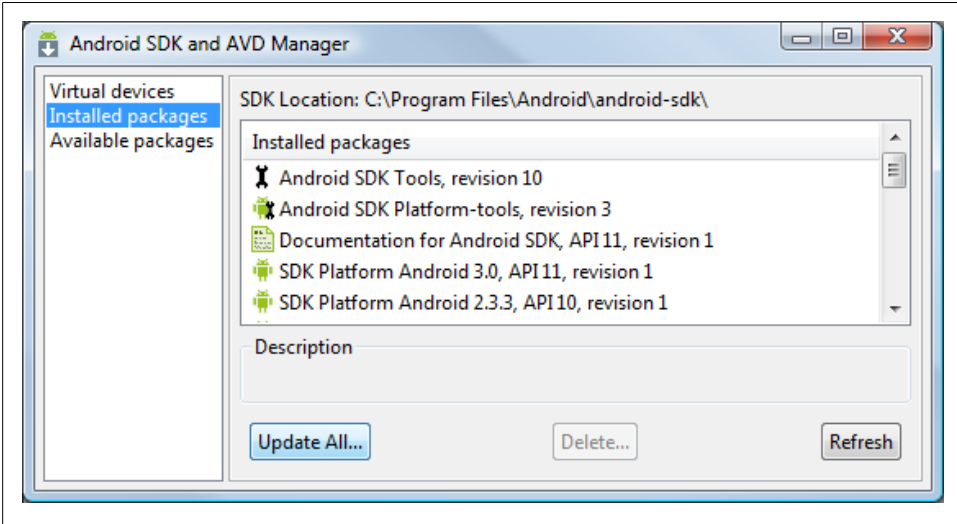


Figure 1-36.

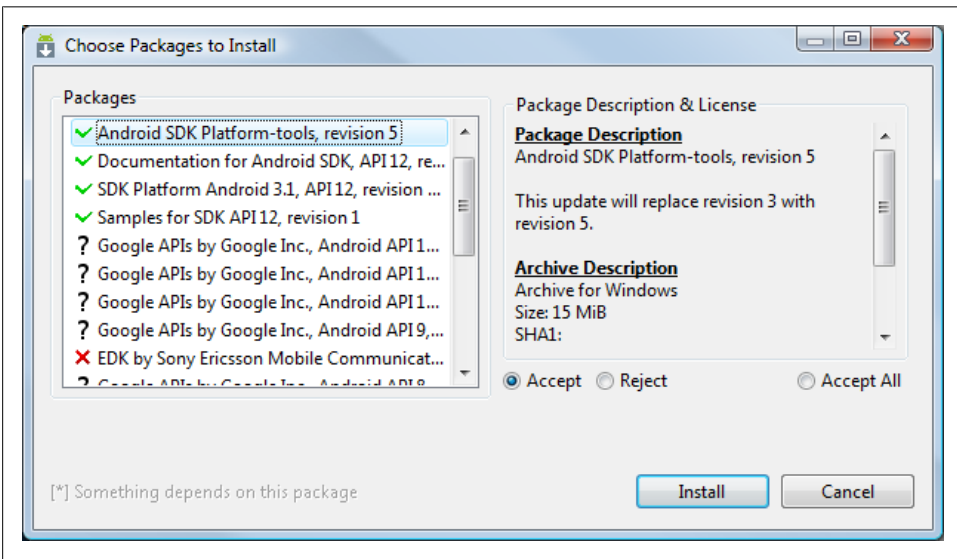


Figure 1-37.

- Browse to `C:\Program Files\Android\Android-sdk`, bring up the Properties for the tools folder (context menu then Properties). Ensure that the Read Only checkbox is cleared.

There may be a need to give permission to change the attributes.

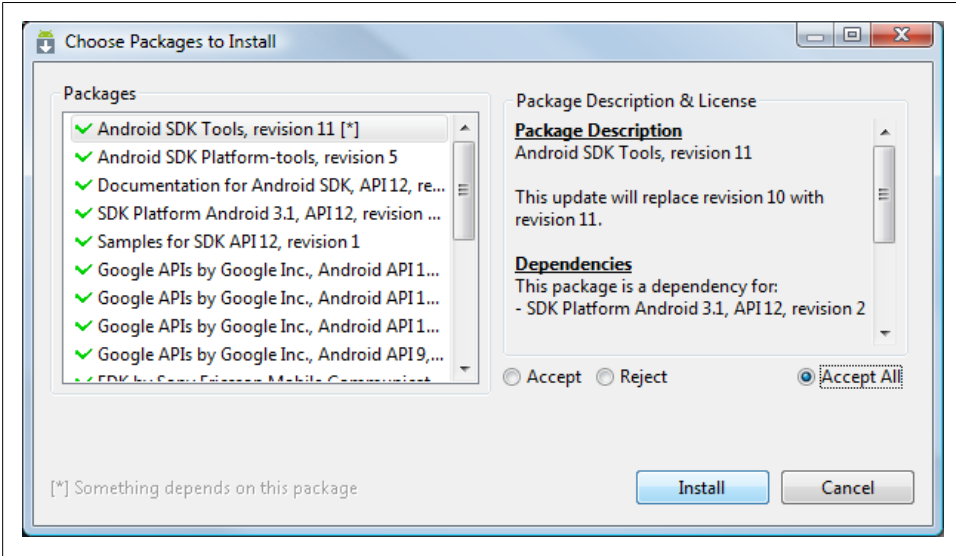


Figure 1-38.

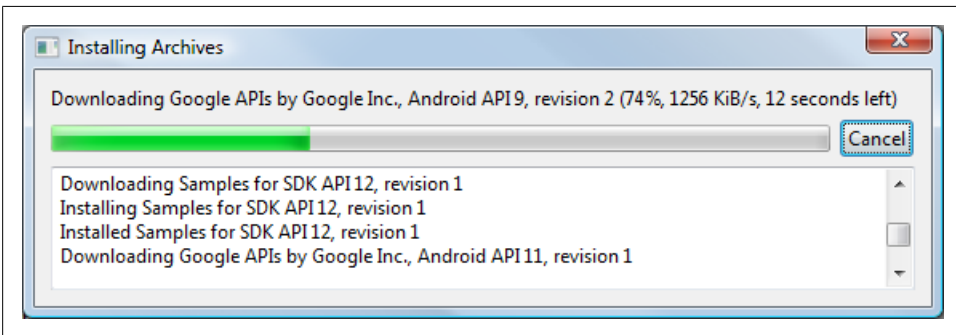


Figure 1-39.

A "Confirm Attribute Changes" dialog will be shown, ensure the option **Apply changes to this folder, subfolders and files** is selected and press **OK**.

- Restart the computer.
- Ensure that all other programs are closed, especially any copies of File Explorer.
- Run `SDK Manager.exe` under the administrator account. Bring up context menu and select **Run as administrator**.

Further information on troubleshooting the SDK Manager and Android Eclipse plugin can be found on the [Android Developers](http://android-developers.com) web site.

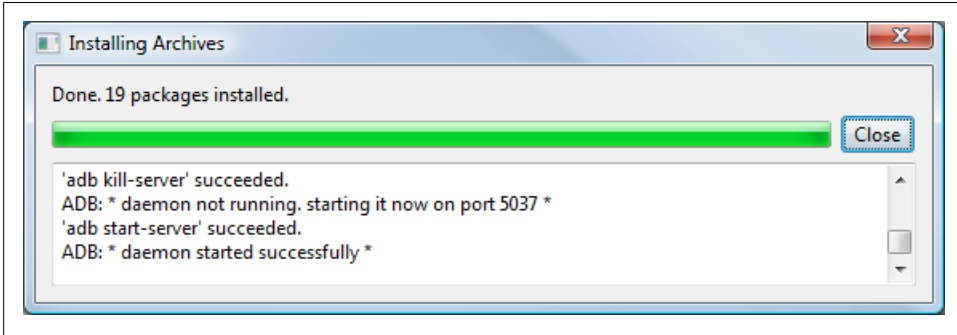


Figure 1-40.

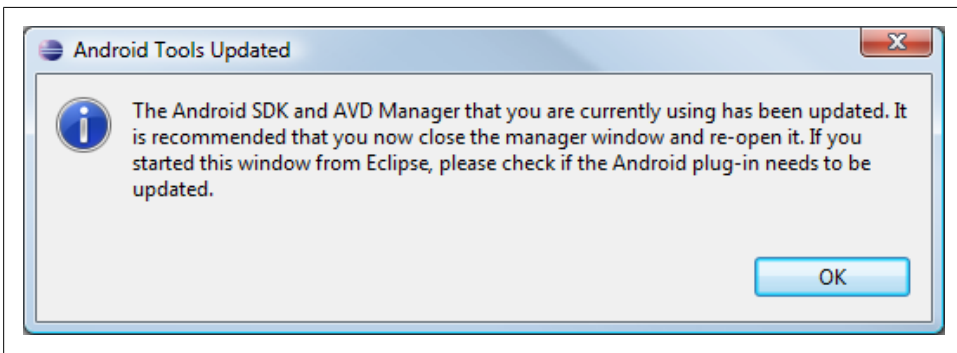


Figure 1-41.

See Also

[Recipe 1.6](#)

[Installing the SDK](#)

[Adding SDK Components](#)

[ADT Plugin for Eclipse](#)

1.18 Five Ways to Wire Up an Event Listener

Daniel Fowler

Problem

Developers need to be familiar with the different ways to code event handlers, they they will come across different methods in tutorials, samples and online snippets.

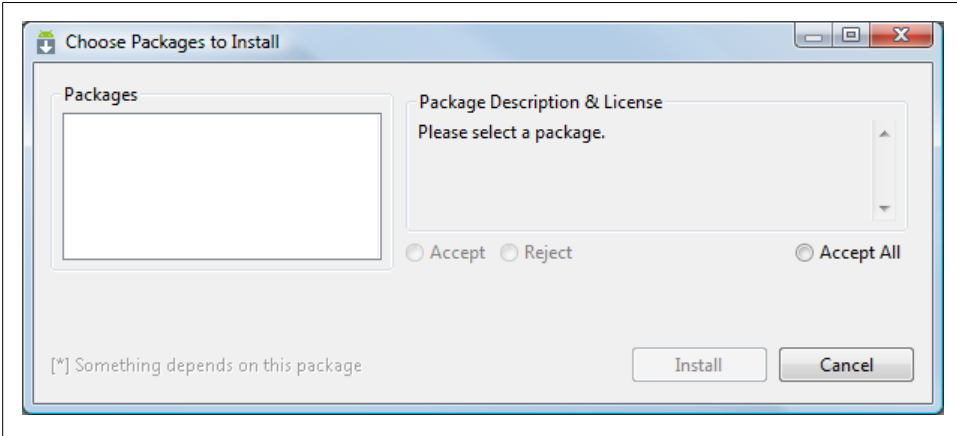


Figure 1-42.

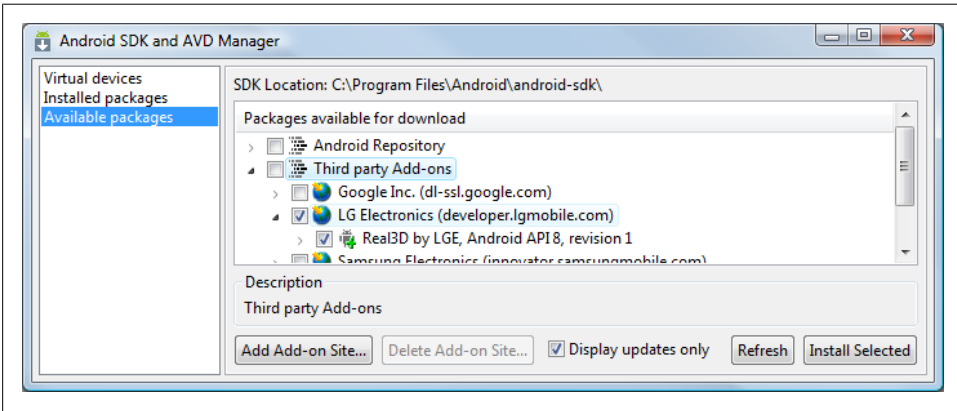


Figure 1-43.

Solution

When writing software very rarely is there only one way to do things, this is true when wiring up View events, five methods are shown here.

Discussion

When a `View` fires an event an `Application` will not respond to it unless it is listening for it. To detect the event a class that implements a listener is instantiated and assigned to the `View`. Take for example the `onClick` event, the most widely used event in Android Apps. Nearly every `View` that can be added to an App screen will fire the event when the user stabs it with their finger (on touch screens) or presses the trackpad/trackball when the `View` has focus. This event is listened to by a class implementing the `OnClickListener`

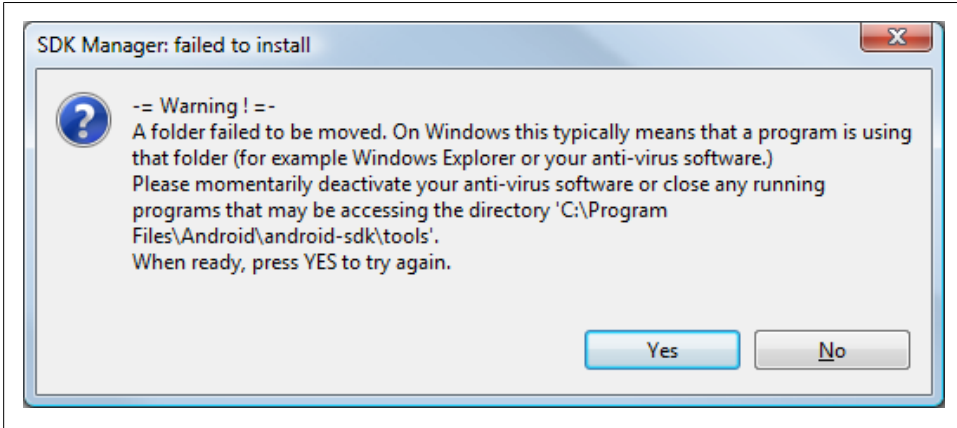


Figure 1-44.

Listener interface. The class instance is then assigned to the required View using the View's `setOnClickListener` method. In the following code an Activity sets the text of a `TextView` (`textview1`) when a `Button` (`button1`) is pressed.

1. Member Class

A class called `HandleClick` implementing `OnClickListener` is declared as a member of the Activity (`main`). This is useful when several listeners require similar processing than can be handled by a single class.

Example 1-19.

```
public class main extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //attach an instance of HandleClick to the Button
        findViewById(R.id.button1).setOnClickListener(new HandleClick());
    }
    private class HandleClick implements OnClickListener{
        public void onClick(View arg0) {
            Button btn = (Button)arg0;    //cast view to a button
            // get a reference to the TextView
            TextView tv = (TextView) findViewById(R.id.textview1);
            // update the TextView text
            tv.setText("You pressed " + btn.getText());
        }
    }
}
```

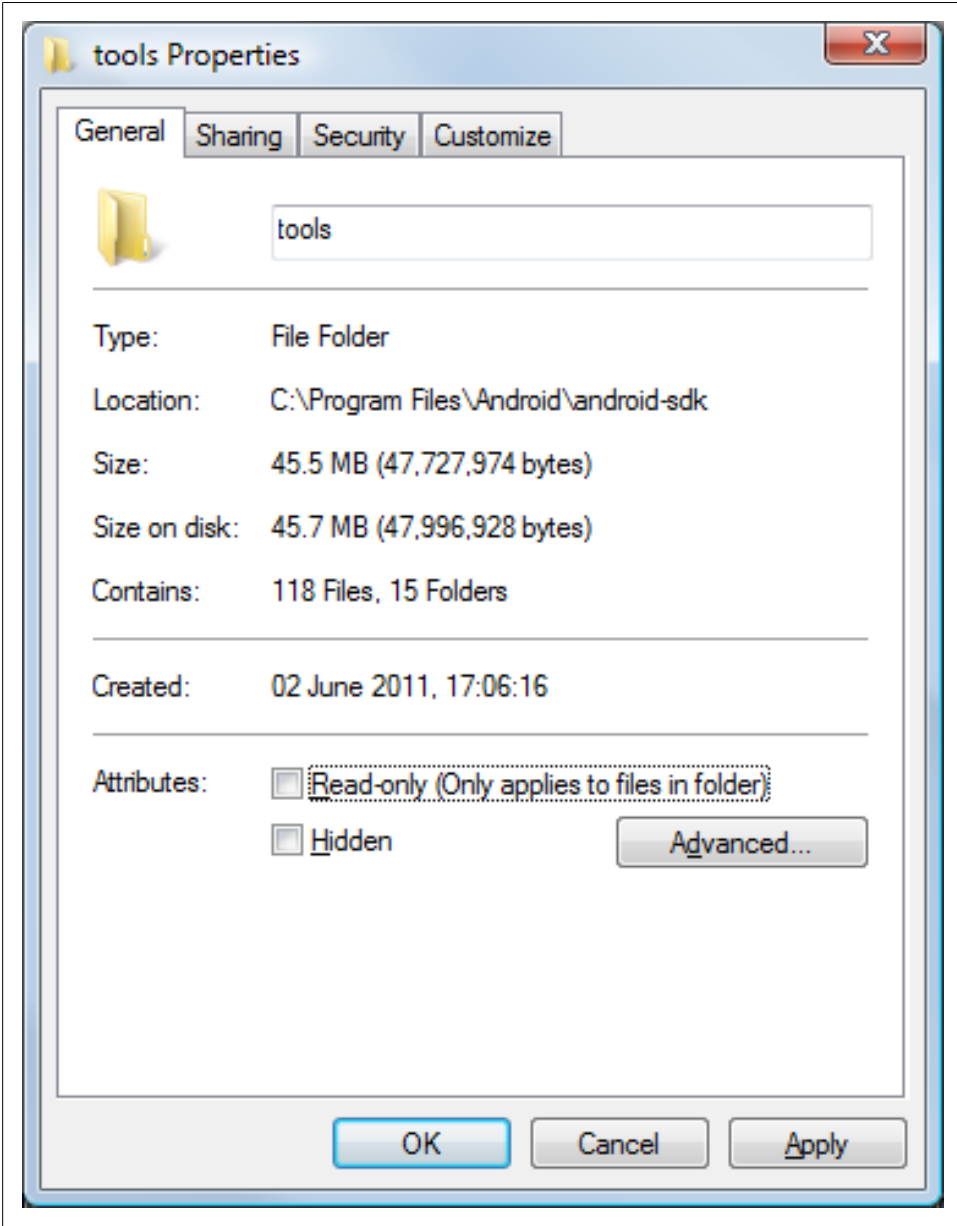


Figure 1-45.

2. Interface Type

In Java an Interface can be used as a type, a variable is declared as an `OnClickListener` and assigned using `new OnClickListener(){...}`, behind the scenes Java is cre-

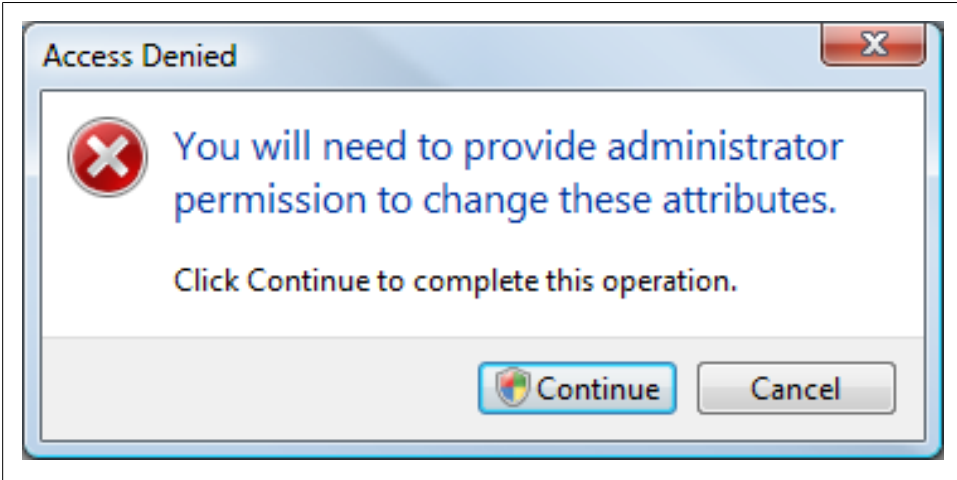


Figure 1-46.

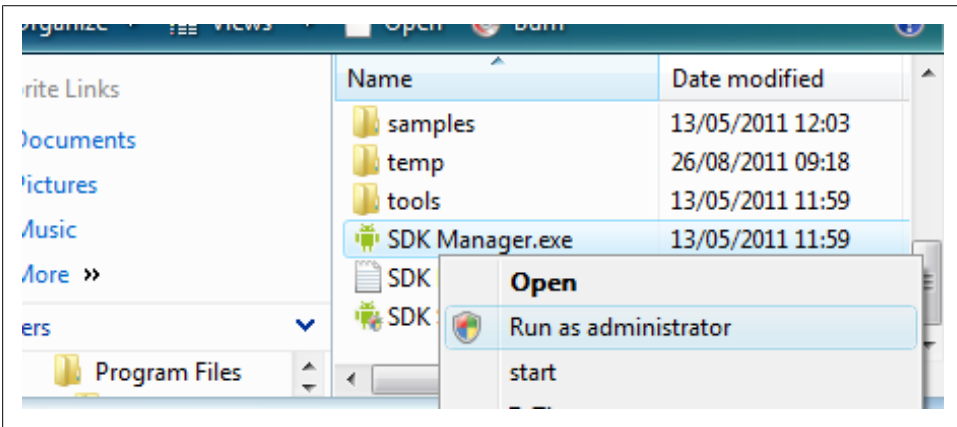


Figure 1-47.

ating an object (an Anonymous Class) that implements `OnClickListener`. This has similar benefits to the first method.

Example 1-20.

```
public class main extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //use the handleClick variable to attach the event listener
        findViewById(R.id.button1).setOnClickListener(handleClick);
    }
}
```



```

private OnClickListener handleClick = new OnClickListener(){
    public void onClick(View arg0) {
        Button btn = (Button)arg0;
        TextView tv = (TextView) findViewById(R.id.textview1);
        tv.setText("You pressed " + btn.getText());
    }
};
}

```

3. Anonymous Inner Class

Declaring the `OnClickListener` within the call to the `setOnClickListener` method is common. This method is useful when each listener does not have functionality that could be shared with other listeners. Some novice developers find this type of code difficult to understand. Again behind the scenes for `new OnClickListener(){...}` Java is creating an object that implements the interface.

Example 1-21.

```

public class main extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        findViewById(R.id.button1).setOnClickListener(new OnClickListener(){
            public void onClick(View arg0) {
                Button btn = (Button)arg0;
                TextView tv = (TextView) findViewById(R.id.textview1);
                tv.setText("You pressed " + btn.getText());
            }
        });
    }
}

```

4. Implementation in Activity

The Activity itself can implement the `OnClickListener`. Since the Activity object (*main*) already exists this saves a small amount of memory by not requiring another object to host the `onClick` method. It does make public a method that is unlikely to be used elsewhere. Implementing multiple events will make the declaration of *main* long.

Example 1-22.

```

public class main extends Activity implements OnClickListener{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        findViewById(R.id.button1).setOnClickListener(this);
    }
    public void onClick(View arg0) {
        Button btn = (Button)arg0;
        TextView tv = (TextView) findViewById(R.id.textview1);
    }
}

```

```

        tv.setText("You pressed " + btn.getText());
    }
}

```

5. Attribute in View Layout for OnClick Events

In Android 1.6 and later (API level 4 and upwards) a named class defined in the Activity can be assigned to the `android:onClick` attribute in a layout file. This can save writing a lot of boilerplate code.

Example 1-23.

```

public class main extends Activity{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    public void HandleClick(View arg0) {
        Button btn = (Button)arg0;
        TextView tv = (TextView) findViewById(R.id.textview1);
        tv.setText("You pressed " + btn.getText());
    }
}

```

In the layout file the Button would be declared with the `android:onClick` attribute.

Example 1-24.

```

<Button android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 1"
        android:onClick="HandleClick"/>

```

The first four methods of handling events can be used with other event types (`onLongClick`, `onKey`, `onTouch`, `onCreateContextMenu`, `onFocusChange`). The fifth method only applies to the `onClick` event. The layout file below declares an additional two buttons and using the `android:onClick` attribute no additional code is required than that defined above, i.e. no additional `findViewById` and `setOnClickListener` for each button is required.

Example 1-25.

```

<?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:id="@+id/textview1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Click a button."
        android:textSize="20dp"/>

```

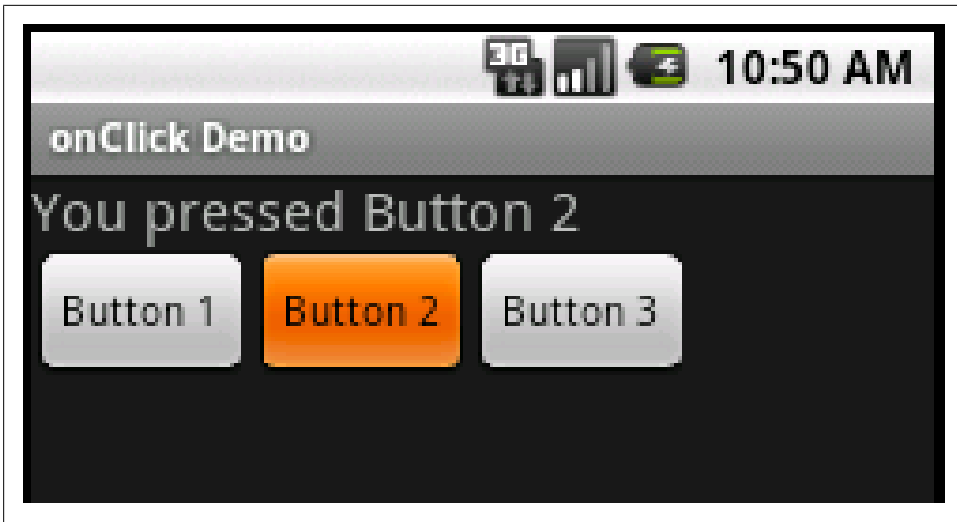


Figure 1-48.

```
<LinearLayout android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <Button android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 1"
        android:onClick="HandleClick"/>
    <Button android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 2"
        android:onClick="HandleClick"/>
    <Button android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 3"
        android:onClick="HandleClick"/>
</LinearLayout>
</LinearLayout>
```

Deciding which technique to use to wire up a listener will depend on the functionality required, how much code is reusable across Views and how easy the code would be to understand by future maintainers. Ideally the code should be succinct and easy to view.

One method not shown here is similar to the first method. In the first method it would be possible to save the listener class in a different class file as a public class. Then instances of that public class could be used by other Activities, passing the Activity's context in via the constructor. However, Activities should try and stay self contained in case they are killed by Android. Sharing listeners across Activities is against the ideals

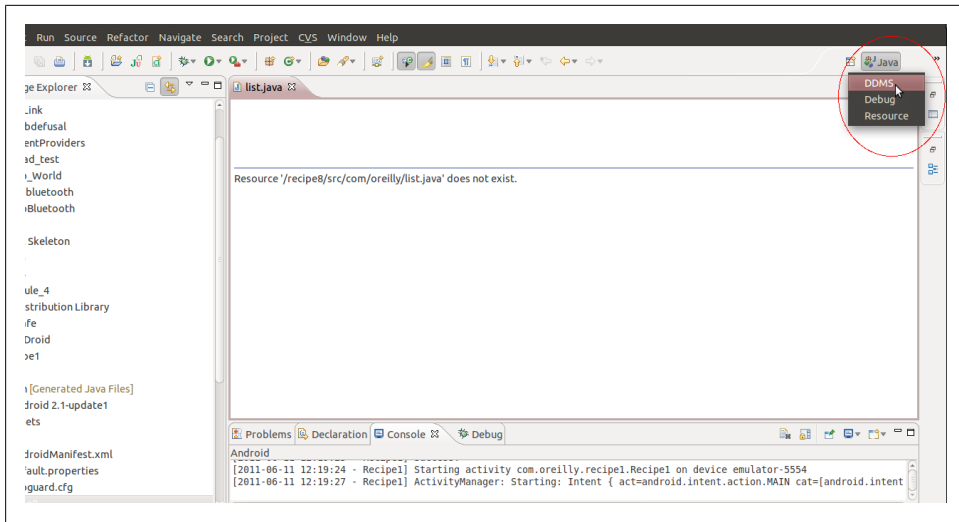


Figure 1-49.

of the Android platform and could lead to unnecessary complexity passing references between the public classes.

1.19 Taking a Screenshot from the Emulator/Android Device

Rachee Singh

Problem

Taking a screen-shot of the application running on an Android Device.

Solution

Use the 'Device Screen Capture' feature of the DDMS View in Eclipse.

Discussion

1. Run the Application in Eclipse and go to the DDMS View (Window Menu->Open Perspective->Other->DDMS) or Window Menu->Show View->Other->Android->Devices; the former is shown in the screenshots in this Recipe).
2. In the DDMS View, go to the Device Screen Capture Icon.
3. A window showing the current screen of the Emulator/Android Device pops up. You can save the screenshot for the describing the app!

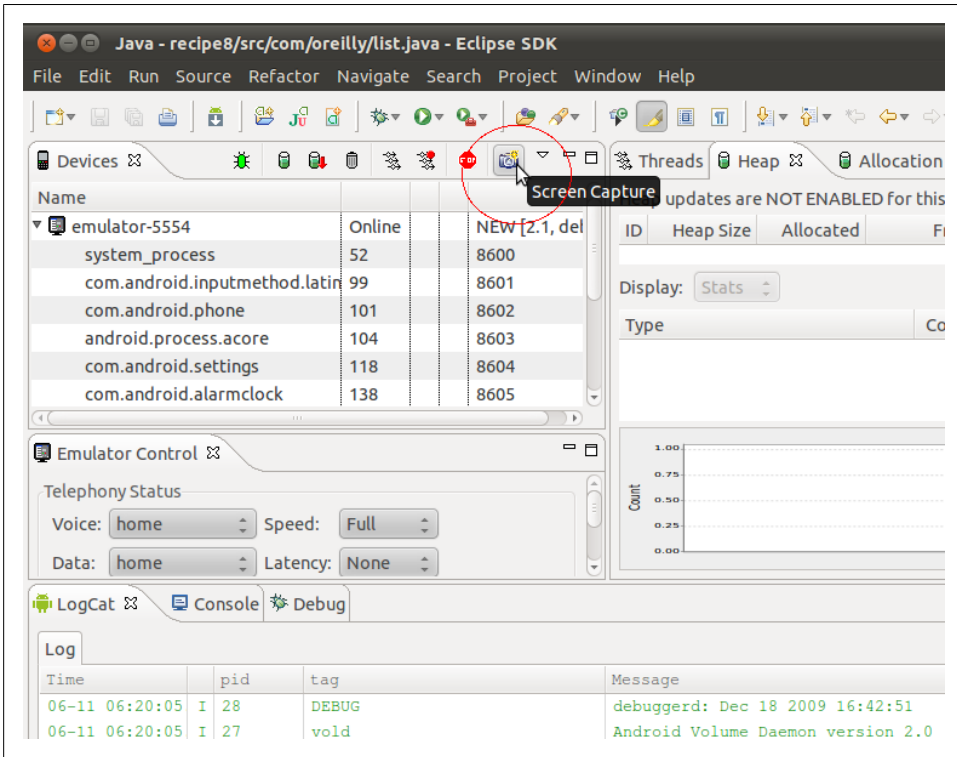


Figure 1-50.

1.20 Program: A Simple CountdownTimer example

Wagied Davids

Problem

Require a simple count down timer

Solution

Android comes with a built-in class for constructing CountdownTimers. It's easy to use, efficient, and works (that goes without saying!).

1. Create a subclass of CountdownTimer.
2. Override the onTick() and onFinish() methods.
3. Instantiate a new instance in your Android Activity,
4. Call the start() method on the new instance created!

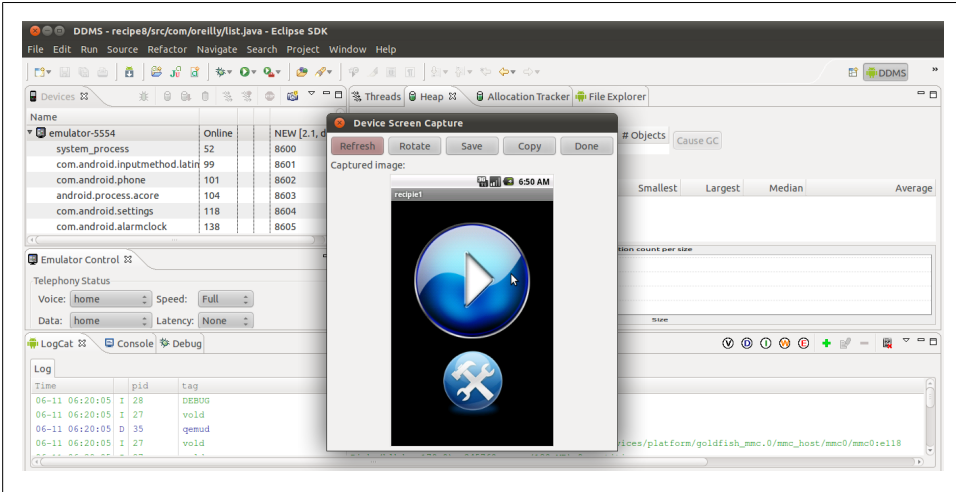


Figure 1-51.

Discussion

File: main.xml

Example 1-26.

```
<?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">
    <Button
        android:id="@+id/button"
        android:text="Start"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
    <TableLayout
        android:padding="10dip"
        android:layout_gravity="center"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TableRow>
            <TextView
                android:id="@+id/timer"
                android:text="Time: "
                android:paddingRight="10dip"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />
            <TextView
                android:id="@+id/timeElapsed"
```

```

        android:text="Time elapsed: "
        android:paddingRight="10dip"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    </TableRow>
</TableLayout>
</LinearLayout>

```

File: Main.java

Example 1-27.

```

package com.examples;

import android.app.Activity;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class Main extends Activity implements OnClickListener
{
    private static final String tag = "Main";
    private MalibuCountDownTimer countDownTimer;
    private long timeElapsed;
    private boolean timerHasStarted = false;
    private Button startB;
    private TextView text;
    private TextView timeElapsedView;

    private final long startTime = 50000;
    private final long interval = 1000;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        startB = (Button) this.findViewById(R.id.button);
        startB.setOnClickListener(this);

        text = (TextView) this.findViewById(R.id.timer);
        timeElapsedView = (TextView) this.findViewById(R.id.timeElapsed);
        countDownTimer = new MalibuCountDownTimer(startTime, interval);
        text.setText(text.getText() + String.valueOf(startTime));
    }

    @Override
    public void onClick(View v)
    {
        if (!timerHasStarted)
        {

```

```

        countdownTimer.start();
        timerHasStarted = true;
        startB.setText("Start");
    }
    else
    {
        countdownTimer.cancel();
        timerHasStarted = false;
        startB.setText("RESET");
    }
}

// CountdownTimer class
public class MalibuCountDownTimer extends CountdownTimer
{
    public MalibuCountDownTimer(long startTime, long interval)
    {
        super(startTime, interval);
    }

    @Override
    public void onFinish()
    {
        text.setText("Time's up!");
        timeElapsedView.setText("Time Elapsed: " + String.valueOf(startTime));
    }

    @Override
    public void onTick(long millisUntilFinished)
    {
        text.setText("Time remain:" + millisUntilFinished);
        timeElapsed = startTime - millisUntilFinished;
        timeElapsedView.setText("Time Elapsed: " + String.valueOf(timeElapsed));
    }
}
}

```

The result looks like this:

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.filefactory.com/file/cbbbc38/n/SimpleCountDownTimerExample.zip>

1.21 Program: Tipster, a tip calculator for the Android OS

Sunit Katkar

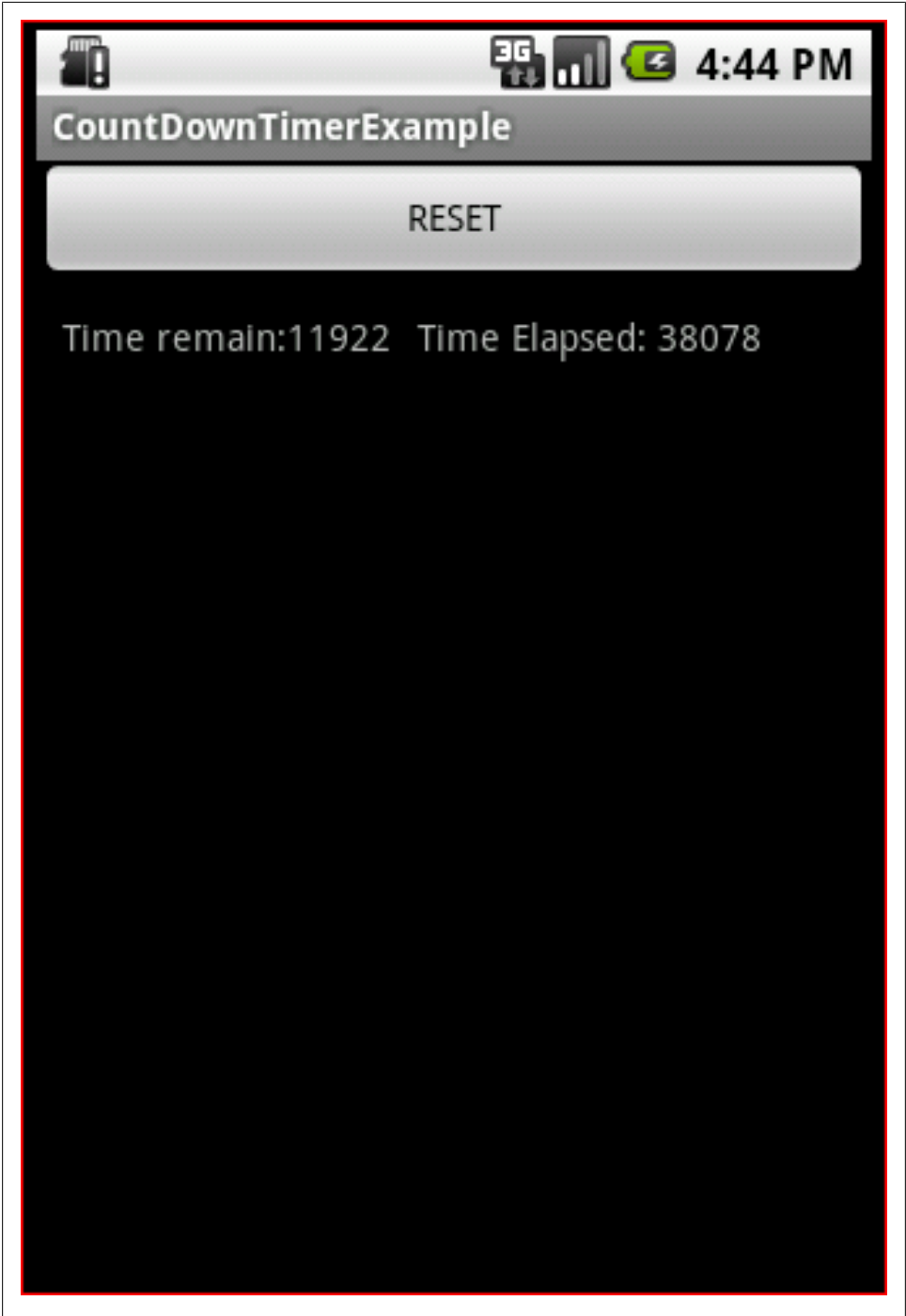


Figure 1-52.

Problem

A tip calculator is quite a simple application. When you go with friends to a restaurant and wish to divide the check and tip, you simply add the tip percentage to the total and divide by the number of diners. Tipster is an implementation of this in Android, to show a complete application.

Solution

This is a simple exercise to use the basic GUI elements in Android and then piecing them together with some simple calculation and some event driven UI code to tie it all together. We will use the following GUI components:

- `TableLayout` - provides a good control over screen layout. This layout allows us to use the HTML Table tag paradigm to layout widgets
- `TableRow` - this defines a row in the `TableLayout`. Its like the HTML TR and TD tag combined.
- `TextView` - this View provides a label for displaying static text on the screen
- `EditText` - this View provides a text field for entering values.
- `RadioGroup` - this groups together radio buttons
- `RadioButton` - this provides a radio button
- `Button` - this is the regular button
- `View` - we will use a `View` to create a visual separator with certain height and color attributes

Discussion

Android uses XML files for the Layout of widgets. In our example project, the Android plugin for Eclipse generates a `main.xml` file for the layout. This file has the XML based definitions of the different widgets and their containers.

There is a `strings.xml` file which has all the string resources used in the application. A default `icon.png` file is provided for the application icon.

Then there is the `R.java` file which is automatically generated (and updated when any changes are made to `main.xml`). This file has the constants defined for each of the layout and widget. Do not edit this file manually. The plugin is does it for you when you do a clean build.

In our example we have `Tipster.java` as the main Java file or the Activity.

Google tutorials highlight how to use the plugin. Using the Eclipse plugin, create an Android project named Tipster. The end result will be a project layout like the following screen shot.

Creating the Layout and placing the Widgets The end goal is to create a layout as shown in the following screen shot.

For this screen layout we will use the following layouts and widgets:

- `TableLayout` - provides a good control over screen layout. This layout allows us to use the HTML Table tag paradigm to layout widgets
- `TableRow` - this defines a row in the `TableLayout`. Its like the HTML TR and TD tag combined.
- `TextView` - this View provides a label for displaying static text on the screen
- `EditText` - this View provides a text field for entering values.
- `RadioGroup` - this groups together radio buttons
- `RadioButton` - this provides a radio button
- `Button` - this is the regular button
- `View` - we will use a View to create a visual separator with certain height and color attributes

Familiarize yourself with these widgets as you will be using these quite a lot in applications you build. When you go to the Javadocs for each of the above, do look up the XML attributes. This will help you correlate the usage in the `main.xml` layout file and the Java code (`Tipster.java` and `R.java`) where these are accessed.

I came across a UI tool Droid Draw, that lets you create a layout by drag-and-drop of widgets from a palette, just like any form designer tool. However, I would recommend that you create the layout by hand in XML, at least in your initial stages of learning Android. Later on you, as you learn all the nuances of the XML layout API, you can delegate the task to such tools.

The Layout file - `main.xml` This file has the layout information. I have posted the file below. The source code comments make the file quite self-explanatory.

A `TableRow` widget creates a single row inside the `TableLayout`. So we use as many `TableRows` as the number of rows we want. In this tutorial we use 8 `TableRows` - 5 for the widgets till the visual separator below the buttons and 3 for the results area below the buttons and separator.

`/res/layout/main.xml`

Example 1-28.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Using table layout to have HTML table like control over layout -->
<TableLayout
    android:id="@+id/TableLayout01"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1"
    xmlns:android="http://schemas.android.com/apk/res/android">
```

```

<!-- Row 1: Text label placed in column zero,
      text field placed in column two and allowed to
      span two columns. So a total of 4 columns in this row -->
<TableRow>
<TextView
    android:id="@+id/txtLbl1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="0"
    android:text="@string/textLbl1"/>
<EditText
    android:id="@+id/txtAmount"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:numeric="decimal"
    android:layout_column="2"
    android:layout_span="2"
    />
</TableRow>
<!-- Row 2: Text label placed in column zero,
      text field placed in column two and allowed to
      span two columns. So a total of 4 columns in this row -->
<TableRow>
<TextView
    android:id="@+id/txtLbl2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="0"
    android:text="@string/textLbl2"/>
<EditText
    android:id="@+id/txtPeople"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:numeric="integer"
    android:layout_column="2"
    android:layout_span="3"/>
</TableRow>
<!-- Row 3: This has just one text label placed in column zero -->
<TableRow>
<TextView
    android:id="@+id/txtLbl3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/textLbl3"/>
</TableRow>
<!-- Row 4: RadioGroup for RadioButtons placed at column zero
      with column span of three, thus creating one radio button
      per cell of the table row. Last cell number 4 has the
      textfield to enter a custom tip percentage -->
<TableRow>
<RadioGroup
    android:id="@+id/RadioGroupTips"
    android:orientation="horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

```

```

        android:layout_column="0"
        android:layout_span="3"
        android:checkedButton="@+id/radioFifteen">
<RadioButton android:id="@+id/radioFifteen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/rdoTxt15"
        android:textSize="15sp" />
<RadioButton android:id="@+id/radioTwenty"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/rdoTxt20"
        android:textSize="15sp" />
<RadioButton android:id="@+id/radioOther"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/rdoTxtOther"
        android:textSize="15sp" />
</RadioGroup>
    <EditText
        android:id="@+id/txtTipOther"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:numeric="decimal"/>
</TableRow>
<!-- Row for the Calculate and Rest buttons. The Calculate button
    is placed at column two, and Reset at column three -->
<TableRow>
<Button
        android:id="@+id/btnReset"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_column="2"
        android:text="@string/btnReset"/>
<Button
        android:id="@+id/btnCalculate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_column="3"
        android:text="@string/btnCalculate"/>
</TableRow>

<!-- TableLayout allows any other views to be inserted between
    the TableRow elements. So inserting a blank view to create a
    line separator. This separator view is used to separate
    the area below the buttons which will display the
    calculation results -->
<View
        android:layout_height="2px"
        android:background="#DDFFDD"
        android:layout_marginTop="5dip"
        android:layout_marginBottom="5dip"/>

<!-- Again table row is used to place the result textviews
    at column zero and the result in textviews at column two -->

```

```

<TableRow android:paddingBottom="10dip" android:paddingTop="5dip">
<TextView
    android:id="@+id/txtLb14"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="0"
    android:text="@string/textLb14"/>
<TextView
    android:id="@+id/txtTipAmount"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="2"
    android:layout_span="2"/>
</TableRow>

<TableRow android:paddingBottom="10dip" android:paddingTop="5dip">
<TextView
    android:id="@+id/txtLb15"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="0"
    android:text="@string/textLb15"/>
<TextView
    android:id="@+id/txtTotalToPay"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="2"
    android:layout_span="2"/>
</TableRow>

<TableRow android:paddingBottom="10dip" android:paddingTop="5dip">
<TextView
    android:id="@+id/txtLb16"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="0"
    android:text="@string/textLb16"/>
<TextView
    android:id="@+id/txtTipPerPerson"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="2"
    android:layout_span="2"/>
</TableRow>
<!-- End of all rows and widgets -->
</TableLayout>

```

TableLayout and TableRow After examining the main.xml, you can gather that the TableLayout and TableRow are straightforward to use. You create the TableLayout once, then insert a TableRow. Now you are free to insert any other widgets like TextView, EditText, etc. inside this TableRow.

Do look at the attributes, especially android:stretchColumns, android:layout_column, android:layout_span which allow widget placement like the way you would use a

regular HTML table. I recommend that you follow the links to these attributes and read up on how they work for a `TableLayout`.

Controlling input values Look at the `EditText` widget in the `main.xml` file at line 19. This is the first text field for entering the 'Total Amount' of the check. We want only numbers here. We can accept decimal numbers because real restaurant checks can be for dollars and cents, and not just dollars. So we use the `android:numeric` attribute with a value of `decimal`. So this will allow whole values like 10 and decimal values 10.12, but will prevent any other type of entry.

This is a simple and concise way to control input values, thus achieving two things, saving us the trouble of writing validation code in the `java` file `Tipster.java`, and ensuring that the user does not enter erroneous values. This XML based constraints feature of Android is quite powerful and useful. You should explore all possible attributes that go with a particular widget to extract maximum benefits from this XML shorthand way of setting constraints. In a future release, unless I have missed it completely in this release, I wish that Android allows for entering ranges for the `android:numeric` attribute, so that we can define what range of numbers we wish to accept.

Since ranges are not currently available (to the best of my knowledge), you will see later on that we do have to check for certain values like zero or empty values to ensure our tip calculation arithmetic does not fail.

Examining `Tipster.java` Next we look at the `Tipster.java` file which controls our application. This is the main class which does the layout, the event handling and the application logic.

Application code - `Tipster.java` The Android Eclipse plugin creates the `Tipster.java` file in our project with default code as follows -

Code Snippet 1 of `/src/com/examples/tipcalc/Tipster.java`

Example 1-29.

```
package com.examples.tipcalc;

import android.app.Activity;

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

The `Tipster` class extends the `android.app.Activity` class. An activity is a single, focused thing that the user can do. The `Activity` class takes care of creating the window and then laying out the UI. You have to call the `setContentView(View view)` method to put

your UI in the Activity. So think of Activity as a outer frame which is empty, and you populate it with your UI.

Now look at the snippet of the Tipster.java class. First we define the widgets as class members. Look at lines 3 to 12 of the code snippet 2 below for reference.

Then we use the `findViewById(int id)` method to locate the widgets. The ID of each widget, defined in your `main.xml` file, is automatically defined in the `R.java` file when you clean and build the project in Eclipse. (If you have set up Eclipse to Build Automatically, then the `R.java` file is instantaneously updated when you update `main.xml`)

Each widget is derived from the `View` class, and provides special graphical user interface features. So a `TextView` provides a way to put labels on the UI, while the `EditText` provides a text field. Look at lines 24 to 41 in the code snippet 2 below. You can see how `findViewById()` is used to locate the widgets.

Code Snippet 2 of `/src/com/examples/tipcalc/Tipster.java`

Example 1-30.

```
public class Tipster extends Activity {
    // Widgets in the application
    private EditText txtAmount;
    private EditText txtPeople;
    private EditText txtTipOther;
    private RadioGroup rdoGroupTips;
    private Button btnCalculate;
    private Button btnReset;

    private TextView txtTipAmount;
    private TextView txtTotalToPay;
    private TextView txtTipPerPerson;

    // For the id of radio button selected
    private int radioCheckedId = -1;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        // Access the various widgets by their id in R.java
        txtAmount = (EditText) findViewById(R.id.txtAmount);
        //On app load, the cursor should be in the Amount field
        txtAmount.requestFocus();

        txtPeople = (EditText) findViewById(R.id.txtPeople);
        txtTipOther = (EditText) findViewById(R.id.txtTipOther);

        rdoGroupTips = (RadioGroup) findViewById(R.id.RadioGroupTips);

        btnCalculate = (Button) findViewById(R.id.btnCalculate);
        //On app load, the Calculate button is disabled
```



```

btnCalculate.setEnabled(false);

btnReset = (Button) findViewById(R.id.btnReset);

txtTipAmount = (TextView) findViewById(R.id.txtTipAmount);
txtTotalToPay = (TextView) findViewById(R.id.txtTotalToPay);
txtTipPerPerson = (TextView) findViewById(R.id.txtTipPerPerson);

// On app load, disable the 'Other tip' percentage text field
txtTipOther.setEnabled(false);

```

Addressing 'ease of use' or Usability concerns Our application must try to be as usable as any other established application or web page. In short, adding Usability features will give a good user experience. To address these concerns look at the code snippet 2 again.

Look at line 26 where we use the method `requestFocus()` of the View class. Since `EditText` widget is derived from the View class, this method is applicable to it. This is done to so that when our application loads, the 'Total Amount' text field will receive focus and the cursor will be placed in it. This is similar to popular web application login screens where the cursor is present in the username textfield.

Now look at line 35 where the 'Calculate' button is disabled by calling the `setEnabled(boolean enabled)` method on the Button widget. This is done so that the user cannot click on it before entering values in the required fields. If we allowed the user to click Calculate without entering values in the 'Total Amount' and 'No. of People' fields, then we would have to write validation code to catch these conditions. This would entail showing an alert popup warning the user about the empty values. This adds unnecessary code and user interaction. When the user sees the 'Calculate' button disabled, its quite obvious that unless all values are entered, the tip cannot be calculated.

Look at line 44 in the code snippet 2. Here the 'Other Percentage' text field is disabled. This is done because the '15% tip' radio button is selected by default when the application loads. This default selection on application load is done via the main.xml file. Look at line 66 of main.xml where the following statement selects the '15% tip' radio button.

Example 1-31.

```
android:checkedButton="@+id/radioFifteen"
```

The `RadioGroup` attribute `android:checkedButton` allows you to select one of the `RadioButton` widgets in the group by default.

Most users, who have used popular applications on the desktop as well as the web, are familiar with the 'disabled widgets enabled on certain conditions' paradigm..

Adding such small conveniences always makes the application more usable and the user experience richer.

Processing UI events Like popular Windows, Java Swing, Flex, etc. UI frameworks, Android too provides an Event model which allows to listen to certain events in the UI caused by user interaction. Let us see how we can use the Android event model in our application.

Listening to radio buttons First let us focus on the radio buttons in the UI. We want to know which radio button was selected by the user, as this will allow us to determine the 'Tip Percentage' in our calculations. To 'listen' to radio buttons, we use the static interface `OnCheckedChangeListener()`. This will notify us when the selection state of a radio button changes.

In our application, we want to enable the 'Other Tip' text field only when the 'Other' radio button is selected. When the 15% and 20% buttons are selected we want to disable this text field. Besides this, we want to add some more logic for sake of usability. As discussed before, we should not enable the 'Calculate' button till all required fields have valid values. In case of the three radio buttons, we want to ensure that the Calculate button gets enabled for the following two conditions -

Other radio button is selected and the 'Other Tip Percentage' text field has valid values
15% or 20% radio button is selected and 'Total Amount' and 'No. Of People' text fields have valid values. Look at the code snippet 3 which deals with the radio buttons. The source code comments are quite self explanatory.

Code Snippet 3 of `/src/com/examples/tipcalc/Tipster.java`

Example 1-32.

```
/*
 * Attach a OnCheckedChangeListener to the
 * radio group to monitor radio buttons selected by user
 */
rdoGroupTips.setOnCheckedChangeListener(new OnCheckedChangeListener() {

@Override
public void onCheckedChanged(RadioGroup group, int checkedId) {
    // Enable/disable Other Percentage tip field
    if (checkedId == R.id.radioFifteen
        || checkedId == R.id.radioTwenty) {
        txtTipOther.setEnabled(false);
        /*
         * Enable the calculate button if Total Amount and No. of
         * People fields have valid values.
         */
        btnCalculate.setEnabled(txtAmount.getText().length() > 0
            && txtPeople.getText().length() > 0);
    }
    if (checkedId == R.id.radioOther) {
        // enable the Other Percentage tip field
        txtTipOther.setEnabled(true);
        // set the focus to this field
        txtTipOther.requestFocus();
    }
}
}
/*
```

```

        * Enable the calculate button if Total Amount and No. of
        * People fields have valid values. Also ensure that user
        * has entered a Other Tip Percentage value before enabling
        * the Calculate button.
        */
        btnCalculate.setEnabled(txtAmount.getText().length() > 0
            && txtPeople.getText().length() > 0
            && txtTipOther.getText().length() > 0);
    }
    // To determine the tip percentage choice made by user
    radioCheckedId = checkedId;
}
});

```

Monitoring key activity in text fields As I mentioned earlier, the 'Calculate' button must not be enabled unless the text fields have valid values. So we have to ensure that the Calculate button will be enabled only if the 'Total Amount', 'No. of People' and 'Other' tip percentage text fields have valid values. The Other tip percentage text field is enabled only if the Other Tip Percentage radio button is selected.

We do not have to worry about the type of values, i.e. whether user entered negative values or alphabetical characters because the `android:numeric` attribute has been defined for the text fields, thus limiting the types of values that the user can enter. We have to just ensure that the values are present.

So we use the static interface `OnKeyListener()`. This will notify us when a key is pressed. The notification reaches us before the actual key pressed is sent to the `EditText` widget.

Look at the code snippet 4 and 5 which deals with key events in the text fields.

The source code comments are quite self explanatory.

Code Snippet 4 of `/src/com/examples/tipcalc/Tipster.java`

Example 1-33.

```

/*
 * Attach a KeyListener to the Tip Amount, No. of People and Other Tip
 * Percentage text fields
 */
txtAmount.setOnKeyListener(mKeyListener);
txtPeople.setOnKeyListener(mKeyListener);
txtTipOther.setOnKeyListener(mKeyListener);

```

Notice that we create just one listener instead of creating anonymous/inner listeners for each textfield. I am not sure if my style is better or recommended, but I always write it in this style if the listeners are going to perform some common actions. Here the common concern for all the text fields is that they should not be empty, and only when they have values should the Calculate button be enabled.

Example 1-34.

```
/*
 * KeyListener for the Total Amount, No of People and Other Tip Percentage
 * fields. We need to apply this key listener to check for following
 * conditions:
 *
 * 1) If user selects Other tip percentage, then the other tip text field
 * should have a valid tip percentage entered by the user. Enable the
 * Calculate button only when user enters a valid value.
 *
 * 2) If user does not enter values in the Total Amount and No of People,
 * we cannot perform the calculations. Hence enable the Calculate button
 * only when user enters a valid values.
 */
private OnKeyListener mKeyListener = new OnKeyListener() {
    @Override
    public boolean onKey(View v, int keyCode, KeyEvent event) {

        switch (v.getId()) {
            case R.id.txtAmount:
            case R.id.txtPeople:
                btnCalculate.setEnabled(txtAmount.getText().length() > 0
                    && txtPeople.getText().length() > 0);
                break;
            case R.id.txtTipOther:
                btnCalculate.setEnabled(txtAmount.getText().length() > 0
                    && txtPeople.getText().length() > 0
                    && txtTipOther.getText().length() > 0);
                break;
        }
        return false;
    }
};
```

In line 18 of code snippet 5, we examine the ID of the View. Remember that each widget has a unique ID as we define it in the main.xml file. These values are then defined in the generated R.java class.

In line 19 and 20, if the key event occurred in the Total Amount or No. of People fields then we check for the value entered in these fields. We are ensuring that the user has not left the fields blank.

In line 24 we check if the user has selected Other radio button, then we ensure that the Other text field is not empty. We also check once again if the Total Amount and No. of People fields are empty.

So the purpose of our KeyListener is now clear - ensure that all text fields are not empty and only then enable the Calculate button.

Listening to Button clicks Next we look at the 'Calculate' and Reset buttons. When the user clicks these buttons, we use the static interface OnClickListener() which will let us know when a button is clicked.

As we did with text fields, just one listener is created and within it we detect which button was clicked. Depending on the button clicked, the `calculate()` and `reset()` methods are called.

Code snippet 6 shows how the click listener is added to the buttons.

Code Snippet 6 of `/src/com/examples/tipcalc/Tipster.java` /* Attach listener to the Calculate and Reset buttons */ `btnCalculate.setOnClickListener(mClickListener); btnReset.setOnClickListener(mClickListener);`

Code snippet 7 shows how to detect which button is clicked by checking for the ID of the View that receives the click event.

Code Snippet 7 of `/src/com/examples/tipcalc/Tipster.java`

Example 1-35.

```
/**
 * ClickListener for the Calculate and Reset buttons.
 * Depending on the button clicked, the corresponding
 * method is called.
 */
private OnClickListener mClickListener = new OnClickListener() {

    @Override
    public void onClick(View v) {
        if (v.getId() == R.id.btnCalculate) {
            calculate();
        } else {
            reset();
        }
    }
};
```

Resetting the application When the user clicks the Reset button, the text fields should be cleared, the default 15% radio button should be selected and any results calculated should be cleared.

Code snippet 8 shows the `reset()` method.

Code Snippet 8 of `/src/com/examples/tipcalc/Tipster.java`

Example 1-36.

```
/**
 * Resets the results text views at the bottom of the screen as well as
 * resets the text fields and radio buttons.
 */
private void reset() {
    txtTipAmount.setText("");
    txtTotalToPay.setText("");
    txtTipPerPerson.setText("");
    txtAmount.setText("");
    txtPeople.setText("");
    txtTipOther.setText("");
}
```

```

        rdoGroupTips.clearCheck();
        rdoGroupTips.check(R.id.radioFifteen);
        // set focus on the first field
        txtAmount.requestFocus();
    }

```

Validating the input to calculate the tip As I said before, we are limiting what type of values the user can enter in the text fields. However, the user could still enter a value of zero in the Total Amount, No. of People and Other Tip Percentage text fields, thus causing error conditions like divide by zero in our tip calculations.

If the user enters zero then we must show an alert popup asking the user to enter non-zero values. We handle this with a method called `showErrorAlert(String errorMessage, final int fieldId)`, but more about it later.

First, look at code snippet 9 which shows the `calculate()` method. Notice how the values entered by the user are parsed as `Double` values.

Checking for zero values Notice line 11 and 16 where we check for zero values. If the user enters zero, then we show an alert popup to warn the user. Next look at line 37, where the Other Tip Percentage text field is enabled because the user selected the Other radio button. Here too, we must check for the tip percentage being zero.

State of radio buttons When the application loads, the 15% radio button is selected by default. If the user changes the selection, then we saw in code snippet 3, in the `OnCheckedChangeListener`, that we assign the ID of the selected radio button to the member variable `radioCheckedId`.

But if the user accepts the default selection, then the `radioCheckedId` will have the default value of `-1`. In short, we will never know which radio button was selected. Offcourse, we know which one is selected by default and could have coded the logic slightly differently, to assume 15% if `radioCheckedId` has the value `-1`. But if you refer the API, you will see that we can call the method `getCheckedRadioButtonId()` on the `RadioGroup` and not on individual radio buttons. This is because the `OnCheckedChangeListener` readily provides us with the ID of the radio button selected.

Showing the results Calculating the tip is simple. If there are no validation errors, the boolean flag `isError` will be false. Look at lines 49 to 51 in code snippet 9 for the simple tip calculations. Next, the calculated values are set to the `TextView` widgets from line 53 to 55.

Code Snippet 9 of `/src/com/examples/tipcalc/Tipster.java`

Example 1-37.

```

/**
 * Calculate the tip as per data entered by the user.
 */
private void calculate() {
    Double billAmount = Double.parseDouble(
        txtAmount.getText().toString());

```

```

Double totalPeople = Double.parseDouble(
    txtPeople.getText().toString());
Double percentage = null;
boolean isError = false;
if (billAmount < 1.0) {
    showErrorAlert("Enter a valid Total Amount.",
        txtAmount.getId());
    isError = true;
}

if (totalPeople < 1.0) {
    showErrorAlert("Enter a valid value for No. of people.",
        txtPeople.getId());
    isError = true;
}

/*
 * If user never changes radio selection, then it means
 * the default selection of 15% is in effect. But its
 * safer to verify
 */
if (radioCheckedId == -1) {
    radioCheckedId = rdoGroupTips.getCheckedRadioButtonId();
}
if (radioCheckedId == R.id.radioFifteen) {
    percentage = 15.00;
} else if (radioCheckedId == R.id.radioTwenty) {
    percentage = 20.00;
} else if (radioCheckedId == R.id.radioOther) {
    percentage = Double.parseDouble(
        txtTipOther.getText().toString());
    if (percentage < 1.0) {
        showErrorAlert("Enter a valid Tip percentage",
            txtTipOther.getId());
        isError = true;
    }
}
}
/*
 * If all fields are populated with valid values, then proceed to
 * calculate the tips
 */
if (!isError) {
    Double tipAmount = ((billAmount * percentage) / 100);
    Double totalToPay = billAmount + tipAmount;
    Double perPersonPays = totalToPay / totalPeople;

    txtTipAmount.setText(tipAmount.toString());
    txtTotalToPay.setText(totalToPay.toString());
    txtTipPerPerson.setText(perPersonPays.toString());
}
}

```

Showing the alerts Android provides the AlertDialog class to show alert popups. This lets us show a dialog with upto three buttons and a message.

Code snippet 10 shows the `showErrorAlert` method which uses this `AlertDialog` to show the error messages. Notice that we pass two arguments to this method - `String` `errorMessage` and `int` `fieldId`. The first argument is the error message we want to show to the user. The `fieldId` is the ID of the field which caused the error condition. After the user dismissed the alert dialog, this `fieldId` will allow us to request the focus on that field, so the user knows which field has the error.

Code Snippet 10 of `/src/com/examples/tipcalc/Tipster.java`

Example 1-38.

```
/**
 * Shows the error message in an alert dialog
 *
 * @param errorMessage
 *         String the error message to show
 * @param fieldId
 *         the Id of the field which caused the error.
 *         This is required so that the focus can be
 *         set on that field once the dialog is
 *         dismissed.
 */
private void showErrorAlert(String errorMessage,
    final int fieldId) {
    new AlertDialog.Builder(this).setTitle("Error")
        .setMessage(errorMessage).setNeutralButton("Close",
            new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog,
                    int which) {
                    findViewById(fieldId).requestFocus();
                }
            })
        .show();
}
```

Conclusion Developing for the Android OS is not so different than developing for any other UI toolkit like Microsoft Windows, X Windows, Java Swing, Adobe Flex, etc. Of course Android has its differences and overall a very good design. The XML layout paradigm is quite cool and useful to build complex UIs using simple XML. The event handling model is simple, feature rich and intuitive to use in code.

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.vidyut.com/sunit/android/tipster.zip>

Binary Download URL

The executable code for this example may be downloaded from this URL: <http://www.vidyut.com/sunit/android/tipster.zip>

Designing a successful Application

2.1 Introduction: Designing a Successful Android application

Colin Wilcox

Discussion

This chapter is about design guidelines for writing imaginative and useful Android applications. There are several recipes that describe specific aspects of successful design. This Preface will list some others.

One purpose of this document is to explain the benefits of developing native Java Android applications in preference to other methods of delivering rich content on mobile devices.

Mobile Phone Market Growth

The mobile phone market is predicted to grow by up to 27% over the next three years. The original dominating platforms such as Symbian and Windows Mobile will count for a largely reduced percentage of the mobile handset market than in past years with recent newcomers Android and Apple iOS (iPhone and iPad) accounting for almost two thirds of the market by 2015.

Industry predictions for each of the key mobile handset platforms are shown in the table below.

Example 2-1.

XXX Make this a real table please

	Android	Apple	Symbian	Blackberry	Microsoft	Other
2010	22.7%	15.7%	37.6%	16.0%	4.2%	3.8%
2011	38.5%	19.4%	19.2%	13.4%	5.6%	3.9%
2012	49.2%	18.9%	5.2%	12.6%	10.8%	3.4%
2015	48.8%	17.2%	0.1%	11.1%	19.5%	3.3%

Predictions for the number of mobile handset sales for each of the main handset platforms (x thousand units), is shown in the table below.

Example 2-2.

	Android	Apple	Symbian	Blackberry	Microsoft	Other
2010	67,225	46,598	111,577	47,452	12,378	11,417
2011	179,873	90,560	89,930	62,600	26,346	18,392
2012	310,088	118,848	32,666	79,335	68,156	21,383
2015	539,318	189,924	661	122,864	215,998	36,133

Requirements of a Native Handset Application There are a number of key requirements needed for successfully delivering any mobile handset application regardless of the platform it will be deployed onto, namely:

- Ease of installation, removal and update on a device.
- An application should address the user needs in a compelling, unique and elegant way.
- Feature rich whilst remaining usable by both novice and expert users.
- An application should be familiar to users who have accessed the same information through other routes, e.g. website.
- Key areas of functionality should be readily accessible.
- Application should have a common look and feel with other native applications on the handset conforming to the target platform standards and style guidelines.
- An application should be stable, scalable, useable and responsive.
- A great application should use the targets capabilities tastefully when it makes the users experience more compelling.

Android Application Design The Android application design will exploit the features and functions unique to the Android OS platform. In general the application will be an Activity based solution allowing independent and controlled access to data on a screen by screen basis. This approach helps to localise potential errors and allows sections of the flow to be readily replaced or enhanced independently of the rest of the application.

Navigation will use a similar approach to that of the Apple iPhone solution in that all key areas of functionality will be accessed from a single navigation bar control. The navigation bar will be accessible from anywhere within the application allowing the user to freely move around the application.

The Android solution will exploit features inherent to Android devices, supporting the touch screen features, support for the hardware button to switch the application into the background and provide support for application switching.

Android provides the ability to jump back into an application at the point where it was switched out. This will be supported, when possible within this design.

The application will use only standard Android user interface controls to make it as portable as possible. The use of themes or custom controls is outside the scope of this design document.

The application will be designed such that it interfaces to a thin layer of RESTFUL web services that provide data in a JSON format. This interface will be the same as used by the Apple iPhone, as well as application written for other platforms.

The application will adopt the Android style and design guidelines wherever possible to provide an application that fits in with other Android application on the device.

Data that is local to each view will be saved when the view is exited and automatically restored with the corresponding user interface controls repopulated when the view is next loaded.

There are a number of important device characteristics that should be considered:

Screen size and density

In order to categorize devices by their screen type, Android defines two characteristics for each device: screen size (the physical dimensions of the screen) and screen density (the physical density of the pixels on the screen, or dpi (dots per inch)). To simplify all the different types of screen configurations, the Android system generalizes them into select groups that make them easier to target.

The design should take into account the most appropriate choices for screen size and screen density when designing the application

By default, your application is compatible with all screen sizes and densities, because the Android system makes the appropriate adjustments to your UI layout and image resources. However, you should create specialized layouts for certain screen sizes and provide specialized images for certain densities, using alternative layout resources, and by declaring in your manifest exactly which screen sizes your application supports.

Input configurations

Many devices provide a different type of user input mechanism, such as a hardware keyboard, a trackball, or a five-way navigation pad. If your application requires a particular kind of input hardware,. However, it is rare that an application should require a certain input configuration.

Device features

There are many hardware and software features that may or may not exist on a given Android-powered device, such as a camera, a light sensor, Bluetooth, a certain version of OpenGL, or the fidelity of the touch screen. You should never assume that a certain feature is available on all Android-powered devices (other than the availability of the standard Android library).

The Android application will provide instances of the two types of menus provided by the Android framework depending on the circumstances.

1. Options menus contain primary functionality that applies globally to the current activity or starts a related activity. It is typically invoked by a user pressing a hard button, often labelled MENU. An Options menu is for any commands that are global to the current activity.
2. Context menus contain secondary functionality for the currently selected item. It is typically invoked by a user's touch & hold on an item. Like on the Options menu, the operation can run either in the current or another activity.

A Context menu is for any commands that apply to the current selection.

The commands on the Context menu that appears when you touch & hold on an item should be duplicated on the activity you get to by a normal press on that item.

- Place the most frequently used operations first.
- Put only the most important commands fixed on the screen.

The system will automatically lay the menus out and provides standard ways for users to access them ensuring that the application will conform to the Android user interface guidelines. In this sense, they are familiar and dependable ways for users to access functionality across all applications.

The Android application will make extensive use of Google's Intent mechanism for passing data between Activity objects. Intents are used not only to pass data between views within a single application but also allow data, or requests, to be passed to external modules. As such much functionality can be adopted by the Android application by embedded functionality from other applications invoked by Intent calls. This reduces the development process and maintains the common look and feel and functionality behaviour across all application.

Data Feeds and Feed Formats

It is not intended to interface directly to any third party data source. The normal approach would be to mitigate the data, from several sources in potentially multiple data formats, through a middleware which then passes data to an application through a series of RESTFUL web service APIs in the form of JSON data streams.

Typically data is provided in such formats as XML, SOAP or some other XML-derived representation. Such representations such as SOAP are heavyweight and as such transferring data from the backend servers in this format increases development time significantly as the responsibility of converting this data into something more manageable falls on either the handset application or an object on the middleware server.

Mitigating the source data through a middleware server also helps to break the dependency between the application and the data. Such a dependency has the disadvantage that if, for some reason, the nature of the data changes or cannot be retrieved the

application may be broken and become unusable and such changes may require the application to be republished. By mitigating the data on a middleware server, the application will continue to work, albeit possibly in a limited fashion, regardless of whether the source data exists or not. The link between the application and the mitigated data will remain. This can be seen from the abstracted diagram below:

2.2 Keeping a Service running while other apps are on display

Ian Darwin

Problem

You want part of your application to continue running in the background while the user switches to interact with other apps.

Solution

Create a Service class to do the background work; start the Service from your main application. Optionally provide a Notification icon to allow the user either to stop the running service or to resume the main application

Discussion

A Service class (`android.app.Service`) runs as part of the same process as your main application, but has the property that it will keep running even if the user switches to another app or even goes to the Home screen and starts up a new app.

As you know by now, Activity classes can be started either by an Intent that matches their Content Provider, or by an intent that mentions them by class name. The same is true for Services. This recipe focuses on starting a service directly; a second recipe will cover starting a service implicitly. The example is taken from JpsTrack, a GPS tracking program for Android. Once you start tracking, you don't want the tracking to stop if you answer the phone or have to look at a map(!), so we made it into a Service. The service is started by the main Activity when you click the Start Tracking button, and stopped by the Stop button, Note that this is so common that `startService()` and `stopService()` are built into the Activity class.

Example 2-3.

```
@Override
public void onCreate(Bundle savedInstanceState) {
    ...
    Intent theIntent = new Intent(this, GpsService.class);
    Button startButton = (Button) findViewById(R.id.startButton);
    startButton.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            startService(theIntent);
        }
    });
}
```

```

        Toast.makeText(Main.this, "Starting", Toast.LENGTH_LONG).show();
    }
});
Button stopButton = (Button) findViewById(R.id.stopButton);
stopButton.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        stopService(theIntent);
        Toast.makeText(Main.this, "Stopped", Toast.LENGTH_LONG).show();
    }
});
...
}

```

The GpsService class directly extends Service, so it has to implement the abstract onBind() method. This is not used when the class is started directly, so it can be a stub method. You will typically override at least the onStartCommand() and onUnbind() methods, to begin and end some activity. Our example starts the GPS service sending us notifications that we save to disk, and we do want that to keep running, hence this Service class.

Example 2-4.

```

public class TrackService extends Service {
    private LocationManager mgr;
    private String preferredProvider;

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        initGPS(); // sets up the LocationManager mgr

        if (preferredProvider != null) {
            mgr.requestLocationUpdates(preferredProvider, MIN_SECONDS * 1000,
                MIN_METRES, this);
            return START_STICKY;
        }
        return START_NOT_STICKY;
    }

    @Override
    public boolean onUnbind(Intent intent) {
        mgr.removeUpdates(this);
        return super.onUnbind(intent);
    }
}

```

XXX Discuss return values from onStartCommand

See Also

Note that a service will not do anything while the phone is suspended; see *the Device from Sleeping*.

2.3 Starting a service after phone reboot

Ashwini Shahapurkar

Problem

You have a service in your app and you want it to start after the phone reboots.

Solution

You need to listen to intent for boot events and start the service when event occurs.

Discussion

Whenever the platform boot is completed, an intent with `android.intent.action.BOOT_COMPLETED` action is broadcasted. You need to register your application to receive this intent. For registering add this to your `AndroidManifest.xml`

Example 2-5.

```
<receiver android:name="ServiceManager">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
</receiver>
```

So you will have **ServiceManager** as broadcast receiver to receive the intent for boot event. The `ServiceManager` class shall be as follows:

Example 2-6.

```
public class ServiceManager extends BroadcastReceiver {

    Context mContext;
    private final String BOOT_ACTION = "android.intent.action.BOOT_COMPLETED";

    @Override
    public void onReceive(Context context, Intent intent) {
        //All registered broadcasts are received by this
        mContext = context;
        String action = intent.getAction();
        if (action.equalsIgnoreCase(BOOT_ACTION)) {
            //check for boot complete event & start your service
            startService();
        }
    }
}
```



```

    }

    private void startService() {
        //here, you will start your service
        Intent mServiceIntent = new Intent();
        mServiceIntent.setAction("com.bootservice.test.DataService");
        mContext.startService(mServiceIntent);
    }
}

```

2.4 Exception Handling

Ian Darwin

Problem

Java has a well-defined exception handling mechanism, but it takes some time to learn to use it effectively without frustrating either users or tech support people.

Solution

Learn about the Exception hierarchy. Learn about Dialogs and Toasts.

Discussion

Java has had two categories of Exceptions (actually of its parent, Throwable) from the beginning, checked and unchecked. In Java Standard Edition the intention was apparently to force the programmer to face the fact that, while certain things could be detected at compile time, others could not. For example, if you were installing a desktop application on a large number of PC's, there would certainly be some on which the disk was almost full, and trying to save data could fail, and others on which some file the application depended upon would go missing, not due to programmer error but to user error, filesystem happenstance, gerbils chewing on the cables, or whatever. So the category of IOException was created as a "checked exception", meaning that the programmer would have to check for it, either by having a `try-catch` clause inside the file-using method or by having a `throws` clause on the method definition. The general rule, which all well-trained Java developers memorize, is the following:

Exception, and all of its subclasses other than RuntimeException or any of its subclasses, is checked. All else is unchecked.

So that means that Error and all of its subclasses are unchecked. If you get a VMError, for example, it means there's a bug in the runtime. Nothing you can do about this as an application programmer. "Nothing here, move along." And RuntimeException subclasses include things like the excessively-long-named `ArrayIndexOutOfBoundsException` - this and friends are unchecked because it is your responsibility to catch them at development time - using (BROKEN XREF TO RECIPE -1 'Unit Testing').

So here is a diagram of the Throwable hierarchy:

Where to Catch Exceptions

The early (over)use of checked exceptions lead a lot of early Java developers to write code that was sprinkled with try/catch blocks, partly because the use of the "throws" clause was not emphasized early enough in some training and books. As Java itself has moved more to Enterprise work and newer frameworks such as Hibernate and Spring have come along emphasizing use of unchecked exceptions, this early problem has been corrected. It is now generally accepted that you want to catch exceptions as close to the user as possible. Code that is meant for re-use - in libraries or even in multiple applications - should not try to do error handling. What it can do is what's called "exception translation" - that is, turning a technology-specific (and usually checked) Exception into a generic, unchecked exception. The basic pattern is:

Example 2-7.

```
public String readTheFile(String f) {
    BufferedReader is = null;
    try {
        is = new BufferedReader(new FileReader(f));
        String line = is.readLine();
        return line;
    } catch (FileNotFoundException fnf) {
        throw new RuntimeException("Could not open file " + f, fnf);
    } catch (IOException ex) {
        throw new RuntimeException("Could not read file " + f, ex);
    } finally {
        if (is != null) {
            try {
                is.close();
            } catch(IOException grr) {
                throw new RuntimeException("Error on close of " + f, grr);
            }
        }
    }
}
```

Note how the use of checked exceptions clutters even this code: it is virtually impossible for the `is.close()` to fail, but since you want to have it in a finally block (to ensure it gets tried if the file was opened but then something went wrong), you have to have an additional try/catch around it. SO: Checked Exceptions are generally a bad thing, should be avoided in new APIs, and should be paved over with unchecked exceptions when using code that requires them.

There is an opposing view, espoused by the official Oracle site and others. Al Sutton points out that "Checked exceptions exist to force developers to acknowledge that an error condition can occur and that they have thought about how they want to deal with it. In many cases there may be little that can be done beyond logging and recovery, but it is still an acknowledgment by the developer that they have considered what should

happen with type of error. The example shown ... stops callers of the method from differentiating between when a file doesn't exist (and thus may need to be re-fetched), and when there is a problem reading the file (and thus the file exists but is unreadable), which are two different types error conditions."

Android, wishing to be faithful to the Java API, has a number of these checked exceptions (including the ones shown in the example), so they should be treated the same way.

What to do with Exceptions

Always report.

Dialog or Toast.

Examples of both.

2.5 Sending/Receive broadcast message

Vladimir Kroz

Problem

You want to make an activity which receives a simple broadcast messages sent by another activity

Solution

1. Setup a broadcast receiver
2. Instantiate message receiver object
3. Create IntentFilter
4. Register your receiver with an activity which must receive broadcast message

Discussion

1. Setup broadcast receiver

Example 2-8.

```
// Instantiate message receiver object. You should
// create this class by extending android.content.BroadcastReceiver
// The method onReceive() of this class will be called when broadcast is sent
MyBroadcastMessageReceiver _bcReceiver = new MyBroadcastMessageReceiver();

// Create IntentFilter
IntentFilter filter = new IntentFilter(
MyBroadcastMessageReceiver.class.getName());

// And register your receiver with your activity which must receive broadcast message
```

```
// Now whenever this type of message is generated somewhere in the system -  
// _bcReceiver.onReceive() method will be called within main thread of myActivity  
myActivity.registerReceiver(_bcReceiver, filter);
```

2. Publish broadcast event

Example 2-9.

```
Intent intent = new Intent(  
MyBroadcastMessageReceiver.class.getName());  
intent.putExtra("some additional data", choice);  
someActivity.sendBroadcast(intent);
```

2.6 Android's Application Object as a "Singleton"

Adrian Cowham

Problem

In the all too common case when you need to access "global" data from within your Android app, the best solution is to use subclass `android.app.Application` and treat it as a Singleton with static accessors.

Solution

Every Android app is guaranteed to have exactly one `android.app.Application` instance for the lifetime of the app. If you choose to subclass `android.app.Application`, Android will create an instance of your class and invoke the `android.app.Application` life cycle methods on it. Because there's nothing preventing you from creating another instance of your subclassed `android.app.Application`, it isn't a genuine Singleton, but it's close enough.

Having objects such as Session Handlers, Web Service Gateways, or anything that your application only needs a single instance of, globally accessible will dramatically simplify your code. Sometimes these objects can be implemented as singletons, and sometimes they cannot because they require a Context instance for proper initialization. In either case, it's still valuable to add static accessors to your subclassed `android.app.Application` instance so that you consolidate all globally accessible data in one place, have guaranteed access to a Context instance, and easily write "correct" singleton code without having to worry about synchronization.

Discussion

When writing your Android app you may find it necessary to share data and services across multiple Activities. For example, if your app has session data, such as the currently logged in user, you will likely want to expose this information. When developing on the Android platform, the pattern for solving this problem is to have your an-

droid.app.Application instance own all global data, and then treat your Application instance as a Singleton with static accessors to the various data and services.

When writing an Android app, you're guaranteed to only have one instance of the android.app.Application class so it's safe (and recommended by the Google Android team) to treat it as a Singleton. That is, you can safely add a static getInstance() method to your Application implementation. Below is an example.

Example 2-10.

```
public class AndroidApplication extends Application {

    private static AndroidApplication sInstance;

    private SessionHandler sessionHandler;

    public static AndroidApplication getInstance() {
        return sInstance;
    }

    public SessionoHandler getSessionHandler()
        return sessionHandler;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        sInstance = this;
        sInstance.initializeInstance();
    }

    protected void initializeInstance() {
        // do all you initialization here
        sessionHandler = new SessionHandler(
            this.getSharedPreferences( "PREFS_PRIVATE", Context.MODE_PRIVATE ) );
    }
}
```

This isn't the classical Singleton implementation but given the constraints of the Android framework, this is the closest thing we have, it's safe, and it works.

Using this technique in my app has simplified and cleaned up our implementation. Also, it has made developing tests much easier. Using this technique in conjunction with the *Robolectric* testing framework, mock out our entire execution environment in a straight forward fashion.

Also, don't forget to add the application declaration to your `AndroidManifest.xml` file.

Example 2-11.

```
<application android:icon="@drawable/app_icon"
    android:label="@string/app_name"
    android:name="com.company.abc.AbcApplication">
```

See Also

<http://mytensions.blogspot.com/2011/03/androids-application-object-as.html>

2.7 Keeping data when the user rotates the device

Ian Darwin

Problem

When the user rotates the device, Android will normally destroy and re-create the current Activity. You want to keep some data across this destroy-re-create cycle, but all the fields in your activity are lost during it.

Solution

There are several approaches. If all your data are either primitive types, Strings, or are Serializable, you can save them in `onSaveInstanceState()` into the Bundle that is passed in.

There is also a solution that lets you return a single arbitrary Object Implement `onRetainNonConfigurationInstance()` in your activity to save some values; call `getLastNonConfigurationInstance()` near the end of your `onCreate()` to see if there is a previous saved value and, if so, assign your fields accordingly.

Discussion

Using `onSaveInstanceState()`

See the recipe (BROKEN XREF TO RECIPE -1 'Normal Life-cycle Methods').

Using `onRetainNonConfigurationInstance()`

The `getLastNonConfigurationInstance()` method's return type is Object, so you can return any value you want from it. You might want to create a Map or write an inner class to store the values in, but it's often easier just to pass a reference to the current Activity, e.g, using `this`.

Example 2-12.

```
/** Returns arbitrary single token object to keep alive across
 * the destruction and re-creation of the entire Enterprise.
 */
@Override
public Object onRetainNonConfigurationInstance() {
    return this;
}
```

The above method will be called when Android destroys your main activity. Suppose you wanted to keep a reference to another object that was being updated by a running Service, that is referred to by a field in your Activity. There might also be boolean to indicate if the service is active. In the above, we return a reference to the Activity, from which all of its fields can be accessed (even private fields, of course, since the outgoing and incoming Activity objects are of the same class). In my geotracking app JPSTrack, for example, I have a FileSaver class which accepts data from the Location Service; I want it to keep getting the location, and saving it to disk, in spite of rotations, rather than having to restart it every time the screen rotates. Rotation is unlikely if your device is anchored in a car dash mount (we hope), but quite likely if the person not driving, or a pedestrian, is taking pictures or other notes while geotracking.

After Android creates the new instance, if of course calls onCreate() to notify the new instance that it has been created. In onCreate you typically do constructor-like actions such as initializing fields and assigning event listeners. Well, you still need to do those, so leave them alone. Near the end of onCreate(), however, you will add some code to get the old instance, if there is one, and get some of the important fields from it. The code should look something like this:

Example 2-13.

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

    saving = false;
    paused = false;

    // lots of other initializations...

    // Now see if we just got interrupted by e.g., rotation
    Main old = (Main) getLastNonConfigurationInstance();
    if (old != null) {
        saving = old.saving;
        paused = old.paused;

        // this is the most important line: keep saving to same file!
        fileSaver = old.fileSaver;
        if (saving) {
            fileNameLabel.setText(fileSaver.getFileName());
        }
    }
    return;
}

// I/O Helper
fileSaver = new GPSFileSaver(...);
}
```

The fileSaver object is the big one, the one we want to keep running, and not re-create every time. If we don't have an old instance, we create the fileSaver only at the very end

of `onCreate()`, since otherwise we'd be creating a new one just to replace it with the old one, which is at least bad for performance.

When the `onCreate` method finishes, we hold no reference to the old instance, so it should be eligible for Java GC.

The net result is that the Activity appears to keep running nicely across screen rotations, despite the re-creation.

An alternative possibility is to set `android:configChanges="orientation"` in your `AndroidManifest.xml`, but this is a bit riskier.

See Also

[Recipe 2.6](#)

Source Download URL

The source code for this example may be downloaded from this URL: <http://projects.darwinsys.com/jpstrack/android>

2.8 Creating a Responsive Application using Threads

Amir Alagic

Problem

How to create a responsive application.

Solution

A successful Android application has to be responsive and since all Activities, Broadcast Receivers and Services are running on the main application thread (UI thread), any time-consuming operations will block the UI. If such an operation takes more than five seconds, in Activities, the Android OS will show the standard "Application ... is not responding" error alert/dialog, often called an ANR for short.

Discussion

To make your application responsive while some time-consuming operations are running on Android OS you have a few options. If you already know Java then you know that you can create class that extends `Thread` class and overrides `public void run()` method and then call `start` method on that object to run some time-consuming process. If your class already extends another class you can implement `Runnable` interface. And another different approach is to create your own class that extends Androids `AsyncTask` class but we will talk about `AsyncTask` in another recipe.

First we will discuss usage of `Thread` class.

Example 2-14.

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

        Thread thread = new Thread(new Runnable(){
            public void run() {
                getServerData();
            }
        });
        thread.start();
    }
}
```

As you see, when we start our activity in `onCreate` method we create a thread object that is constructed with a `Runnable` object. The `Runnable` method `run()` will be executed after we call `start()` method on the thread object. From here you can call another method or few other methods and operations that are time-consuming and that would otherwise block the main thread and make your application look unresponsive.

Often when we are done with the thread we get results that we want to present to the application user. If you try to update the GUI from the thread that you started (not the main thread) your application will crash. You can read error messages and see that the problem is in fact a thread other than the main UI thread tried to change UI on the main thread.

Of course it is possible to change UI with help of a `Handler` class.

In your activity create a final `Handler` object and within your `public void run()` method call its `post()` method.

Example 2-15.

```
final Handler handler = new Handler();

Thread thread = new Thread(new Runnable(){
    public void run() {
        handler.post(new Runnable(){
            public void run() {
                //textView is a UI control created on the main thread
                textView.setText("Change Text");
            }
        });
    }
});
thread.start();
```

Threads created and started in this way will continue to run even if the user leaves your application. To be sure that your thread(s) stop when the user leaves your application, before you call start() method on thread object set thread as daemon thread.

Example 2-16.

```
thread.setDaemon(true);
```

Beside that sometimes it can be useful to name the thread.

You can give a name to your thread(s) when you create the thread object

Example 2-17.

```
Thread thread = new Thread();  
Thread thread = new Thread(runnable, "ThreadName1");
```

or you can call setName() method on thread object.

Example 2-18.

```
thread.setName("ThreadName2");
```

These names will not be visible to the user, but will show up in various diagnostic logs, to help you find which thread is causing problems.

2.9 Eating Too Much CPU Time In The UI Causes A Nasty Result

Daniel Fowler

Problem

Any time consuming or intensive code in an event listener will make the UI appear slow and potentially cause an Application Not Responding error.

Solution

The main UI thread should just be that, keeping the UI going and updated. Any heavy duty, regularly executing or potentially slow code needs shoving to a background task, the Android AsyncTask class is ideal for that job.

Discussion

When Android starts an application it assigns it to run on a single main thread, also known as the User Interface (UI) thread. The UI thread, as the name suggests handles the interface, posting events for the various widgets on the screen and then running the code in the listeners for those events. If the code executing from a listener takes too long it will slow down event processing, including the UI events that tell widgets to redraw, the UI becomes unresponsive. The slow code running on the UI thread ultimately results in the Application Not Responding (ANR) error. This can occur if screen

elements do not get a chance to process their pending requests after about five seconds. When an ANR appears the user can then forcibly close the App (and then probably remove it which would be a nasty result).

Tasks that can chew up UI CPU cycles included:

- Accessing large amounts of data, especially through slow connections or peripherals.
- Jittery connections when accessing networks and Internet services.
- The need to run some recurring code, for animation, polling, timing.
- Parsing large data files or XML files.
- Creating, reading, updating, deleting large databases records.
- Code with too many loops.
- Intensive graphics operations.

In the following code an Activity with a TextView and Button is calling a method, `ThisTakesAWhile()`, which mimics a slow process. The code tries to keep the UI updated on progress by updating the TextView. This is the layout for the screen:

Example 2-19.

```
<?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="wrap_content" >
    <Button android:id="@+id/button1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="GO"
        android:textSize="14sp"
        android:textStyle="bold" />
    <TextView android:id="@+id/textview1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Press GO to start."
        android:textSize="14sp"
        android:textStyle="bold"
        android:gravity="center_horizontal" />
</LinearLayout>
```

And the code:

Example 2-20.

```
public class main extends Activity {
    TextView tv; //for class wide reference to update status
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

```

        //get the references to on screen items
        tv=(TextView) findViewById(R.id.textview1);
        //handle button presses
        findViewById(R.id.button1).setOnClickListener(new doButtonClick());
    }

    class doButtonClick implements OnClickListener {
        public void onClick(View v) {
            tv.setText("Processing, please wait.");
            ThisTakesAWhile();
            tv.setText("Finished.");
        }
    }

    private void ThisTakesAWhile() {
        //mimic long running code
        int count = 0;
        do{
            SystemClock.sleep(1000);
            count++;
            tv.setText("Processed " + count + " of 5.");
        } while(count<5);
    }
}

```

When this code is run and the Button pressed the text only changes after five seconds to "Finished". The messages that are meant to provide feedback are never shown. This illustrates how the UI can be blocked by code executing on the main thread. In fact bash the button a few times and an ANR occurs.

The solution is to run the time consuming code away from UI events. To help achieve this there are standard Java classes, such as the `Timer`, the `Thread` and the `ScheduledThreadPoolExecutor`. Android also provides helpful classes with the `CountDownTimer`, the `Handler` and `Service`. Ideally for potentially slow operations we want to start them from the UI in a background thread, get regular reports on progress, cancel them if need be and get a result when they have finished. In Android the `AsyncTask` class does all of that easily without having to crank out a lot of code.

With `AsyncTask` the time consuming code is placed into a `doInBackground()` method, there are `onPreExecute()` and `onPostExecute()` methods that can be overridden (for pre and post task work), and an `onProgressUpdate()` method can be overridden to provide feedback. The background task can be cancelled by calling the `cancel()` method (causing the overridden `onCancelled()` to execute). The above example is changed to a use an `AsyncTask` object. First the `count` variable is moved to module level and a couple more module variables are added:

Example 2-21.

```

int count;           //number of times process has run, used for feedback
boolean processing; //defaults false, set true when the slow process starts
Button bt;          //used to update button caption

```

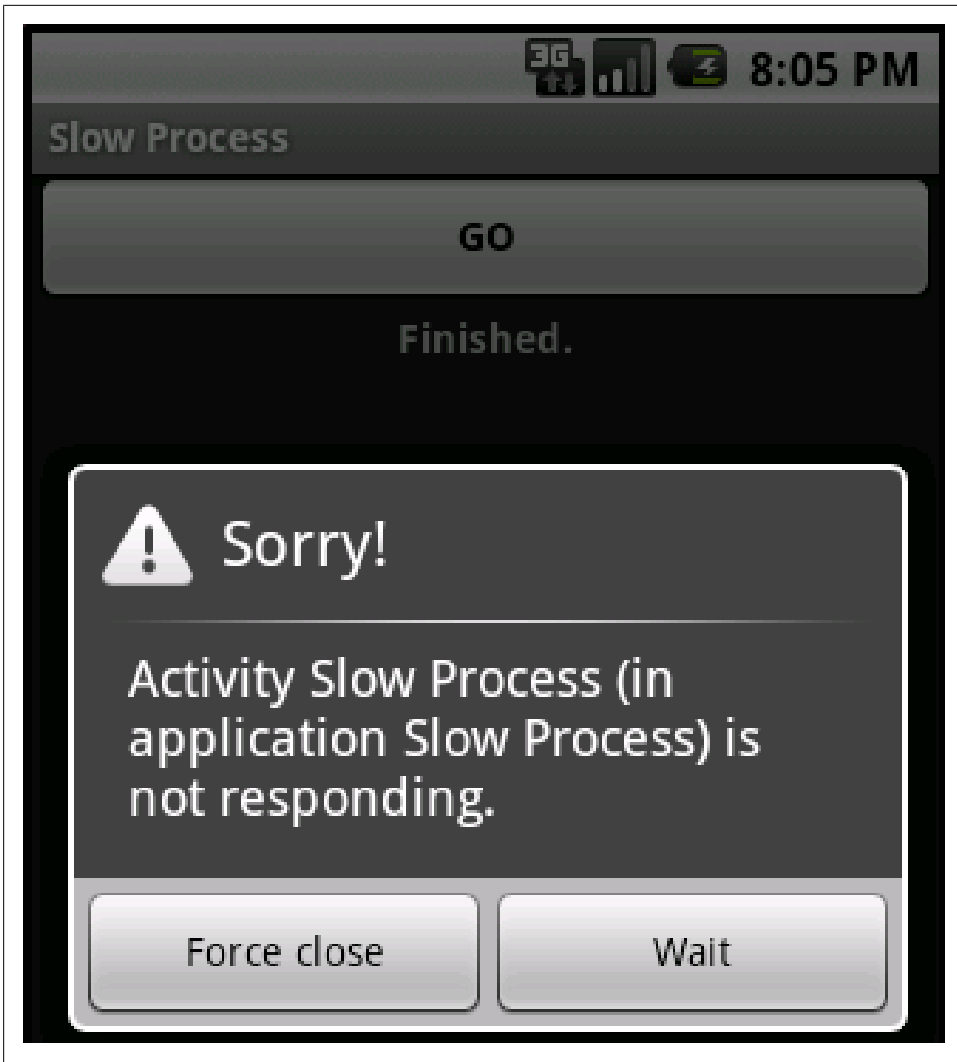


Figure 2-1.

The button reference is set up in `onCreate()`:

Example 2-22.

```
bt=(Button) findViewById(R.id.button1);
```

The method used to run the slow process is replaced with an `AsyncTask` object:

Example 2-23.

```
//AsyncTask can take any type here arg0, arg1, arg2 all integers  
class ThisTakesAWhile extends AsyncTask<Integer, Integer, Integer> {
```

```

int numcycles; //total number of times to execute process
@Override
protected void onPreExecute(){
    //Executes in UI thread before task begins
    //Can be used to set things up in UI such as showing progress bar
    count=0; //count number of cycles
    processing=true;
    tv.setText("Processing, please wait.");
    bt.setText("STOP");
}
@Override
protected Integer doInBackground(Integer... arg0) {
    //Runs in a background thread
    //Used to run code that could block the UI
    numcycles=arg0[0]; //Run arg0 times
    //Need to check isCancelled to see if cancel was called
    while(count < numcycles && !isCancelled()) {
        //wait one second (simulate a long process)
        SystemClock.sleep(1000);
        //count cycles
        count++;
        //signal to the UI (via onProgressUpdate)
        //class arg1 determines type of data sent
        publishProgress(count);
    }
    //return value sent to UI via onPostExecute
    //class arg2 determines result type sent
    return count;
}
@Override
protected void onProgressUpdate(Integer... arg1){
    //called when background task calls publishProgress
    //in doInBackground
    if(isCancelled()) {
        tv.setText("Cancelled! Completed " + arg1[0] + " processes.");
    } else {
        tv.setText("Processed " + arg1[0] + " of " + numcycles + ".");
    }
}
@Override
protected void onPostExecute(Integer result){
    //result comes from return value of doInBackground
    //runs on UI thread, not called if task cancelled
    tv.setText("Processed " + result + ", finished!");
    processing=false;
    bt.setText("GO");
}
@Override
protected void onCancelled() {
    //run on UI thread if task is cancelled
    processing=false;
    bt.setText("GO");
}
}

```

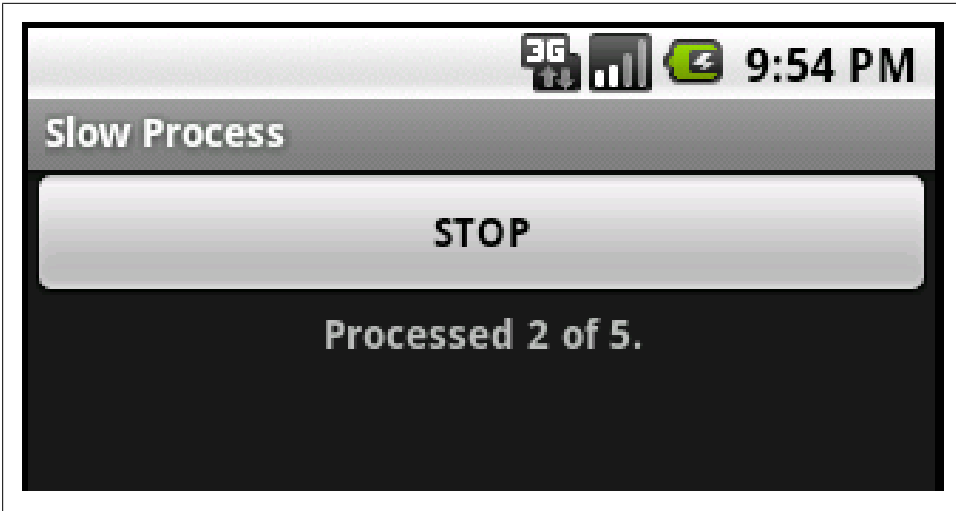


Figure 2-2.

The button click handler now uses the new object. The `execute()` method starts the ball rolling. While the `ThisTakesAWhile` object executes the slow process the button can be used to stop it with `cancel()`:

Example 2-24.

```
class doButtonClick implements OnClickListener {
    ThisTakesAWhile ttaw;//defaults null
    public void onClick(View v) {
        if(!processing){
            ttaw = new ThisTakesAWhile();
            ttaw.execute(5);    //loop five times
        } else {
            ttaw.cancel(true);
        }
    }
}
```

With the slow process wrapped up in the `AsyncTask` object the UI thread is freed from waiting and the feedback messages get displayed to tell the users what is happening:

When `cancel` is called the background process may not stop immediately. For example two cycles have completed and the third starts just as `cancel` is called. Then the third cycle could still complete. In some scenarios the third process may need to be thrown away, rolled back, or handled differently. With version 3.0 of Android `AsyncTask` was extended to provide an `onCancelled(Object)` method that can be overridden, also called when `cancel()` is executed. This version of `onCancelled` takes the same object as that returned by `doInBackground`. This allows the program to determine the state of the

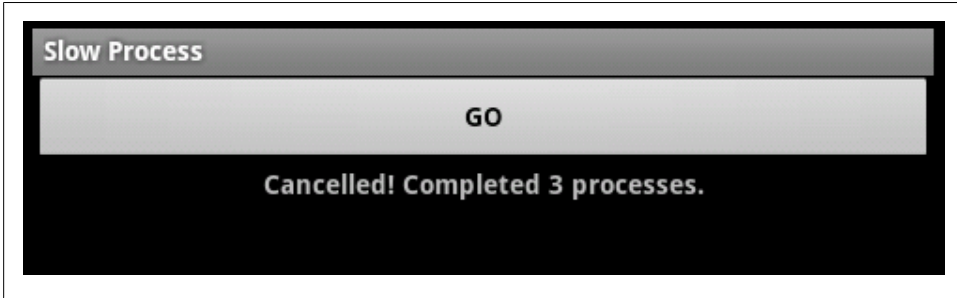


Figure 2-3.

background task when cancel was called. If this version was being used then the overridden *onCancelled* could be:

Example 2-25.

```
@Override
protected void onCancelled(Integer result) {
    //run on UI thread if task is cancelled
    //result comes from return value of doInBackground
    tv.setText("Cancelled called after "+ result + " processes.");
    processing=false;
    bt.setText("GO");
}
```

Thus in the above scenario it would be known that cancelled was called after two process completed even though the third then completed.

This recipe has shown how to move code that could potentially frustrate users and cause poor performance into the useful *AsyncTask* object. For more details on *AsyncTask* see the Android Reference web site.

See Also

<http://developer.android.com/reference/android/os/AsyncTask.html>

2.10 AsyncTask: Do background tasks and modify the GUI

Roberto Calvo Palomino

Problem

If you run a thread to execute some source code, you can't use or change any graphical widget from this thread and you don't know when the thread has finished.

Solution

You can use `AsyncTasks` to develop and run background tasks and modify the GUI from this `AsyncTask`. It's often used to show progress dialog when the data is downloaded

Discussion

(NOT FINISHED YET)

Example 2-26.

```
private class getDataTask extends AsyncTask<Void, Void, Void> {

    ProgressDialog pd;

    // In this method you can modify any graphical widget
    protected void onPreExecute() {
        pd = ProgressDialog.show(Main.this, "Please wait...", "Retrieving data ...", true);
    }

    @Override
    protected Void doInBackground(Void... params) {

    }

    // In this method you can modify any graphical widget
    protected void onPostExecute(Void unused) {
        if (!isFinishing())
        {
            pd.dismiss();
        }
    }
}
```

See Also

<http://developer.android.com/reference/android/os/AsyncTask.html>

2.11 Monitoring the Battery Level of your Android Device

Pratik Rupwal

Problem

Requirement of detecting the battery level for notifying the user when the battery level goes down a certain threshold.

Solution

A broadcast receiver to receive the broadcast message sent when battery status changes can identify the battery level and can be used for issuing alerts for the users.

Discussion

Sometimes we need to show an alert to the user when the battery level of the device goes below a certain limit. The following code sets the broadcast message to be sent whenever the battery level changes and creates a broadcast receiver to receive the broadcast message which can alert the user when battery gets discharged below a certain level.

Example 2-27.

```
public class MainActivity extends Activity {

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        /**This registers the receiver for a broadcast message to be sent when the battery level is changed*/

        this.registerReceiver(this.myBatteryReceiver,
            new IntentFilter(Intent.ACTION_BATTERY_CHANGED));

        /**Intent.ACTION_BATTERY_CHANGED can be replaced with Intent.ACTION_BATTERY_LOW for broadcasting
        a message only when battery level is low rather than sending a broadcast message every time battery level ch
        */

        private BroadcastReceiver myBatteryReceiver
        = new BroadcastReceiver(){

            @Override
            public void onReceive(Context arg0, Intent arg1) {
                // TODO Auto-generated method stub
                int bLevel = arg1.getIntExtra("level", 0);// this variable gets the battery level in integer
                Log.i("Level", ""+bLevel);
            }
        };
    }
}
```

2.12 Splash Screens in Android: Method 1

Rachee Singh

Problem

Most applications require an introductory opening screen.

Solution

Such introductory screens are called Splash screens. An activity that is finished within a span of 2-3 seconds or dismissed on the click of a button is a splash screen.

Discussion

The splash screen displays for 2 seconds and then the main activity of the application appears. For this, we add a java class that displays the splash screen. It uses a thread to wait for 2 seconds and then it uses an intent to start the next activity.

Example 2-28.

```
public class SplashScreen extends Activity {
    private long ms=0;
    private long splashTime=2000;
    private boolean splashActive = true;
    private boolean paused=false;
    @Override
    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.splash);
        Thread mythread = new Thread() {
            public void run() {
                try {
                    while (splashActive && ms < splashTime) {
                        if(!paused)
                            ms=ms+100;
                        sleep(100);
                    }
                } catch(Exception e) {}
                finally {
                    Intent intent = new Intent(SplashScreen.this, Splash.class);
                    startActivity(intent);
                }
            }
        };
        mythread.start();
    }
}
```

The layout of the splash activity, splash.xml is like:

Example 2-29.

```
<?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">
    <ImageView android:src="@drawable/background"
        android:id="@+id/image"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content" />
        <ProgressBar android:id="@+id/progressBar1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/image"
            android:layout_gravity="center_horizontal">
        </ProgressBar>
    </LinearLayout>
```

The splash screen looks like:

In 2 seconds, this activity leads to the next activity, which is the standard Hello World Android activity :

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rE_SQKgad5LZGY1N2RjYzQtZGQxNC00Njk5LWIyM2ItNDdlN2IwZjg4MmVj&hl=en_US&authkey=COOL9NwM

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rE_SQKgad5LMDQyYjE0ZTMtZGZkMy00ZWE2LTk3YWYtM2U2YjdmMTc1Yjkw&hl=en_US&authkey=COGIiroM

2.13 Splash Screens in Android: Method 2

Rachee Singh

Problem

Most applications require an introductory opening screen.

Solution

Such introductory screens are called Splash screens. An activity that is finished within a span of 2-3 seconds or dismissed on the click of a button is a splash screen.

Discussion

The splash screen displays until the Menu button the Android device is not pressed and then the main activity of the application appears. For this, we add a java class that displays the splash screen.

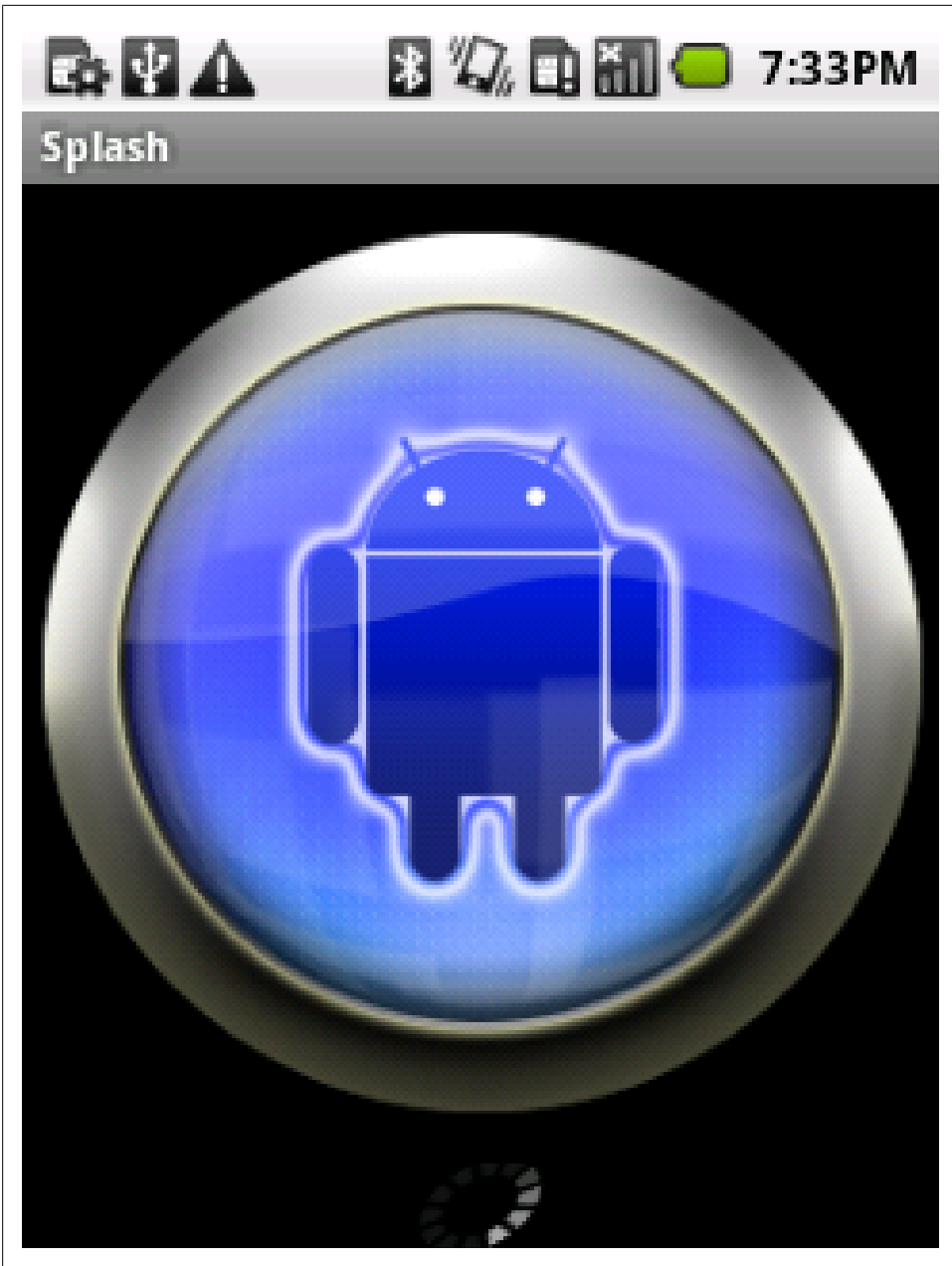


Figure 2-4.

We check for the pressing of Menu key by checking the Key code and then finishing the activity.

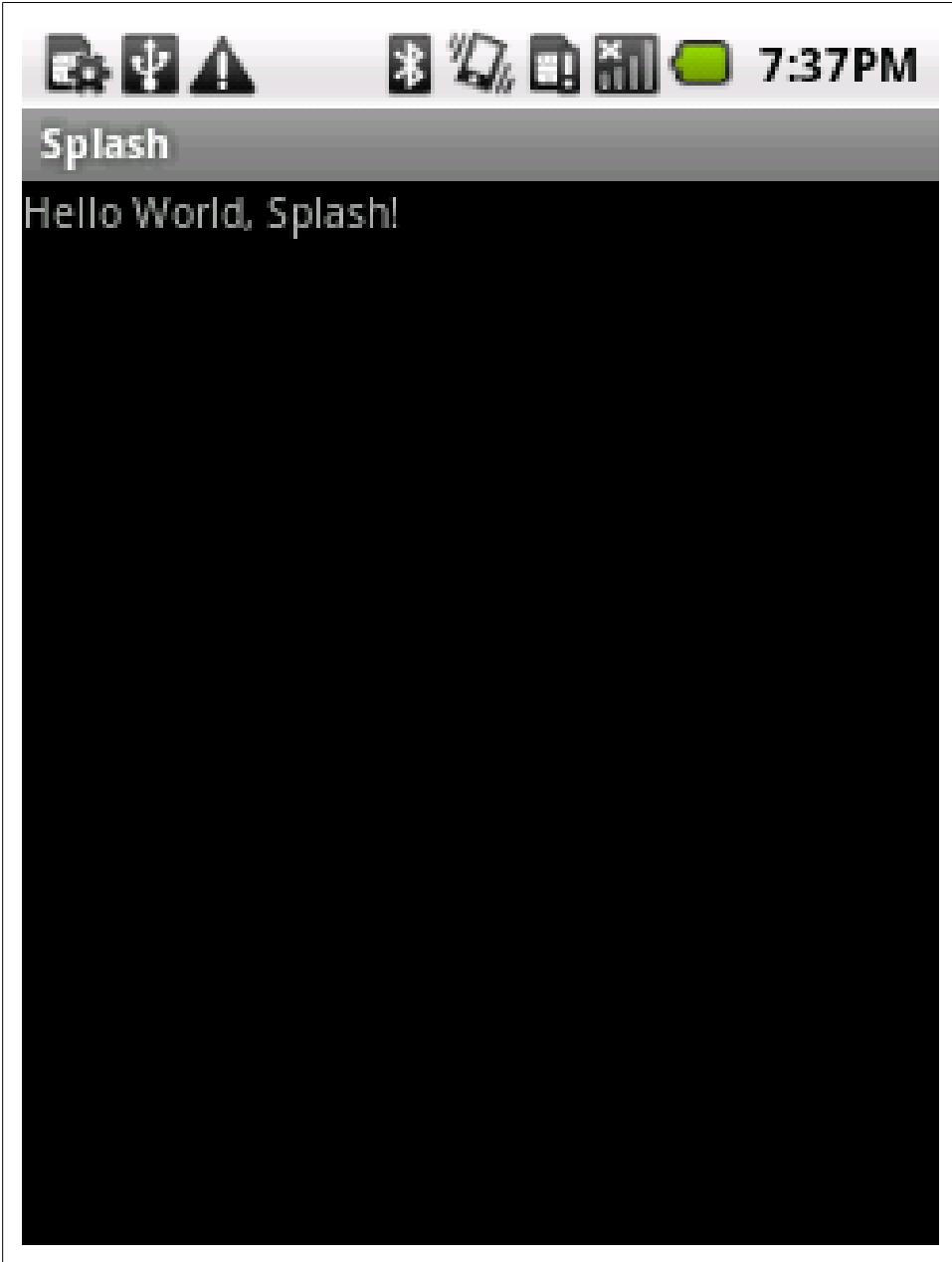


Figure 2-5.

Example 2-30.

```
public class SplashScreen extends Activity {
    private long ms=0;
    private long splashTime=2000;
    private boolean splashActive = true;
    private boolean paused=false;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.splash);
    }

    public boolean onKeyDown(int keyCode, KeyEvent event) {
        super .onKeyDown(keyCode, event);
        if(KeyEvent.KEYCODE_MENU == keyCode) {
            Intent intent = new Intent(SplashScreen.this, Splash.class);
            startActivity(intent);
        }
        if(KeyEvent.KEYCODE_BACK == keyCode) {
            finish();
        }
        return false;
    }
}
```

The layout of the splash activity, splash.xml is like:

Example 2-31.

```
<?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">
    <ImageView android:src="@drawable/background"
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <ProgressBar android:id="@+id/progressBar1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/image"
        android:layout_gravity="center_horizontal">
    </ProgressBar>
</LinearLayout>
```

The splash screen looks like:

In 2 seconds, this activity leads to the next activity, which is the standard Hello World Android activity.

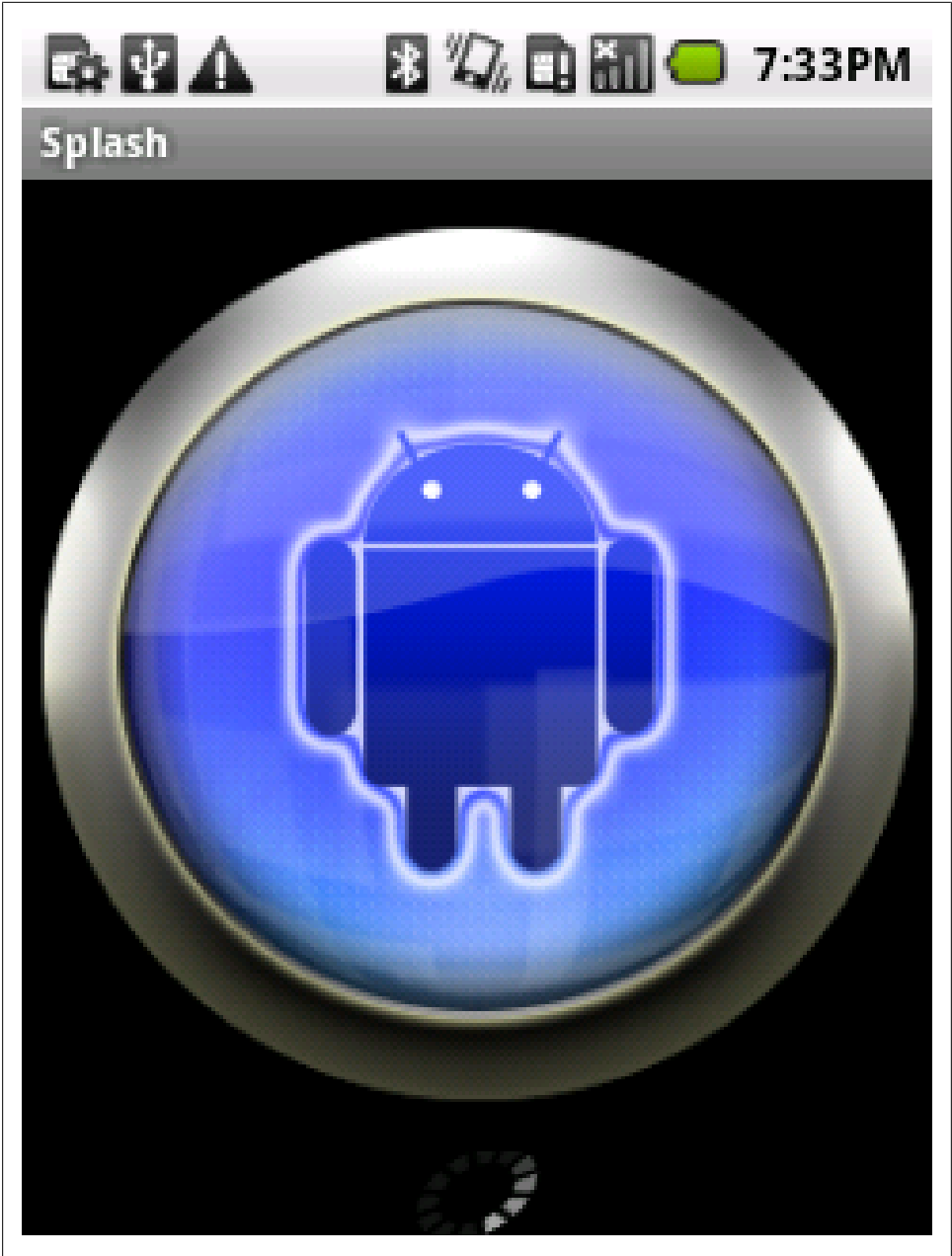


Figure 2-6.

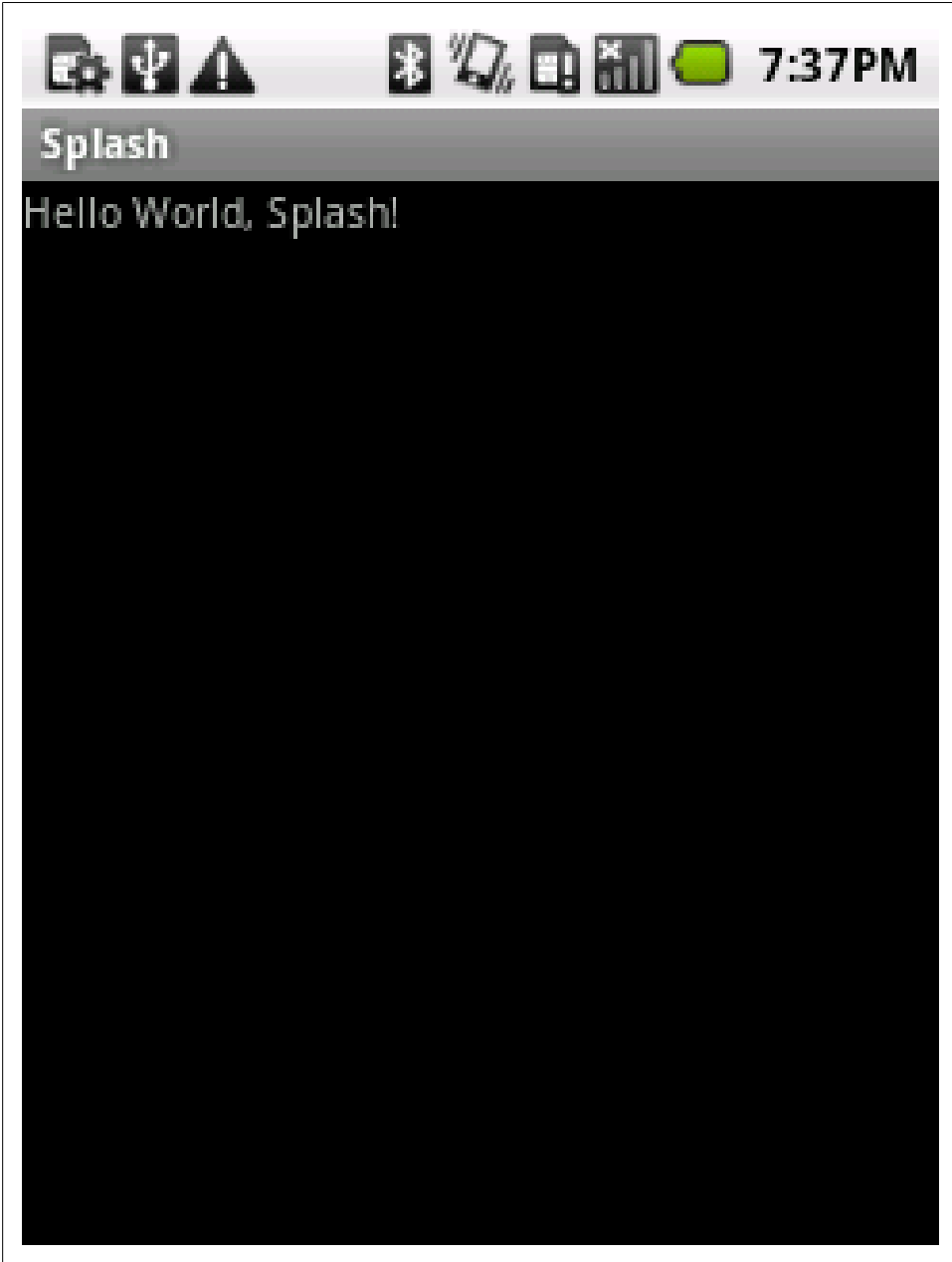


Figure 2-7.

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LOTQ1Nzg1YWItN2UxMy00OTI1LTliZGMtNjRkNzFkZTMzMTE3&hl=en_US&authkey=ClyI8_8F

2.14 Designing a Conference/*Camp/Hackathon App

Ian Darwin

Problem

You want to design an app for use at a Conference, BarCamp, or Hackathon.

Solution

See the checklist below.

Discussion

A good conference app needs at least the following functions:

- Building Map, showing the rooms where meetings are held, food service, wash-rooms, emergency exits, and so on. Extra points if you provide a visual slider for moving up or down if your conference takes place on more than one floor or level in the building (think about a 3D fly-through of San Francisco's Moscone Center, including the huge escalators).
- Exhibit Hall map (if there is a show floor, have a map and an easy way to find a given exhibitor). Ditto for poster papers if your conference features these.
- Schedule view. Highlight changes in red as they happen; additions, last-minute cancellations, room changes.
- If your conference has BOFs, you might want a signup button and maybe even a "Suggest a new BOF" activity.
- Local Area Map. Something more detailed than the standard Map functions. Add folklore, points of interest, navigation shortcuts, etc. Limit it to a few blocks so you can get the details right. A University Campus is about the right size.
- City Overview map - again, not the Google map, but an artistic, neighborhood/zone view with just the highlights.
- Tourist Attractions within an hour of the site. Your mileage may vary.
- Food Finder. People always get tired of convention food and set out on foot to find something greater to eat.

- Friend Finder. If Latitude were open you could tie into it. If it's a security conference, implement this functionality yourself.
- Private voice chat. If it's a small security gathering, provide a SIP server on a well-connected host, with carefully controlled access; it should be possible to have almost walkie-talkie-like service.
- Signups for impromptu group formation for trips to the tourist attractions or any other purpose.
- Functionality to post comments to Twitter, Facebook, LinkedIn.
- Note-taking! Many people will have Android on large-screen tables, so a "Notepad" equivalent, optionally linked to the session the notes are taken in, will be useful.
- FastFood - a way of signalling your chosen friends that you want to eat (at a certain time, in so many minutes, RIGHT NOW), with (type of food, restaurant name), and seeing if they're also interested - interactive.

See Also

The rest of the book shows how to implement most of these functions.

2.15 Implementing Autocompletion in Android.

Rachee Singh

Problem

Save the user from typing entire words, instead auto-complete the entries.

Solution

Using the widget: `AutoCompleteTextView` that acts as something in between `EditText` and a `Spinner`, enabling auto completion.

Discussion

This layout includes a `TextView` which supports auto-completion. Auto-completion is done using a `AutoTextCompleteTextView` widget. Here's what the layout XML code looks like:

Example 2-32.

```
<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:id="@+id/field"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
  />
  <AutoCompleteTextView
    android:id="@+id/autocomplete"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:completionThreshold="2"/>

```

```
</LinearLayout>
```

The `completionThreshold` field in the `AutoCompleteTextView` sets the minimum number of characters that the user has to enter in the `TextView` so that auto-completion options corresponding to his/her input start showing up.

The Activity (in which we are implementing auto-completion) should implement `TextWatcher` so we can override `onTextChanged()` method:

Example 2-33.

```
public class AutoComplete extends Activity implements TextWatcher {
```

We would need to override the unimplemented methods: `onTextChanged`, `afterTextChanged` and `beforeTextChanged`.

We also require 3 fields:

1. A handle on to the `TextView`
2. A handle on to the `AutoCompleteTextView`
3. A list of `String` items within which the auto-completion would happen.

Example 2-34.

```
private TextView field;
private AutoCompleteTextView autocomplete;
String autocompleteItems [] = {"apple", "banana", "mango", "pineapple", "apricot", "orange", "pear", "grapes"}
```

Example 2-35.

```
@Override
public void onTextChanged(CharSequence arg0, int arg1, int arg2, int arg3) {
    field.setText(autocomplete.getText());
}
```

In the `onCreate` method of the same activity, we get a handle on the `TextView` and the `AutoCompleteTextView` components of the layout. To the `AutoCompleteTextView` we will set a `String` adapter.

Example 2-36.

```
setContentView(R.layout.main);
field = (TextView) findViewById(R.id.field);
autocomplete = (AutoCompleteTextView) findViewById(R.id.autocomplete);
```

```
autocomplete.addTextChangedListener(this);
autocomplete.setAdapter(new ArrayAdapter<String>(this, android.R.layout.simple_dropdown_item_1line, autocomp
```

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LYzVkOTdLOGUtODg5My00ZTRmLWlyNTYtMDdiMzA0NjhiNGRk&hl=en_US

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LY2Q0ZjhkZGYtOTRhNi00YWE5LTk0MDYtNDRmMWQxYTE2YTU4&hl=en_US

2.16 Using Google Analytics in Android Application

Ashwini Shahapurkar

Problem

Often developers want to track their application in terms of features used by users. How can one find which feature is most used by the App's users?

Solution

We can use Google Analytics to track the App based on defined criteria, similar to the website tracking mechanism.

Discussion

Before we use Google Analytics in our app, we need an Analytics account and the Google Analytics SDK.

Download the Analytics SDK from <http://code.google.com/mobile/analytics/download.html>. Unzip the SDK and add libGoogleAnalytics.jar to your project's build path.

Add following permissions in your project's AndroidManifest.xml.

Example 2-37.

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

Now, sign into your analytics account and create a website profile for the App. The website URL can be fake but shall be descriptive enough. It is suggested that you use the reverse package name for this. For example, if the application package name is com.example.analytics.test; then the website URL for this app can be <http://test.analytics.example.com>. After successful creation of website profile, a web property ID is

generated for that profile. Note it down as we will be using this in our app. This web property ID, also known as UA number of your tracking code, uniquely identifies the website profile.

Note: You must mention in your App that you are collecting anonymous user data in your App to track your App.

After this setup, we are ready to track our application. Obtain the singleton instance of the tracker by calling `GoogleAnalyticsTracker.getInstance()` method. Then start tracking by calling its `start()` method. Usually, we will want to track more than activities in the App. In such a scenario it is a good idea to have this tracker instance in `OnCreate()` method of the Application class of the app.

Example 2-38.

```
public class TestApp extends Application {

    /*define your web property ID obtained after profile creation for the app*/
    private String webId = "UA-XXXXXXXX-Y";

    /*Analytics tracker instance*/
    GoogleAnalyticsTracker tracker;

    @Override
    public void onCreate() {
        super.onCreate();
        //get the singleton tracker instance
        tracker = GoogleAnalyticsTracker.getInstance();
        //start tracking app with your web property ID
        tracker.start(webId, getApplicationContext());
        //your app specific code goes here
    }

    /* This is getter for tracker instance. This is called in activity to get reference to tracker instance */
    public GoogleAnalyticsTracker getTracker() {
        return tracker;
    }
}
```

You can track pageviews and events in the activity by calling `trackPageView()` and `trackEvent()` methods on tracker instance.

Example 2-39.

```
public class MainActivity extends Activity
{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        //track the page view for activity
        GoogleAnalyticsTracker tracker = ((TestApp)getApplication()).getTracker();
        tracker.trackPageView("/MainActivity");
    }
}
```

```

        /*You can track events like button clicks*/
        findViewById(R.id.actionButton).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                GoogleAnalyticsTracker tracker = ((TestApp)getApplication()).getTracker();
                tracker.trackEvent("Action Event","Action Button", "Button clicked",0);
                tracker.dispatch();
            }
        });
        //your stuff goes here
    }
}

```

Remember, your events and pageviews will not be sent to the server until you call the `dispatch()` method on tracker. In this way we can track all the activities and events inside them.

2.17 Using AsyncTask to do background processing

Johan Pelgrim

Problem

You have to do some heavy processing, or load resources from the network and want to show the progress and results in the UI.

Solution

Use `AsyncTask` and `ProgressDialog` to achieve this

Discussion

Introduction

As we can read in the [Processes and Threads](#) section of the Android Dev Guide we know blocking the UI thread or accessing the Android UI toolkit from outside the UI thread are **not done**. Not... Ever... NEVER! Bad things will happen to you when you do... (you cannot repeat this often enough).

There are several methods to run processes in the background and updating the UI inside the UI thread (a.k.a. main thread) but using the `AsyncTask` class is very convenient and should be in every Android Developer's toolkit.

It boils down to creating a class which extends `AsyncTask`. `AsyncTask` itself is abstract and has one abstract method, `Result doInBackground(Params... params);`. The `AsyncTask` simply creates a callable working thread in which your implementation of `doInBackground` runs. `Result` and `Params` are two of the types we need to define in our class definition. The third is the `Progress` type which we will talk about later.

Let's revisit the [Recipe 9.13](#) recipe. The processing bit processes the content of a web page, which is an XML document, and returns the result as a list of `Datum` objects. Typically something we want to do outside of the UI thread.

Our first implementation will do everything in the background, showing the user a spinner in the title bar and updating the `ListView` once the processing is done. This is the typical use case, not interfering with the user's task at hand and updating the UI when you have retrieved the result.

The second implementation will use a *modal* dialog to show the processing progressing in the background. In some cases we want to prevent the user from doing anything else when some processing takes place and this is a good way to do just that.

We will create a UI which contains three `Buttons` and a `ListView`. The first button is to kick off our first refresh process. The second for the other refresh process and the third to clear the results in the `ListView`

Example 2-40.

```
<?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">
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <Button android:text="Refresh 1" android:id="@+id/button1"
            android:layout_width="fill_parent" android:layout_height="wrap_content" android:layout_weight="1"
        <Button android:text="Refresh 2" android:id="@+id/button2"
            android:layout_width="fill_parent" android:layout_height="wrap_content" android:layout_weight="1"
        <Button android:text="Clear" android:id="@+id/button3"
            android:layout_width="fill_parent" android:layout_height="wrap_content" android:layout_weight="1"
    </LinearLayout>
    <ListView android:id="@+id/listView1" android:layout_height="fill_parent"
        android:layout_width="fill_parent"></ListView>
</LinearLayout>
```

We assign these UI elements to various fields in `onCreate` and add some click listeners.

Example 2-41.

```
ListView mListView;
Button mClear;
Button mRefresh1;
Button mRefresh2;

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

    mListView = (ListView) findViewById(R.id.listView1);
    mListView.setTextFilterEnabled(true);
    mListView.setOnItemClickListener(this);
```



```

mRefresh1 = (Button) findViewById(R.id.button1);

mClear = (Button) findViewById(R.id.button3);
mClear.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        mListView.setAdapter(null);
    }
});

}

public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    Datum datum = (Datum) mListView.getItemAtPosition(position);
    Uri uri = Uri.parse("http://androidcookbook.com/Recipe.seam?recipeId=" + datum.getId());
    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    this.startActivity(intent);
}
}

```

Use Case 1: Processing in the background

First we create an inner class which extends `AsyncTask`

Example 2-42.

```

protected class LoadRecipesTask1 extends AsyncTask<String, Void, ArrayList<Datum>> {
}

```

As we can see we must supply three types to the class definition. The first is the type of the parameter which we will provide when starting this background task, in our case a `String`, containing a URL. The second type is used for progress updates (we will use this later). The third type is the type which is returned by our implementation of the `doInBackground` method, and typically something you can update a specific UI element with (a `ListView` in our case).

Let's implement the `doInBackground` method

Example 2-43.

```

@Override
protected ArrayList<Datum> doInBackground(String... urls) {
    ArrayList<Datum> datumList = new ArrayList<Datum>();
    try {
        datumList = parse(urls[0]);
    } catch (IOException e) {
        e.printStackTrace();
    } catch (XmlPullParserException e) {
        e.printStackTrace();
    }
    return datumList;
}
}

```

As you can see this is pretty simple. The parse method -- which creates a list of `Datum` objects -- is described in the [Recipe 9.13](#) recipe. The result of the `doInBackground` method is then passed as an argument to the `onPostExecute` method in the same (inner) class. In this method we are allowed to update the UI elements in our layout, so we set the adapter of the `ListView` to show our result.

Example 2-44.

```
@Override
protected void onPostExecute(ArrayList<Datum> result) {
    mListView.setAdapter(new ArrayAdapter<Datum>(MainActivity.this, R.layout.list_item, result));
}
```

Now we have to somehow start this task. We do this in the `mRefresh1`'s `onClick` listener by calling the `execute(Params... params)` method of `AsyncTask` (`execute(String... urls)` in our case).

Example 2-45.

```
mRefresh1.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        LoadRecipesTask1 mLoadRecipesTask = new LoadRecipesTask1();
        mLoadRecipesTask.execute("http://androidcookbook.com/seam/resource/rest/recipe/list");
    }
});
```

Now, when you start the app it indeed retrieves the recipes and fills the `ListView` but the user has no idea that something is happening in the background. We can set the progress bar indeterminate window feature in this case, which displays a small progress animation in the top-right of our app title bar.

To do this we request this feature by calling the following method in `onCreate`: `requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS)`;

Then we can start the progress animation by calling the `setProgressBarIndeterminateVisibility(Boolean visibility)` method from within a new method in our inner class, the `onPreExecute` method.

Example 2-46.

```
protected void onPreExecute() {
    MainActivity.this.setProgressBarIndeterminateVisibility(true);
}
```

We stop the spinning progress bar in our window title by calling the same method from within our `onPostExecute` method, which will become:

Example 2-47.

```
protected void onPostExecute(ArrayList<Datum> result) {
    mListView.setAdapter(new ArrayAdapter<Datum>(MainActivity.this, R.layout.list_item, result));
}
```

```

        MainActivity.this.setProgressIndicatorIndeterminateVisibility(false);
    }

```

We're done! Take your app for a *spin* (pun intended).

(BROKEN XREF TO RECIPE -1 'image:use-case-1.png')

A nifty feature for creating a better user experience!

Use Case 2: Processing in the foreground

In our second example we show a modal dialog to the user which displays the progress of loading the recipes in the background. Such a dialog is called a `ProgressDialog`. First we add it as a field to our activity.

Example 2-48.

```

    ProgressDialog mProgressDialog;

```

Then we add the `onCreateDialog` method to be able to answer `showDialog` calls and create our dialog.

Example 2-49.

```

    protected Dialog onCreateDialog(int id) {
        switch (id) {
            case DIALOG_KEY: // 1
                mProgressDialog = new ProgressDialog(this);
                mProgressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL); // 2
                mProgressDialog.setMessage("Retrieving recipes..."); // 3
                mProgressDialog.setCancelable(false); // 4
                return mProgressDialog;
        }
        return null;
    }

```

1. We should handle the request and creation of all dialogues here. The `DIALOG_KEY` is an `int` constant with an arbitrary value (we used 0) to identify this dialog.
2. We set the progress style to `STYLE_HORIZONTAL`, which shows a horizontal progress bar. The default is `STYLE_SPINNER`
3. We set our custom message, which is displayed above the progress bar
4. By calling `setCancelable` with argument `false` we simply disable the back-button, making this dialog *modal*

Then our new implementation of `AsyncTask`

Example 2-50.

```

    protected class LoadRecipesTask2 extends AsyncTask<String, Integer, ArrayList<Datum>> {

        @Override
        protected void onPreExecute() {

```

```

        mProgressDialog.show(); // 1
    }

    @Override
    protected ArrayList<Datum> doInBackground(String... urls) {
        ArrayList<Datum> datumList = new ArrayList<Datum>();
        for (int i = 0; i < urls.length; i++) { // 2
            try {
                datumList = parse(urls[i]);
                publishProgress((int) (((i+1) / (float) urls.length) * 100)); // 3
            } catch (IOException e) {
                e.printStackTrace();
            } catch (XmlPullParserException e) {
                e.printStackTrace();
            }
        }
        return datumList;
    }

    @Override
    protected void onProgressUpdate(Integer... values) { // 4
        mProgressDialog.setProgress(values[0]); // 5
    }

    @Override
    protected void onPostExecute(ArrayList<Datum> result) {
        mListview.setAdapter(new ArrayAdapter<Datum>(MainActivity.this, R.layout.list_item, result));
        mProgressDialog.dismiss(); // 6
    }
}

```

We see a couple of new things here.

1. Before we start our background process we show the modal dialog.
2. In our background process we loop through all the urls, expecting to receive more than one. This will give us a good indication on our progress.
3. We can update the progress by calling `publishProgress`. Notice that the argument is of type `int`, which will be autoboxed to the second type defined in our class definition, `Integer`.
4. The call to `publishProgress` will result in a call to `onProgressUpdate` which again has arguments of type `Integer`. You could of course use `String` or something else as the argument type by simply changing the second type in the inner class definition to `String` and of course in the call to `publishProgress`.
5. We use the first `Integer` to set the new progress value in our `ProgressDialog`.
6. Finally we dismiss the dialog, which removes it

Now we can all bind this together by implementing our `onClickListener` for our second refresh button.

Example 2-51.

```

mRefresh2.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {

```

```

        LoadRecipesTask2 mLoadRecipesTask = new LoadRecipesTask2();
        String url = "http://androidcookbook.com/seam/resource/rest/recipe/list";
        showDialog(DIALOG_KEY); // 1
        mLoadRecipesTask.execute(url, url, url, url, url); // 2
    }
});

```

1. We show the dialog by calling `showDialog` with the `DIALOG_KEY` argument, which will trigger our previously defined `onCreateDialog` method.
2. We execute our new task with 5 URLs, simply to show a little bit of progress.

It will look something like this.

(BROKEN XREF TO RECIPE -1 'image:use-case-2.png')

Conclusion

Implementing background tasks with `AsyncTask` is very simple and should be done for all long running processes which also need to update your user interface.

See Also

[Recipe 9.13](http://developer.android.com/guide/topics/fundamentals/processes-and-threads.html) <http://developer.android.com/guide/topics/fundamentals/processes-and-threads.html>

Source Download URL

The source code for this example may be downloaded from this URL: <https://github.com/downloads/jpelgrim/androidcookbook/RecipeList.zip>

2.18 A Simple Torch Light

Saketkumar Srivastav

Problem

How can we use our smart phone to behave as a torch lamp when there is no light or a power failure.

Solution

A simple solution will be to use the camera flash present in our smartphone or android phone to use it as a torch lamp.

Discussion

To begin with, let's see the design of the application.

- Access the Camera object of the phone.

- Access the parameters of our Camera
- Get the flashmodes supported by our camera
- Set the flashlight parameter in ON state to FLASH_MODE_TORCH and FLASH_OFF on OFF state.

The following code implements the logic required for the application.

Example 2-52.

```

if(context.getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA_FLASH)){
    mTorch = (ToggleButton) findViewById(R.id.toggleButton1);
    mTorch.setOnCheckedChangeListener(new OnCheckedChangeListener() {

        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

            try{
                if(cam != null){
                    cam = Camera.open();
                }
                camParams = cam.getParameters();
                List<String> flashModes = camParams.getSupportedFlashModes();
                if(isChecked){
                    if (flashModes.contains(Parameters.FLASH_MODE_TORCH)) {
                        camParams.setFlashMode(Parameters.FLASH_MODE_TORCH);
                    }else{
                        showDialog(MainActivity.this, FLASH_TORCH_NOT_SUPPORTED);
                    }
                }else{
                    camParams.setFlashMode(Parameters.FLASH_MODE_OFF);
                }
                cam.setParameters(camParams);
                cam.startPreview();
            }catch (Exception e) {
                e.printStackTrace();
                cam.stopPreview();
                cam.release();
            }
        }
    });
}
else{
    showDialog(MainActivity.this, FLASH_NOT_SUPPORTED);
}

```

The basic logic implemented below is as follows:

- Check for the existence of the flash in the device.
- Get the camera object and open it to access it.
- Get the parameters of the captured camera object.
- Check the supported flash modes available from the current camera object using getSupportedFlashModes().

- If the toggle state is ON the set the flash mode of the camera as FLASH_MODE_TORCH otherwise set it as FLASH_MODE_OFF .

Example 2-53.

```
public void showDialog (Context context, int dialogId){
    switch(dialogId){
        case FLASH_NOT_SUPPORTED:
            builder = new AlertDialog.Builder(context);
            builder.setMessage("Sorry, Your phone does not support Flash")
                .setCancelable(false)
                .setNegativeButton("Close", new OnClickListener() {

                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        finish();
                    }
                });
            alertDialog = builder.create();
            alertDialog.show();
            break;
        case FLASH_TORCH_NOT_SUPPORTED:
            builder = new AlertDialog.Builder(context);
            builder.setMessage("Sorry, Your camera flash does not support torch feature")
                .setCancelable(false)
                .setNegativeButton("Close", new OnClickListener() {

                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        finish();
                    }
                });
            alertDialog = builder.create();
            alertDialog.show();
        }
    }
}
```

Source Download URL

The source code for this example may be downloaded from this URL: <https://github.com/SaketSrivastav/SimpleTorchLight>

2.19 Adapting Android Phone Application to Tablet

Pratik Rupwal

Problem

I have developed an application for my smart phone. Is there a way to run it gracefully on a tablet without any significant changes to my code?

Solution

Following a few guidelines for porting a smartphone application to a tablet makes the application appear graceful on tablet.

Discussion

Assuming that you have android SDK installed on your computer, follow these instructions,

1. Launch the Android SDK and AVD Manager and install the following: a. SDK Platform Android 3.0 b. Android SDK Tools, revision 10 c. Android SDK Platform-tools, revision 3 d. Documentation for Android SDK, API 11 e. Samples for SDK API 11
2. Create an AVD for a tablet-type device, if you do not have an actual device yet. Set the target to "Android 3.0" and the skin to "WXGA" (the default skin).
3. Open your manifest file and update the *uses-sdk* element to set android:targetSdkVersion to "11". For example:

Example 2-54.

```
<manifest ... >
  <uses-sdk android:minSdkVersion="4"
            android:targetSdkVersion="11" />
  <application ... >
    ...
  <application>
</manifest>
```

By targeting the Android 3.0 platform, the system automatically applies the holographic theme to each activity when your application runs on an Android 3.0 device. The holographic theme provides a new design for widgets, such as buttons and text boxes, and new styles for other visual elements. This is the standard theme for applications built for Android 3.0, so your application will look and feel consistent with the system and other applications when it is enabled.

4. Build your application against the same version of the Android platform you have been using previously (such as the version declared in your android:minSdkVersion), but install it on the Android 3.0 AVD. (You should not build against Android 3.0 unless you are using new APIs.) Repeat your tests to be sure that your user interface works well with the holographic theme.

Optional guidelines:

1. Landscape layout: The "normal" orientation for tablet-type devices is usually landscape (wide), so you should be sure that your activities offer a layout that's optimized for a wide viewing area.
2. Button position and size: Consider whether the position and size of the most common buttons in your UI make them easily accessible while holding a tablet with two hands.

In some cases, you might need to resize buttons, especially if they use "wrap_content" as the width value. To enlarge the buttons, if necessary, you should either: add extra padding to the button; specify dimension values with dp units; or use android:layout_weight when the button is in a linear layout. Use your best judgment of proportions for each screen size-you don't want the buttons to be too big, either.

3. Font sizes: Be sure your application uses sp units when setting font sizes. This alone should ensure a readable experience on tablet-style devices, because it is a scale-independent pixel unit, which will resize as appropriate for the current screen configuration. In some cases, however, you still might want to consider larger font sizes for extra-large configurations.

2.20 First Run preferences

Ashwini Shahapurkar

Problem

I have an application which collects app usage data anonymously. I would like to make user aware of this on first run. How can I achieve this?

Solution

We can use the shared preferences as persistent storage to store the value which gets updated only once. Each time application launches it shall check for this value in preferences. If value is available, it is not the first run of application, else it is.

Discussion

We can manage application lifecycle by using the Application class of Android framework. We will use shared preferences as persistent storage to store the value of first run.

We will store a boolean flag if it is first run in preferences. When the application is installed and used for the first time, there are no preferences available for it. That will be created for us. In that case the flag shall return true value. After getting true flag, we can update this flag with false value as we no longer need it to be true.

Example 2-55.

```
public class MyApp extends Application {  
  
    SharedPreferences mPrefs;  
  
    @Override  
    public void onCreate() {  
        super.onCreate();  
  
        Context mContext = this.getApplicationContext();  
        //0 = mode private. only this app can read these preferences
```

```

        mPrefs = mContext.getSharedPreferences("myAppPrefs", 0);

        //here goes your app initialization code
    }

    public boolean getFirstRun() {
        return mPrefs.getBoolean("firstRun", true);
    }

    public void setRunned() {
        SharedPreferences.Editor edit = mPrefs.edit();
        edit.putBoolean("firstRun", false);
        edit.commit();
    }
}

```

This flag from preferences will be tested in launcher activity like this:

Example 2-56.

```

if(((MyApp) getApplication()).getFirstRun()){
    //This is first run
    ((MyApp) getApplication()).setRunned();

    // your code for first run goes here

}
else{
    // this is the case other than first run
}

```

The updates published to the app will not modify the preferences, so the code works for only first run after installation. Consequent updates to the app will not bring the code into picture, unless the user has manually uninstalled and installed the app.

Note: We can use a similar technique for distributing shareware versions of an Android app. I.e., we can limit the number of trials of the application using similar technique. In that case, we would use the integer count value in preferences for number of trials. Each trial would update the preferences. After the desired value is reached, we would block the usage of application.

2.21 Formatting the time and date display

Pratik Rupwal

Problem

I want to display the date and time in different standard formats. What is the the easiest approach to achieve this?

Solution

DateFormat class provides APIs for formatting time and date in custom format. Using these APIs requires minimal efforts to fulfil the requirement.

Discussion

Please refer the below code which adds 5 different TextViews for showing the time and date in different formats.

Example 2-57.

```
<?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:id="@+id/textview1"
    />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/textview2"
    />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/textview3"
    />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/textview4"
    />
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/textview5"
    />
</LinearLayout>
```

Below code obtains the current time and date using java.util.Date class and then displays it in different formats (Please refer the comments for sample outputs):

Example 2-58.

```
package com.sym.dateformatdemo;

import java.util.Calendar;
import android.app.Activity;
import android.os.Bundle;
import android.text.format.DateFormat;
import android.widget.TextView;

public class TestDateFormatterActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        TextView textView1 = (TextView) findViewById(R.id.textview1);
        TextView textView2 = (TextView) findViewById(R.id.textview2);
        TextView textView3 = (TextView) findViewById(R.id.textview3);
        TextView textView4 = (TextView) findViewById(R.id.textview4);
        TextView textView5 = (TextView) findViewById(R.id.textviews5);

        String delegate = "MM/dd/yy hh:mm a"; // 09/21/2011 02:17 pm
        java.util.Date noteTS = Calendar.getInstance().getTime();
        textView1.setText("Found Time :: "+DateFormat.format(delegate,noteTS));

        delegate = "MMM dd, yyyy h:mm aa"; // Sep 21,2011 02:17 pm
        textView2.setText("Found Time :: "+DateFormat.format(delegate,noteTS));

        delegate = "MMMM dd, yyyy h:mmaa"; //September 21,2011 02:17pm
        textView3.setText("Found Time :: "+DateFormat.format(delegate,noteTS));

        delegate = "E, MMMM dd, yyyy h:mm:ss aa";//Wed, September 21,2011 02:17:48 pm
        textView4.setText("Found Time :: "+DateFormat.format(delegate,noteTS));

        delegate = "EEEE, MMMM dd, yyyy h:mm aa"; //Wednesday, September 21,2011 02:17:48 pm
        textView5.setText("Found Time :: "+DateFormat.format(delegate,noteTS));
    }
}
```

See Also

Usage

DateUtils

This class contains various date-related utilities for creating text for things like elapsed time and date ranges, strings for days of the week and months, and AM/ text etc.

Formatter

Utility class to aid in formatting common values that are not covered by java.util.Formatter

Time

The Time class is a faster replacement for the java.util.Calendar and java.util.GregorianCalendar classes.

2.22 Controlling Input with KeyListeners

Pratik Rupwal

Problem

My application contains a few text boxes in which I want to restrict the users to enter only numbers; also in some cases to only positive numbers or even integers as well as dates. How do I achieve this with minimum efforts?

Solution

Android provides KeyListener classes to help you restrict the users to enter only numbers/positive numbers/integers/positive integers and much more.

Discussion

Android.text.method package includes an interface 'KeyListener' and some classes like DigitsKeyListener, DateKeyListener, etc. which implement this interface.

Below is a sample application which demonstrate few of them:

This layout file creates 5 textviews and 5 edittexts textviews display the input type allowed for their respective edittexts.

Example 2-59.

```
<?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:id="@+id/textview1"
        android:text="digits listener with signs and decimal points"
    />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/editText1"
    />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textview2"
        android:text="digits listener without signs and decimal points"
    />
    <EditText
```

```

        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/editText2"
    />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textview3"
        android:text="date listener"
    />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/editText3"
    />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textview4"
        android:text="multitap listener"
    />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/editText4"
    />

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textview5"
        android:text="qwerty listener"
    />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/editText5"
    />
</LinearLayout>

```

Below is the code for the activity which restricts the edittext input to numbers/positivengters/etc.(Refer the comments for group of keys allowed):

Example 2-60.

```

import android.app.Activity;
import android.os.Bundle;
import android.text.method.DateKeyListener;
import android.text.method.DigitsKeyListener;
import android.text.method.MultiTapKeyListener;
import android.text.method.QwertyKeyListener;
import android.text.method.TextKeyListener;
import android.widget.EditText;

```

```

public class KeyListenerDemo extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        EditText editText1=(EditText)findViewById(R.id.editText1);
        DigitsKeyListenerdigkl1=DigitsKeyListener.getInstance(true,true);//allows digits with positive/negative
        editText1.setKeyListener(digkl1);

        EditText editText2=(EditText)findViewById(R.id.editText2);
        DigitsKeyListener digkl2=DigitsKeyListener.getInstance();//allows positive inetger only(no decimal v
        editText2.setKeyListener(digkl2);

        EditText editText3=(EditText)findViewById(R.id.editText3);
        DateKeyListener dtkl=new DateKeyListener();//allows date only
        editText3.setKeyListener(dtkl);

        EditText editText4=(EditText)findViewById(R.id.editText4);
        MultiTapKeyListener multitapkl=new MultiTapKeyListener(TextKeyListener.Capitalize.WORDS,true);//allo
        editText4.setKeyListener(multitapkl);

        EditText editText5=(EditText)findViewById(R.id.editText5);
        QwertyKeyListener qkl=new QwertyKeyListener(TextKeyListener.Capitalize.SENTENCES,true);//allows qwer
        editText5.setKeyListener(qkl);
    }
}

```

For using 'MultiTapKeyListener' your phone should support the 12-key layout and it needs to be activated. To activate this 12-key layout, go to setting->Language and Keyboard -> On screen Keyboard layout -> select 'Phone layout' options.

the above options could be 'Qwerty layout'/'Phone layout'/'Compact qwerty layout'.

See Also

Usage

BaseKeyListener

Abstract base class for key listeners.

DateKeyListener

For entering dates and times in the same text field.

MetaKeyListener

This base class encapsulates the behavior for tracking the state of meta keys such as SHIFT,ALT and SYM as well as the pseudo-meta state of selecting text.

NumberKeyListener

For numeric text entry.

TextKeyListener

Usage

This is the key listener for typing normal text.

TimeKeyListener

For entering times in a text field.

2.23 Android Application Data Backup

Pratik Rupwal

Problem

If a user performs a factory reset or converts to a new Android-powered device, the application loses stored data or application settings.

Solution

Android's Backup Manager helps to automatically restore your backup data or application settings when the application is re-installed.

Discussion

Android's Backup mechanism basically operated in two modes, backup and restore. During a backup operation, Android's Backup Manager (BackupManager class) queries your application for backup data, then hands it to a backup transport, which then delivers the data to the cloud storage. During a restore operation, the Backup Manager retrieves the backup data from the backup transport and returns it to your application so your application can restore the data to the device. It's possible for your application to request a restore, which is not necessary as Android automatically performs a restore operation when your application is installed and there exists backup data associated with the user. The primary scenario in which backup data is restored when a user resets their device or upgrades to a new device and their previously installed applications are re-installed.

Below application describes how to implement BackupManager for your application so that you can save the current state of your application.

Basic description of procedure in step-by-step form:

- 1) Create a Project 'BackupManagerExample' in eclipse.
- 2) Open and insert the following code in layout/backup_restore.xml:

Example 2-61.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
```



```

<ScrollView
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:layout_weight="1">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <TextView android:text="@string/filling_text"
            android:textSize="20dp"
            android:layout_marginTop="20dp"
            android:layout_marginBottom="10dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>

        <RadioGroup android:id="@+id/filling_group"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="20dp"
            android:orientation="vertical">

            <RadioButton android:id="@+id/bacon"
                android:text="@string/bacon_label"/>
            <RadioButton android:id="@+id/pastrami"
                android:text="@string/pastrami_label"/>
            <RadioButton android:id="@+id/hummus"
                android:text="@string/hummus_label"/>

        </RadioGroup>

        <TextView android:text="@string/extras_text"
            android:textSize="20dp"
            android:layout_marginTop="20dp"
            android:layout_marginBottom="10dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>

        <CheckBox android:id="@+id/mayo"
            android:text="@string/mayo_text"
            android:layout_marginLeft="20dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>

        <CheckBox android:id="@+id/tomato"
            android:text="@string/tomato_text"
            android:layout_marginLeft="20dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>

    </LinearLayout>

```

```

    </ScrollView>
</LinearLayout>

```

3) Open values/string.xml and insert the following code in it,

Example 2-62.

```

<resources>
    <string name="hello">Hello World, BackupManager!</string>
    <string name="app_name">BackupManager</string>
    <string name="filling_text">Choose Settings for your application:</string>
    <string name="bacon_label">Sound On</string>
    <string name="pastrami_label">Vibration On</string>
    <string name="hummus_label">Backlight On</string>
    <string name="extras_text">Extras:</string>
    <string name="mayo_text">Use Orientation?</string>
    <string name="tomato_text">Use Camera?</string>
</resources>

```

4)Your manifest file will look like this:

Example 2-63.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.sym.backupmanager"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-sdk android:minSdkVersion="9" />

    <application android:label="Backup/Restore" android:icon="@drawable/icon"
        android:backupAgent="ExampleAgent"> <!-- Here you specify the backup agent-->

        <!--Some backup transports may require API keys or other metadata-->
        <meta-data android:name="com.google.android.backup.api_key"
            android:value="INSERT YOUR API KEY HERE" />

        <activity android:name=".BackupManagerExample">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity> </application>

</manifest>

```

5) The following code completes the implementation of the BackupManager for your application.

Example 2-64.

```

package com.sym.backupmanager;

import android.app.Activity;

```

```

import android.app.backup.BackupManager;
import android.os.Bundle;
import android.util.Log;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.RadioGroup;
import java.io.File;
import java.io.IOException;
import java.io.RandomAccessFile;

public class BackupManagerExample extends Activity {
    static final String TAG = "BRActivity";

    static final Object[] sDataLock = new Object[0];

    static final String DATA_FILE_NAME = "saved_data";

    RadioGroup mFillingGroup;
    CheckBox mAddMayoCheckbox;
    CheckBox mAddTomatoCheckbox;

    File mDataFile;

    BackupManager mBackupManager;

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

        setContentView(R.layout.backup_restore);

        mFillingGroup = (RadioGroup) findViewById(R.id.filling_group);
        mAddMayoCheckbox = (CheckBox) findViewById(R.id.mayo);
        mAddTomatoCheckbox = (CheckBox) findViewById(R.id.tomato);

        mDataFile = new File(getFilesDir(), BackupManagerExample.DATA_FILE_NAME);

        mBackupManager = new BackupManager(this);

        populateUI();
    }

    void populateUI() {
        RandomAccessFile file;

        int whichFilling = R.id.pastrami;
        boolean addMayo = false;
        boolean addTomato = false;

        synchronized (BackupManagerExample.sDataLock) {
            boolean exists = mDataFile.exists();
            try {
                file = new RandomAccessFile(mDataFile, "rw");
                if (exists) {
                    Log.v(TAG, "datafile exists");
                }
            }
        }
    }
}

```

```

        whichFilling = file.readInt();
        addMayo = file.readBoolean();
        addTomato = file.readBoolean();
        Log.v(TAG, " mayo=" + addMayo
            + " tomato=" + addTomato
            + " filling=" + whichFilling);
    } else {
        Log.v(TAG, "creating default datafile");
        writeToFileLocked(file,
            addMayo, addTomato, whichFilling);

        mBackupManager.dataChanged();
    }
} catch (IOException ioe) {
}
}

mFillingGroup.check(whichFilling);
mAddMayoCheckbox.setChecked(addMayo);
mAddTomatoCheckbox.setChecked(addTomato);

mFillingGroup.setOnCheckedChangeListener(
    new RadioGroup.OnCheckedChangeListener() {
        public void onCheckedChanged(RadioGroup group,
            int checkedId) {
            Log.v(TAG, "New radio item selected: " + checkedId);
            recordNewUIState();
        }
    });

CompoundButton.OnCheckedChangeListener checkListener
    = new CompoundButton.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView,
            boolean isChecked) {
            Log.v(TAG, "Checkbox toggled: " + buttonView);
            recordNewUIState();
        }
    };
mAddMayoCheckbox.setOnCheckedChangeListener(checkListener);
mAddTomatoCheckbox.setOnCheckedChangeListener(checkListener);
}

void writeToFileLocked(RandomAccessFile file,
    boolean addMayo, boolean addTomato, int whichFilling)
    throws IOException {
    file.setLength(0L);
    file.writeInt(whichFilling);
    file.writeBoolean(addMayo);
    file.writeBoolean(addTomato);
    Log.v(TAG, "NEW STATE: mayo=" + addMayo
        + " tomato=" + addTomato
        + " filling=" + whichFilling);
}

```

```

void recordNewUIState() {
    boolean addMayo = mAddMayoCheckbox.isChecked();
    boolean addTomato = mAddTomatoCheckbox.isChecked();
    int whichFilling = mFillingGroup.getCheckedRadioButtonId();
    try {
        synchronized (BackupManagerExample.sDataLock) {
            RandomAccessFile file = new RandomAccessFile(mDataFile, "rw");
            writeDataToFileLocked(file, addMayo, addTomato, whichFilling);
        }
    } catch (IOException e) {
        Log.e(TAG, "Unable to record new UI state");
    }

    mBackupManager.dataChanged();
}
}

```

Data backup is not guaranteed to be available on all Android-powered devices. However, your application is not adversely affected in the event that a device does not provide a backup transport. If you believe that users will benefit from data backup in your application, then you can implement it as described in this document, test it, then publish your application without any concern about which devices actually perform backup. When your application runs on a device that does not provide a backup transport, your application operates normally, but will not receive callbacks from the Backup Manager to backup data.

Although you cannot know what the current transport is, you are always assured that your backup data cannot be read by other applications on the device. Only the Backup Manager and backup transport have access to the data you provide during a backup operation.

Caution: Because the cloud storage and transport service can differ from device to device, Android makes no guarantees about the security of your data while using backup. You should always be cautious about using backup to store sensitive data, such as usernames and passwords.

See Also

Testing Your Backup Agent:

Once you've implemented your backup agent, you can test the backup and restore functionality with the following procedure, using `bmgr`.

1. Install your application on a suitable Android system image. If using the emulator, create and use an AVD with Android 2.2 (API Level 8). If using a device, the device must be running Android 2.2 or greater and have Android Market built in.
2. Ensure that backup is enabled. If using the emulator, you can enable backup with the following command from your SDK tools/ path:

```
adb shell bmgr enable true
```

If using a device, open the system Settings, select Privacy, then enable Back up my data and Automatic restore. 3. Open your application and initialize some data

If you've properly implemented backup in your application, then it should request a backup each time the data changes. For example, each time the user changes some data, your app should call `dataChanged()`, which adds a backup request to the Backup Manager queue. For testing purposes, you can also make a request with the following `bmgr` command:

```
adb shell bmgr backup your.package.name
```

4. Initiate a backup operation:

```
adb shell bmgr run
```

This forces the Backup Manager to perform all backup requests that are in its queue.

5. Uninstall your application:

```
adb uninstall your.package.name
```

6. Re-install your application.

If your backup agent is successful, all the data you initialized in step 4 is restored.

2.24 Making password fields

Rachee Singh

Problem

You need to designate an `EditText` as a password field

Solution

Android provides the `password` attribute on `EditText`s

Discussion

If your application requires the user to enter a password, the `EditText` being used should be `password`. It should hide the characters entered. This can be done by adding this property to the `EditText` in XML:

Example 2-65.

```
android:password="True"
```

After doing this, this is how the `EditText` in which password is being entered looks:



Figure 2-8.

2.25 Working Without Tooltips: Use Hints Instead

Daniel Fowler

Problem

Android devices can have small screens, there may not be room for help text, and tooltips are not part of the platform.

Solution

Android provides the *hint* attribute for Views.

Discussion

Sometimes an input field needs clarification with regards to the value being entered. For example a stock ordering application asking for item quantities may need to state the minimum order size. In desktop programs, with large screens and the use of a mouse, extra messages can be displayed in the form of tooltips (a popup label over a field when the mouse moves over it). Alternatively, long descriptive labels may be used. With Android devices the screen may be small and no mouse is generally used. The alternative here is to use the `android:hint` attribute on a `View`. This causes a "watermark" containing the hint text to be displayed in the input field when it is empty; this disappears when the user starts typing in the field. The corresponding function for `android:hint` is `setHint(int resourceId)`.

The color of the hint text can be set with `android:textColorHint` with `setHintTextColor(int color)` being the associated function.

Using these 'hints' can also help with screen layouts when space is tight. It can allow labels to be removed to gain more space as the hints provide the necessary prompt for the user. A screen design can sometimes be improved by removing a label and using a hint.

The `EditText` definition in the above screen is shown here to see `android:hint` in use.

Example 2-66.

```
<EditText android:id="@+id/etQuantity"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="Number of boxes of ten"
    android:textSize="18sp"/>
```

Hints can guide users filling in App fields, though like any feature over use is possible. Hints must not be used when it is obvious what is required, a field with a label of *First Name* would not need a hint such as *Enter your first name here*.

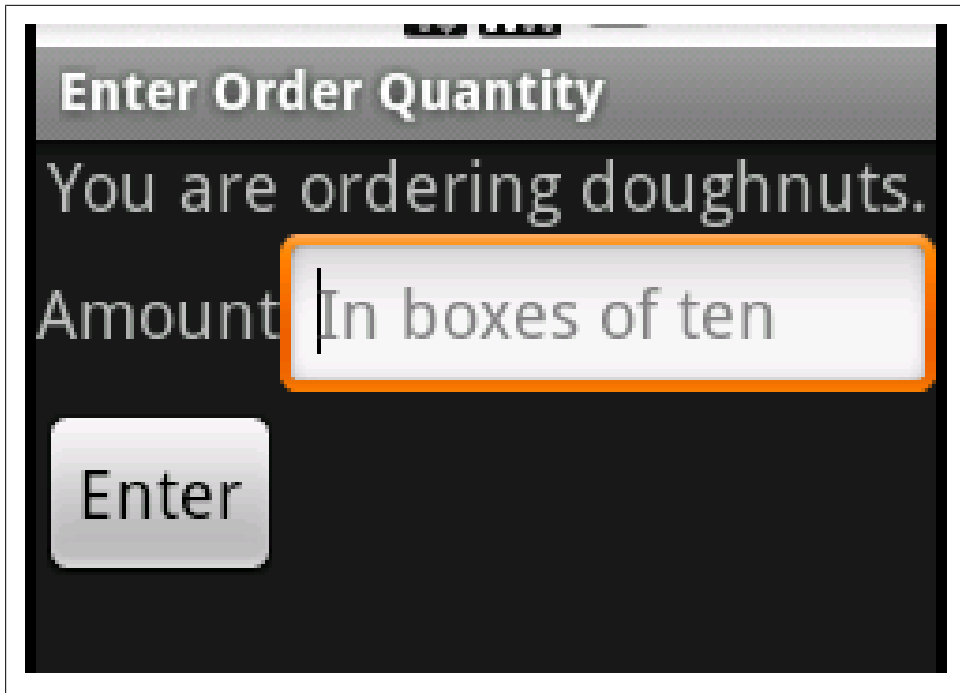


Figure 2-9.

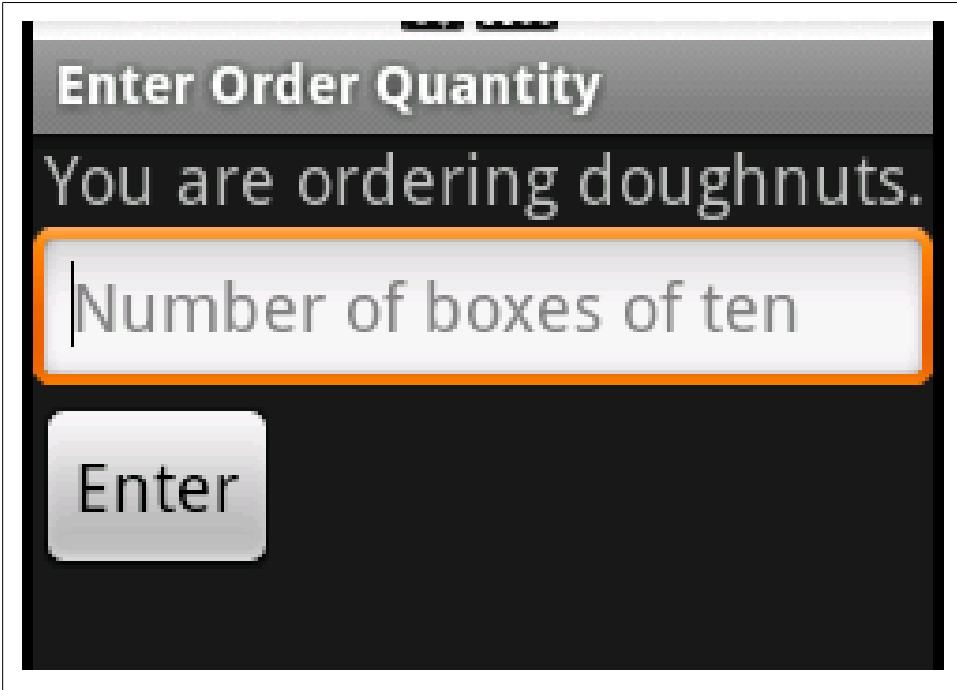


Figure 2-10.

3.1 Introduction: Testing

Ian Darwin

Discussion

Test early and often is a common cry among advocates of testing. As is the all-important question "If you don't have a test, how do you know your code works?"

There are many types of testing. Unit Testing checks out individual components in isolation (not hitting the network or the database). JUnit and TestNG are the leading frameworks here. Mock Objects are used where interaction with other components is required; there are several good mocking frameworks for Java.

XXX Other Types ...

Android provides an Application Testing facility. We will have a recipe on it soon.

3.2 How to TDD(test driven development) Android App

kailuo wang

Problem

Lack of mocking support making TDD Android App cumbersome.

Solution

Setup 2 test projects: one created using the Android tool for the UI related tests, and another standard Unit Test projects for mock supported tests. Extract as much as your logic to the classes that can be unit tested.

Discussion

In the official guide, the test related articles are mostly about testing the UI tests. An Android test project needs to be created so that it can be instrumented and deployed and test the app on a simulator environment. It's very cool and necessary for testing the UI related logic, but it also made mocking very difficult. There is some work around but they make things a bit ad hoc and potentially painful. If you step back and look at them from a higher level, these tests are more like integration tests than pure unit tests. They take longer to run, they require the whole environment up. Without mocking, they might need to test a lot more than a unit of functionality. All these justify the decision to make such tests a separate project/module from the normal unit test project/module. We can call this android tool created project/module the XYZ UI Test project, whose responsibility is to test only UI logic. Now you can setup another standard Unit test project as you always do. Let's call it the XYZ Unit Test project. Here you can use your favorite tools including mock frameworks. Also it's testing only all the non-UI related logic which avoids all the not-that-test-friendly Android UI API. Now all you need to do is to extract as much logic out of the nasty UI dependent classes and have fun TDD :)

See Also

http://developer.android.com/resources/tutorials/testing/helloandroid_test.html

3.3 How to troubleshoot "The application has stopped unexpectedly. Please try again"

Ulysses Levy

Problem

Your app crashes and you are uncertain as to why.

Solution

Begin by viewing the log.

Discussion

app crash

This is what an app crash looks like.

logcat

We can use `adb logcat` to view our AVD's log.

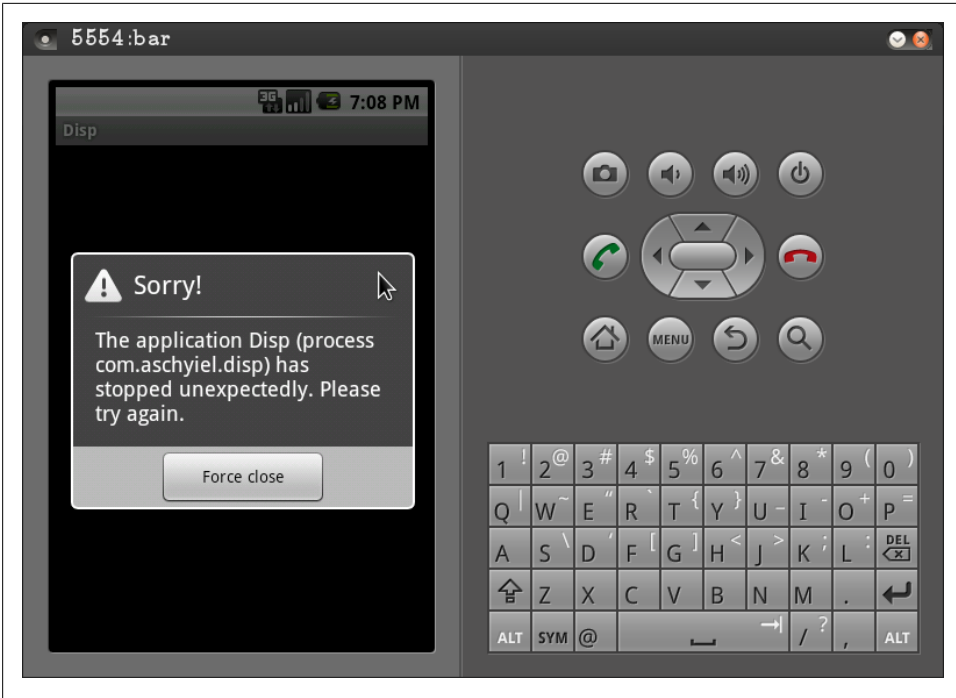


Figure 3-1.

Example 3-1.

```
adb remount
adb logcat
```

Example 3-2.

...

```
E/DatabaseUtils( 53): Writing exception to parcel
E/DatabaseUtils( 53): java.lang.SecurityException: Permission Denial: writing com.android.providers.settings
E/DatabaseUtils( 53):     at android.content.ContentProvider$Transport.enforceWritePermission(ContentProvi
E/DatabaseUtils( 53):     at android.content.ContentProvider$Transport.insert(ContentProvider.java:149)
E/DatabaseUtils( 53):     at android.content.ContentProviderNative.onTransact(ContentProviderNative.java:1
E/DatabaseUtils( 53):     at android.os.Binder.execTransact(Binder.java:287)
E/DatabaseUtils( 53):     at com.android.server.SystemServer.init1(Native Method)
E/DatabaseUtils( 53):     at com.android.server.SystemServer.main(SystemServer.java:497)
E/DatabaseUtils( 53):     at java.lang.reflect.Method.invokeNative(Native Method)
E/DatabaseUtils( 53):     at java.lang.reflect.Method.invoke(Method.java:521)
E/DatabaseUtils( 53):     at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:86
E/DatabaseUtils( 53):     at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:618)
E/DatabaseUtils( 53):     at dalvik.system.NativeStart.main(Native Method)
D/AndroidRuntime( 430): Shutting down VM
W/dalvikvm( 430): threadid=3: thread exiting with uncaught exception (group=0x4001b188)
...
```

Example Solution

In this scenario, we have a permission issue. So the solution to this particular instance is to add the `WRITE_SETTINGS` permission to our `AndroidManifest.xml` file.

Example 3-3.

```
<manifest ... >
  <application ... />
  <uses-permission android:name="android.permission.WRITE_SETTINGS" />
</manifest>
```

Example 2 : Null Pointer Exception

The Null Pointer Exception (NPE) is fairly common.

logcat. Here's the logcat output:

Example 3-4.

```
I/ActivityManager( 53): Displayed activity com.android.launcher/.Launcher: 28640 ms (total 28640 ms)
I/ActivityManager( 53): Starting activity: Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] }
I/ActivityManager( 53): Start proc com.aschyiell.disp for activity com.aschyiell.disp/.Disp: pid=214 uid=1000
I/ARMAssembler( 53): generated scanline_00000177:03515104_00000001_00000000 [ 73 ipp] (95 ins) at [0x47c50000-0x47c50004]
I/ARMAssembler( 53): generated scanline_00000077:03545404_00000004_00000000 [ 47 ipp] (67 ins) at [0x47c70000-0x47c70004]
I/ARMAssembler( 53): generated scanline_00000077:03010104_00000004_00000000 [ 22 ipp] (41 ins) at [0x47c80000-0x47c80004]
D/AndroidRuntime( 214): Shutting down VM
W/dalvikvm( 214): threadid=3: thread exiting with uncaught exception (group=0x4001b188)
E/AndroidRuntime( 214): Uncaught handler: thread main exiting due to uncaught exception
E/AndroidRuntime( 214): java.lang.RuntimeException: Unable to start activity ComponentInfo{com.aschyiell.disp/com.aschyiell.disp.Disp}: java.lang.NullPointerException
E/AndroidRuntime( 214):     at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2496)
E/AndroidRuntime( 214):     at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2512)
E/AndroidRuntime( 214):     at android.app.ActivityThread.access$2200(ActivityThread.java:119)
E/AndroidRuntime( 214):     at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1863)
E/AndroidRuntime( 214):     at android.os.Handler.dispatchMessage(Handler.java:99)
E/AndroidRuntime( 214):     at android.os.Looper.loop(Looper.java:123)
E/AndroidRuntime( 214):     at android.app.ActivityThread.main(ActivityThread.java:4363)
E/AndroidRuntime( 214):     at java.lang.reflect.Method.invokeNative(Native Method)
E/AndroidRuntime( 214):     at java.lang.reflect.Method.invoke(Method.java:521)
E/AndroidRuntime( 214):     at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:843)
E/AndroidRuntime( 214):     at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:618)
E/AndroidRuntime( 214):     at dalvik.system.NativeStart.main(Native Method)
E/AndroidRuntime( 214): Caused by: java.lang.NullPointerException
E/AndroidRuntime( 214):     at com.aschyiell.disp.Disp.onCreate(Disp.java:66)
E/AndroidRuntime( 214):     at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1047)
E/AndroidRuntime( 214):     at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2459)
E/AndroidRuntime( 214):     ... 11 more
```

Example Code (with error). And the code...

Example 3-5.

```
import ...

public class Disp extends Activity
{
```

```

private TextView foo;
@Override
public void onCreate( Bundle savedInstanceState )
{
    ...

    foo.setText("bar");
}
}

```

The above code fails because we forgot to use `findViewById()`.

example code (with fix). Once more, with the fix:

Example 3-6.

import ...

```

public class Disp extends Activity
{
    private TextView foo;
    @Override
    public void onCreate( Bundle savedInstanceState )
    {
        ...

        foo = (TextView) findViewById( R.id.id_foo );
        foo.setText("bar");
    }
}

```

The above code should make our error go away.

See Also

Google I/O 2009 - Debugging Arts of the Ninja Masters http://groups.google.com/group/android-developers/browse_thread/thread/92ea776cfd42aa45

3.4 Debugging using Log.d and LogCat

Rachee Singh

Problem

Many a times even though the Java code compiles without error, running the application crashes giving a 'Force Close' (or similar) error message.

Solution

Debugging the code using Log Cat messages is a useful technique that a developer must be equipped with in such scenarios.

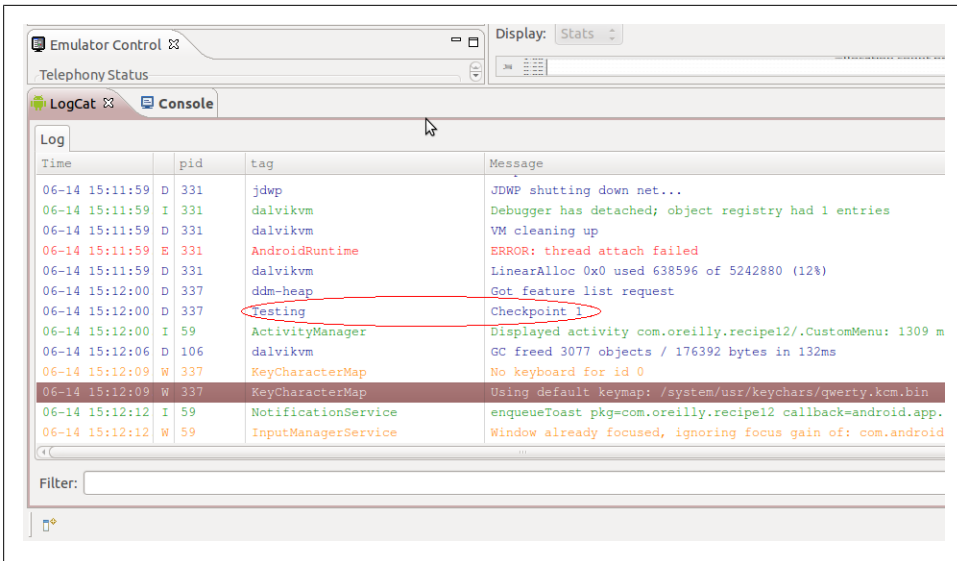


Figure 3-2.

Discussion

Those familiar with Java programming have probably used `System.out.println` statements while debugging their code. Similarly, debugging an Android application can be facilitated by the use of using the `Log.d` method. This helps print necessary values and messages in the Log Cat window. Start with the import of the Log class:

Example 3-7.

```
import android.util.Log;
```

Then, insert this line in the code at places where you wish to check the status of the application:

Example 3-8.

```
Log.d("Testing", "Checkpoint 1");
```

'Testing' is the Tag (which appears under the Tag column of Log Cat. 'Checkpoint 1' is the message (which appears in the Message Column of Log Cat). `Log.d` takes these 2 arguments. Corresponding to these, an appropriate message is displayed in the Log Cat. So if you have inserted this as a check point and you get the message displayed in Log Cat implies that the code works fine till that point.

The `Log.d` method does not accept variable arguments, so if you wish to format more than one item, use String concatenation or `String.format` (but omit the trailing `%n`):

Example 3-9.

```
Log.d("Testing", String.format("x0 = %5.2f, x1=%5.2f", x0, x1));
```

3.5 Keep Your App Snappy With StrictMode

Adrian Cowham

Problem

Making sure your app's GUI is as snappy as possible isn't easy. But with the help of Android's StrictMode it doesn't seem all that bad.

Solution

Android has a tool called Strict Mode that they introduced in Gingerbread, Strict Mode will detect all cases where an ANR might occur. For example, it will detect and log to LogCat all database reads and writes that happen on the main thread (i.e. the GUI thread).

Discussion

I wish I could've used a tool like StrictMode back when I was doing Java Swing Desktop development. Making sure our Java Swing App was snappy was a constant challenge, green and seasoned engineers would invariably perform database operations on the UI thread that would cause the app to hiccup. Typically we found these hiccups when QA (or customers) would use the app with a larger dataset than the engineers were testing with. Having QA find these little defects was unacceptable and ultimately a waste of everyone's time (and the company's money). We eventually solved the problem by investing more heavily in peer reviews, but having a tool like StrictMode would have been comparatively cheaper.

The example code below illustrates how easily StrictMode can be turned on in your app.

Example 3-10.

```
// make sure you import StrictMode
import android.os.StrictMode;

// In you app's android.app.Applicatoin instance, add the following
// lines to the onCreate(...) method.
if ( Build.VERSION.SDK_INT >= 9 && isDebug() ) {
    StrictMode.enableDefaults();
}
```

Please note that the `isDebug()` implementation has been intentionally omitted, as this will vary among developers. I recommend only enabling StrictMode when your app is in Debug mode, it's unwise to put your app in the Market with StrictMode running in the background and consuming resources unnecessarily.

StrictMode is highly configurable, it allows you to customize what problems to look for. For detailed information on customizing StrictMode policies, see the StrictMode link below.

See Also

<http://developer.android.com/reference/android/os/StrictMode.html>

3.6 Barrel of Monkeys

Adrian Cowham

Problem

Not quite the Turing test, but if your app can pass the monkey test, you deserve a pat on the back.

Solution

Testing is so easy a monkey can do it, literally. Despite the lack of testing tools for Android, I have to admit that monkey is pretty cool. If you're not familiar with Android's monkey, it's a testing tool that comes with the Android SDK and simulates a monkey (or perhaps a child) using an Android device. Imagine a monkey sitting at keyboard and flailing away, get the idea? What better way to flush out those hidden ANRs?

Discussion

Running monkey is as simple as starting the emulator (or connecting your development device to your development machine) and launching the monkey script. I hate to admit this, but by running monkey on a daily basis we've repeatedly found defects that probably would've escaped a normal QA pass and would've been very challenging to troubleshoot if found in the field. Or worse yet, caused users to stop using our app.

Here we talk about a few best practices for using monkey in your development process.

First, use monkey.

Second, create your own monkey script that wraps Android's monkey script. This is to ensure that all the developers on your team are running monkey with the same parameters. If you're a team of one, this helps with predictability (discussed below).

Next, you'll want to configure monkey so that it runs long enough to catch defects and not so long that it's a productivity killer. In our development process, we configured monkey to run for a total of 50,000 events. This took about 40 minutes to run on a Samsung Galaxy Tab. Not too bad, but I would've liked it to be in the 30 minute range. Obviously, faster tablets will have a higher throughput.

Monkey is random, so when we first started running monkey every developer was getting different results and we were unable to reproduce defects. We then figured out that monkey allows you to set the seed for its random number generator. So, configure your wrapper script to set monkey's seed. This will ensure uniformity and predictability across monkey runs in your development team.

Lastly, once you gain confidence in your app with a specific seed value, change it, you'll never know what monkey will find.

Below is a monkey script wrapper, and a description of its arguments.

- `-p package name` will ensure that monkey only targets the package specified
- `--throttle` is the delay between events
- `-s` is the seed value
- `-v` is the verbose option
- 50000 is the number of event monkey will simulate.

Example 3-11.

```
#!/bin/bash
# Utility script to run monkey
#
# See: http://developer.android.com/guide/developing/tools/monkey.html

rm tmp/monkey.log
adb shell monkey -p package.name.here --throttle 100 -s 43686 -v 50000 | tee tmp/monkey.log
```

There are many more configuration options for monkey, we deliberately chose not to mess around with what types of events monkey generates because we appreciate the pain. For example, the seed value we chose causes monkey to disable Wifi about half-way through the run. This was really frustrating at first because we felt like we weren't getting the coverage we wanted. Turns out, monkey did us a favor by disabling Wifi and then relentlessly playing with our app. After discovering and fixing a few defects, we soon had complete confidence that our app operated as expected with no network connection.

Good monkey.

See Also

<http://developer.android.com/guide/developing/tools/monkey.html>

3.7 Sending text messages and placing calls between AVDs

Johan Pelgrim

Problem

You have made an app which needs to place or listen for calls or send or receive text messages and you want to test this.

Solution

Fire up two AVDs and use the process id to send text messages and place calls

Discussion

When you create an app that listens for incoming calls or text messages -- like the one in recipe [Recipe 10.2](#) -- you can of course use the DDMS perspective in Eclipse to simulate placing calls or sending text messages, but you can also fire up another AVD!

If you look at the AVD window title you see a number before your AVD's title. This is the port number which you can use to telnet to your AVD's shell (e.g. `telnet localhost 5554`). Fortunately for testing purposes this number is your AVDs *phone number* as well. So you can use this number to place calls

(BROKEN XREF TO RECIPE -1 'image:call-other-avd.png')

... or send text.

(BROKEN XREF TO RECIPE -1 'image:send-text-to-other-avd.png')

See Also

[Recipe 10.2](#)

3.8 Activity LifeCycle Scenarios for Testing

Daniel Fowler

Problem

Apps should be resilient to the Activity Lifecycle. Developers need to know how to reproduce different lifecycle scenarios.

Solution

Use logging to get a good understanding of the Activity Lifecycle. Lifecycle scenarios are then easier to reproduce for App testing.

Discussion

Android is designed for life on the go, where a user is engaged in multiple tasks, taking calls, checking email, sending SMS, social networking, taking pictures, accessing the Internet, running Apps and more, maybe including some work! As such a device can

have multiple Apps and hence many Activities loaded in memory. The foreground App and its current Activity can be interrupted and paused at any moment. Apps, and hence Activities, that are paused can be removed from memory to free up space for newly started Apps. An App has a lifecycle which it cannot control as it is the Android operating system that starts, monitors, pauses, resumes and destroys the Apps Activities. Yet an Activity does know what is going on, as Activities are instantiated, hidden and destroyed various functions are called. This allows the Activity to keep track on what the operating system is doing to the App, see the recipe [Recipe 1.7](#).

App developers become familiar with the functions invoked when an Activity starts:

- `onCreate(Bundle savedInstanceState){...};`
- `onStart(){...};`
- `onResume(){...};`

And the functions called when an Activity is paused and then removed from memory (destroyed).

- `onPause(){...};`
- `onStop(){...};`
- `onDestroy(){..};`

It is easy to see them in action open the simple program from the [Recipe 1.4](#) recipe. Then in the `Main.java` class override the above functions, calling through to the super class versions. Add a call to `Log.d()` to pass in the name of the App and the function being invoked. The code will look like this:

Example 3-12.

```
public class Main extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Log.d("MyAndroid", "onCreate");
    }
    @Override
    public void onStart() {
        super.onStart();
        Log.d("MyAndroid", "onStart");
    }
    @Override
    public void onResume() {
        super.onResume();
        Log.d("MyAndroid", "onResume");
    }
    @Override
    public void onPause() {
        super.onPause();
        Log.d("MyAndroid", "onPause");
    }
}
```

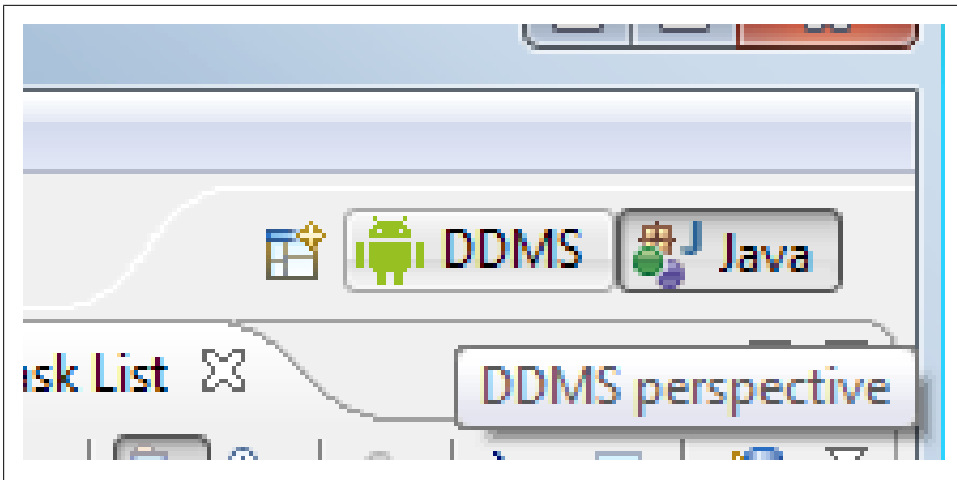


Figure 3-3.

```
public void onStop() {
    super.onStop();
    Log.d("MyAndroid", "onStop");
}
public void onDestroy() {
    super.onDestroy();
    Log.d("MyAndroid", "onDestroy");
}
}
```

Run the program. To see the debug messages the LogCat View needs to be displayed. This is visible by default in the Dalvik Debug Monitor Server (DDMS) perspective, or can be opened via the **Window** menu option. Click **Window->Show View->Other**, expand **Android** and select **LogCat**. The LogCat view appears on the bottom tabs.

To open the DDMS perspective click the DDMS button in the top right corner of Eclipse:

The LogCat view will be on the bottom tabs. If not visible use the **Window** method above or select **Window->Reset Perspective**. LogCat can be dragged off into its own window by dragging the tab from Eclipse. After starting the program the three debug messages added to the start up functions can be seen.

When the back key is pressed the three teardown messages are seen.

To see only the messages from the App add a LogCat filter. Click on the green plus in the top right of the LogCat screen. Give the filter a name and type "MyAndroid" in the **Log Tagtag** field.

LogCat will now show a new tab with only the messages explicitly sent from the App.

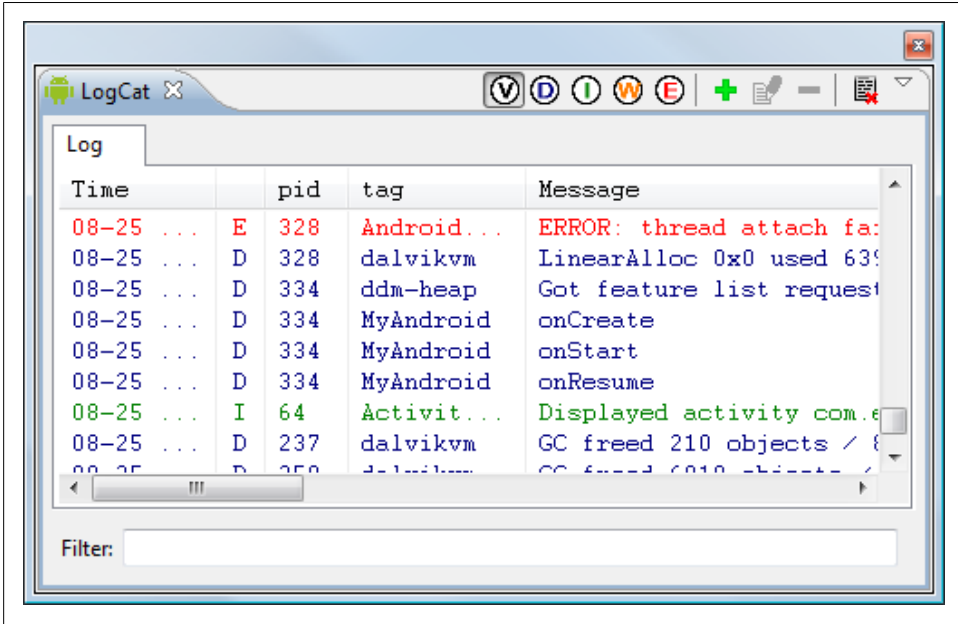


Figure 3-4.

The LogCat output can be cleared by clicking the top right icon that shows a page with a small red x. Useful to have a clean sheet before performing an action to watch for more messages.

To see the functions called when a program is paused open up an application over the MyAndroid program. First add the function for `onRestart()`, and the debug message.

Example 3-13.

```

@Override
public void onRestart() {
    super.onRestart();
    Log.d("MyAndroid", "onRestart");
}

```

Run the program, click the Home button, then launch the program again from the device (or emulator).

LogCat shows the usual start up function sequence, then when the Home button is pressed `onPause()` and `onStop()` run, but no `onDestroy()`. The program is not ending but effectively sleeping. When the program is run again it is not reloaded so no `onCreate()` executes, instead `onRestart()` is called.

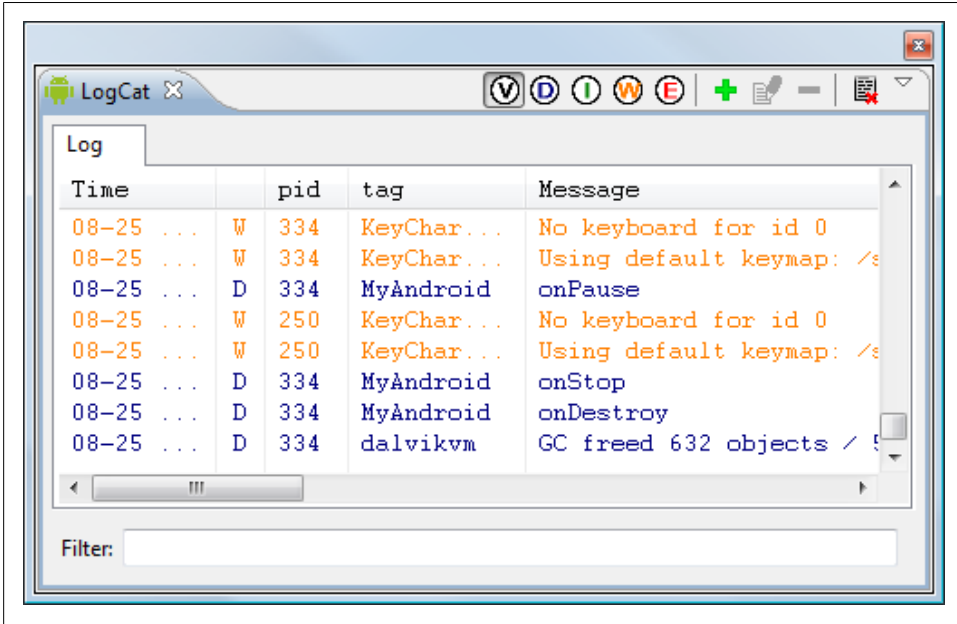


Figure 3-5.

Run the program again, on the device or emulator, go into **Manage Applications** (via **Settings** then **Applications**), select the program and press the **Force Close** button. Then start the program again from the device (or emulator).

The usual start up functions are invoked then the Activity "sleeps". No `onDestroy()` is seen as the second instance is run.

The above has shown different lifecycle scenarios.

- Normal start up and then finish.
- Start up, pause and then restart.
- Start up, pause, forced removal from memory and then start up again.

These scenarios result in different sequences of lifecycle functions being executed. Using these scenarios when testing ensures an App performs correctly for a user. The techniques shown here can be extended when implementing additional overridden functions. The techniques also apply to using **Fragments** in an Activity and testing their lifecycle.

See Also

[Recipe 1.7](#)

[Recipe 1.4](#)

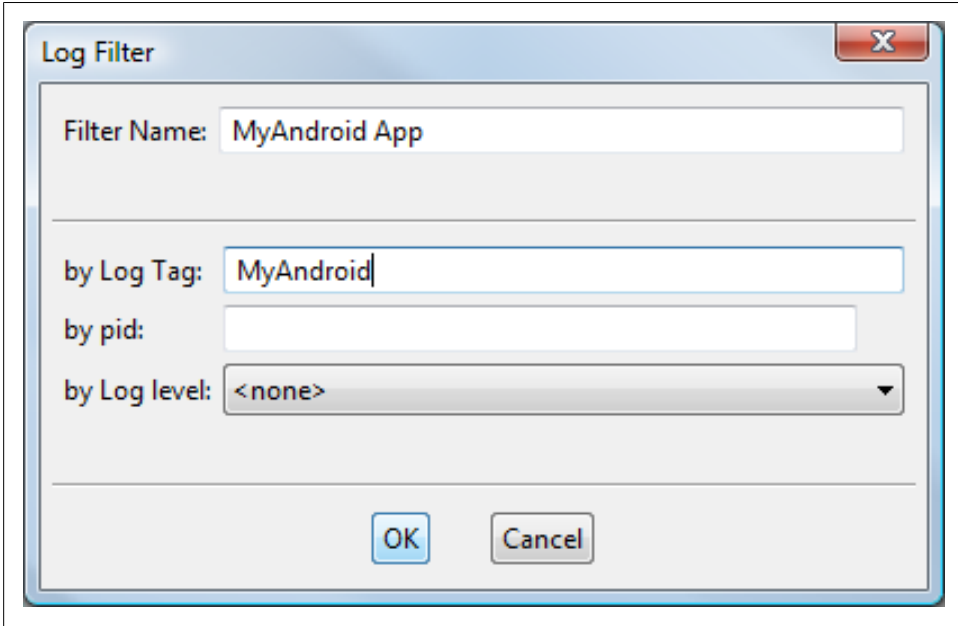


Figure 3-6.

<http://developer.android.com/reference/android/app/Activity.html>

<http://developer.android.com/reference/android/util/Log.html>

<http://developer.android.com/guide/topics/fundamentals/fragments.html>

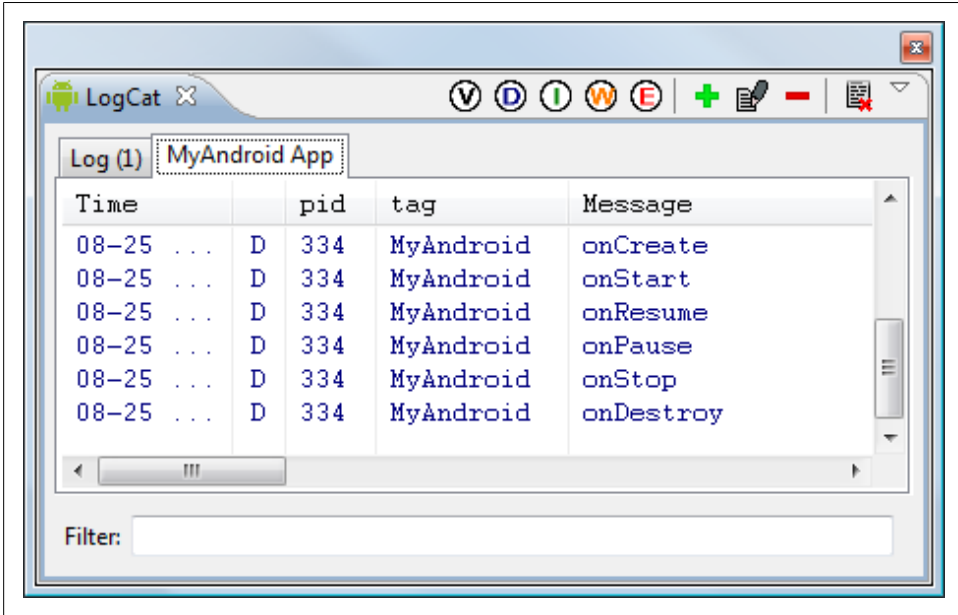


Figure 3-7.

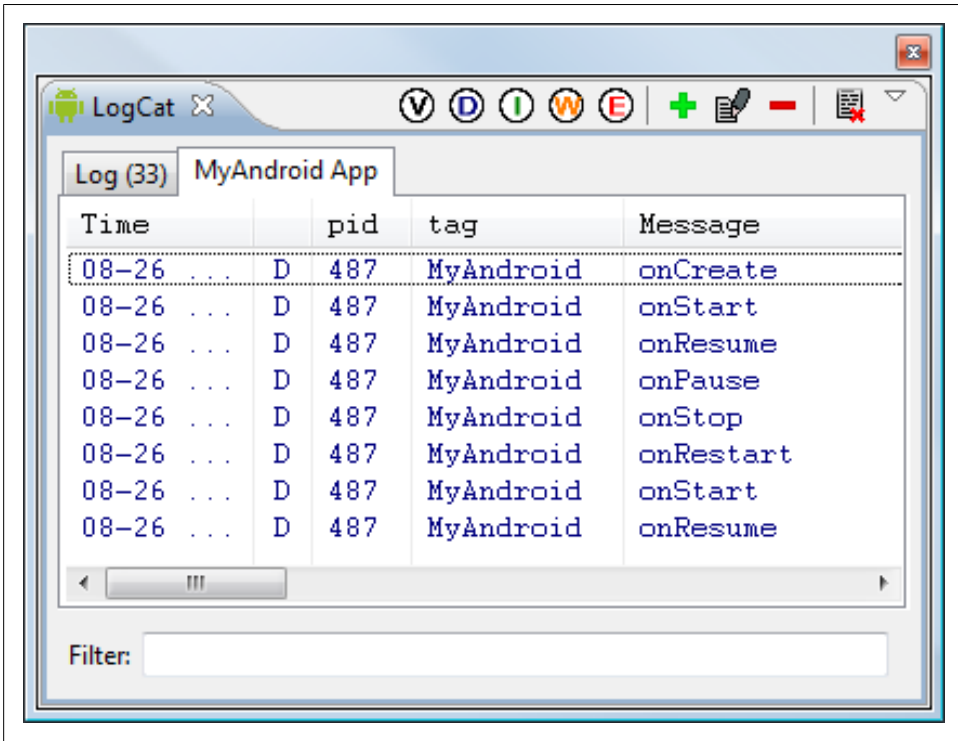


Figure 3-8.

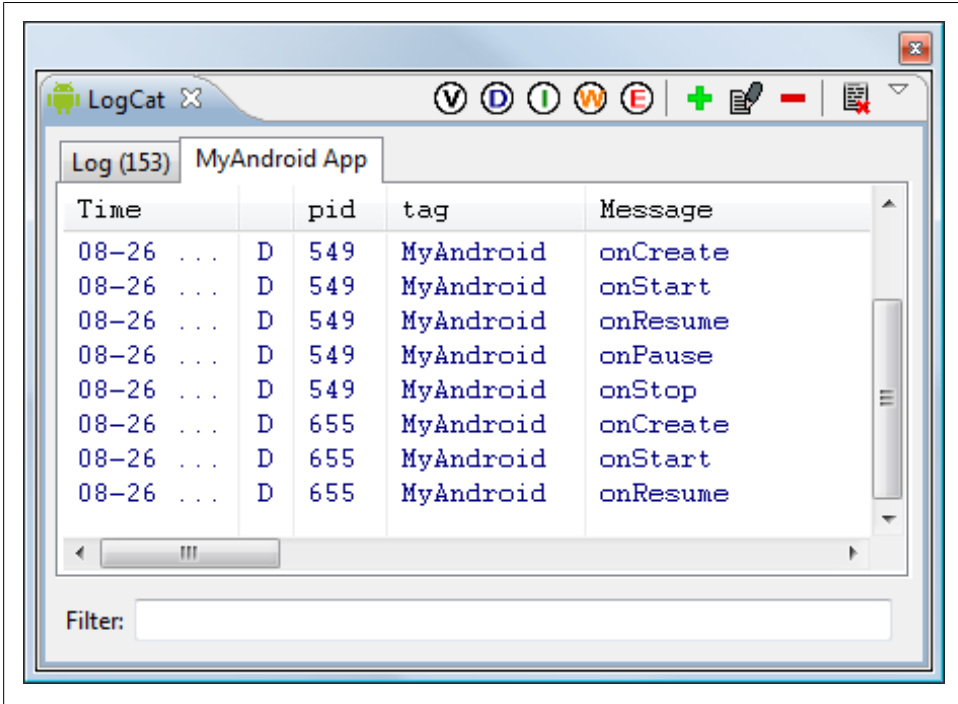


Figure 3-9.

Content Providers

4.1 Introduction: Content Providers

Ian Darwin

Discussion

The Content Provider is one of Android's more clever ideas. It allows totally unrelated applications to share data, which is usually stored in an SQLite database, without prior arrangement, knowing only the names of the tables and fields in the data.

One widely-used Content Provider is the Android Contacts data. The first Recipe in this chapter thus shows how easy it is to make an initial selection of data (this is done using an Intent, as you might guess, but it returns a URI, not the actual data). You then drill down using an SQLite cursor or two.

Then we have a recipe that shows you how to create your own Content Provider. Again as you might expect, "there's an interface for that".

Finally, while it's not directly related to Content Providers, Android also offers a more general Remote Procedure mechanism layered on IDL (interface definition language), and the Recipe for that is at the end of this chapter since it's a similar topic.

4.2 Retrieving Data from a Content Provider

Ian Darwin

Problem

You want to read from a Content Provider such as Contracts.

Solution

Create a PICK uri, open it in an Intent using `startActivityForResult`, extract the URI from the returned intent, use `Activity.getContentProvider()`, and process the data using SQLite Cursor methods.

Discussion

This is part of the Contact selection code from `TabbyText`, my SMS text message sender for WiFi-Only Honeycomb tablets (the rest of the code is in the Persistence chapter, under (BROKEN XREF TO RECIPE -1 'Reading Contacts Data')).

First, the main program sets up an `OnClickListener` to launch the Contacts app from a "Find Contact" button.

Example 4-1.

```
b.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View arg0) {
        Uri uri = ContactsContract.Contacts.CONTENT_URI;
        System.out.println(uri);
        Intent intent = new Intent(Intent.ACTION_PICK, uri);
        startActivityForResult(intent, REQ_GET_CONTACT);
    }
});
```

The URI is pre-defined for us; it actually has the value "content://com.android.contacts/contacts". The constant `REQ_GET_CONTACT` is arbitrary; it's just there to associate this Intent startup with the handler code, since more complex apps will often start more than one intent, and need to handle the results differently. Once this button is pressed, control passes from our app, out to the Contacts app. The user can then select a contact they wish to SMS. The Contacts app then is backgrounded and control returns to our app at the `onActivityResult()` method, to indicate that the activity we started has completed and delivered a result.

The next bit of code shows how the `onActivityResult()` method converts the response from the activity into an SQLite cursor.

Example 4-2.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQ_GET_CONTACT) {
        switch(resultCode) {
            case Activity.RESULT_OK:
                // The Contacts API is about the most complex to use.
                // First we have to retrieve the Contact, since we only get its URI from the Intent
                Uri resultUri = data.getData(); // e.g., content://contacts/people/123
                Cursor cont = getContentResolver().query(resultUri, null, null, null, null);
                if (!cont.moveToNext()) { // expect 001 row(s)
                    Toast.makeText(this, "Cursor contains no data", Toast.LENGTH_LONG).show();
                }
            }
        }
    }
}
```

```
        return;  
    }
```

Key things here: make sure the request code is the one you started, and the resultCode is `RESULT_OK` or `RESULT_CANCELED` (if not, pop up a warning dialog). Then, extract the URL for the response you picked - the Intent data from the returned Intent - and use that to create a Query, using the inherited Activity method `getContentResolver()` to get the ContentResolver and its `query()` method to make up an SQLite cursor.

We expect the user to have selected one Contact, so if that's not the case we error out. Otherwise we'd go ahead and use the SQLite cursor to read the data. The exact formatting of the Contact database is a bit out of scope for this recipe, so it's been deferred to the Recipe [Recipe 9.16](#).

4.3 Writing a Content Provider

Ashwini Shahapurkar

Problem

Often the application generates data, which can be processed and analyzed by another application. What is the safest way to achieve this, without giving direct access to our application's database?

Solution

Writing a custom Content Provider is the Android way to allow other applications to access data generated by our app.

Discussion

Content providers allow other applications to access the data generated by our app. For a custom content provider, we need to have the app database built up and we will be providing the wrapper over it for other applications. To make other apps aware that a content provider is available, declare it in `AndroidManifest.xml` as:

Example 4-3.

```
<provider android:authorities="com.example.android.contentprovider"  
        android:name="MyContentProvider" />
```

Here the name refers to the class `MyContentProvider` which extends `ContentProvider` class. You need to override the following methods in this class.

Example 4-4.

```
onCreate();  
delete(Uri, String, String[]);  
getType(Uri);
```



```

insert(Uri, ContentValues);
query(Uri, String[], String, String[], String);
update(Uri, ContentValues, String, String[]);

```

Usually these are wrapper functions for SQL queries on the sqlite database. You parse the input parameters and perform the queries on your database.

Example 4-5.

```

public class MyContentProvider extends ContentProvider {

    DatabaseHelper mDatabase;
    private static final int RECORDS = 1;
    public static final Uri CONTENT_URI = Uri
        .parse("content://com.example.android.contentprovider");

    public static final String AUTHORITY = "com.example.android.contentprovider";
    private static final UriMatcher matcher = new UriMatcher(
        UriMatcher.NO_MATCH);

    static {
        matcher.addURI(AUTHORITY, "records", RECORDS);
    }

    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        // the app specific code for deleting records from database goes here
        return 0;
    }

    @Override
    public String getType(Uri uri) {
        int matchType = matcher.match(uri);
        switch (matchType) {
            case RECORDS:
                return ContentResolver.CURSOR_DIR_BASE_TYPE + "/records";
            default:
                throw new IllegalArgumentException("Unknown or Invalid URI " + uri);
        }
    }

    @Override
    public Uri insert(Uri uri, ContentValues values) {
        //your app specific insertion code goes here
        // it can be as simple as follows; inserting all values in database and returning the record
        long id = mDatabase.getWritableDatabase().insert(Helper.TABLE_NAME,
            null, values);
        uri = Uri.withAppendedPath(uri, "/" + id);
        return uri;
    }

    @Override
    public boolean onCreate() {
        //initialize your database constructs
        return true;
    }
}

```

```

    }

    @Override
    public Cursor query(Uri uri, String[] projection, String selection,
        String[] selectionArgs, String sortOrder) {
        //build your query with SQLiteQueryBuilder
        SQLiteQueryBuilder qBuilder = new SQLiteQueryBuilder();
        qBuilder.setTables(Helper.TABLE_NAME);
        int uriType = matcher.match(uri);

        //query the database and get result in cursor
        Cursor resultCursor = qBuilder.query(mDatabase.getWritableDatabase(),
            projection, selection, selectionArgs, null, null, sortOrder,
            null);
        resultCursor.setNotificationUri(getContext().getContentResolver(), uri);
        return resultCursor;
    }

    @Override
    public int update(Uri uri, ContentValues values, String selection,
        String[] selectionArgs) {
        // to be implemented
        return 0;
    }
}
}

```

By providing a Content Provider you avoid giving access to your database to other developers and also reduce the chances of database inconsistency.

4.4 Android Remote Service

Rupesh Chavan

Problem

How to write remote service in the android? How to access it from other application?

Solution

Android has provided AIDL based programming interface that both the client and service agree upon in order to communicate with each other using interprocess communication(IPC).

Discussion

Inter-process communication (IPC) is the key features of the Android programming model, to achieve the above goal it has provided following two mechanisms:

1. Intent based communication

2. Remote Service based communication

In this recipe we will be concentrating on Remote service based communication approach. This android feature allows you to make method calls that look "local" but are executed in another process. They involve use of Android's interface definition language(AIDL). The service has to declare a service interface in an aidl file and the AIDL tool will automatically create a java interface corresponding to the aidl file. The AIDL tool also generates a stub class that provides an abstract implementation of the service interface methods. The actual service class will have to extend this stub class to provide the real implementation of the methods exposed through the interface.

The service clients will have to invoke the `onBind()` method on the service to be able to connect to the service. The `onBind()` method returns an object of the stub class to the client. Here are the code related code snippets:

The AIDL file:

Example 4-6.

```
package com.demoapp.service;

interface IMyRemoteService {

    String getMessage();
}
```

If you are using eclipse, it will automatically generate the Remote interface corresponding to your aidl file. The remote interface will also provide a stub inner class which has to have an implementation provided by the RemoteService class. The stub class implementation within the service class is as given here:

Example 4-7.

```
private IMyRemoteService.Stub myRemoteServiceStub = new IMyRemoteService.Stub() {
    public int getMessage() throws RemoteException {
        return "Hello World!";
    }
};
// The onBind() method in the service class:
public IBinder onBind(Intent arg0) {
    Log.d(getClass().getSimpleName(), "onBind()");
    return myRemoteServiceStub;
}
```

Now, let us quickly look at the meat of the service class before we move on to how the client connects to this service class. My RemoteService class is just returning a string. Here are the over-ridden `onCreate()`, `onStart()` and `onDestroy()` methods. `onCreate()` method of service will be called only once in a service lifecycle. `onStart()` method will be called everytime service is started. Note that the resources are all released in the `onDestroy()` method.

Example 4-8.

```
public void onCreate() {
    super.onCreate();
    Log.d(getClass().getSimpleName(),"onCreate()");
}
public void onStart(Intent intent, int startId) {
    super.onStart(intent, startId);
    Log.d(getClass().getSimpleName(), "onStart()");
}
public void onDestroy() {
    super.onDestroy();
    Log.d(getClass().getSimpleName(),"onDestroy()");
}
```

Now coming to the client class - Here, for simplicity sake, I have put the start, stop, bind, release and invoke methods all in the same client. While in reality, one client may start and another can bind to the already started service.

There are 5 buttons one each for start, stop, bind, release and invoke actions. A client needs to bind to a service before it can invoke any method on the service. Here are the start and the bind methods.

Example 4-9.

```
private void startService(){
    if (started) {
        Toast.makeText(RemoteServiceClient.this, "Service already started", Toast.LENGTH_SHORT).show();
    } else {
        Intent i = new Intent();
        i.setClassName("com.demoapp.service", "com.demoapp.service.RemoteService");
        startService(i);
        started = true;
        updateServiceStatus();
        Log.d( getClass().getSimpleName(), "startService()" );
    }
}
```

An explicit intent is created and the service is started with the `Context.startService(i)` method. Rest of the code is to update some status on the UI. There is nothing specific to a remote service invocation here. It is on the `bindService()` method that we see the difference from a local service.

Example 4-10.

```
private void bindService() {
    if(conn == null) {
        conn = new RemoteServiceConnection();
        Intent i = new Intent();
        i.setClassName("com.demoapp.service", "com.demoapp.service.RemoteService");
        bindService(i, conn, Context.BIND_AUTO_CREATE);
        updateServiceStatus();
        Log.d( getClass().getSimpleName(), "bindService()" );
    } else {
```

```

        Toast.makeText(RemoteServiceClient.this, "Cannot bind - service already bound", Toast.LENGTH_S
    }
}

```

Here we get a connection to the remote service through the RemoteServiceConnection class which implements ServiceConnection Interface. The connection object is required by the bindService() method - an intent, connection object and the type of binding are to be specified. So, how do we create a connection to the RemoteService? Here is the implementation:

Example 4-11.

```

class RemoteServiceConnection implements ServiceConnection {
    public void onServiceConnected(ComponentName className,
        IBinder boundService ) {
        remoteService = IMyRemoteService.Stub.asInterface((IBinder)boundService);
        Log.d( getClass().getSimpleName(), "onServiceConnected()" );
    }

    public void onServiceDisconnected(ComponentName className) {
        remoteService = null;
        updateServiceStatus();
        Log.d( getClass().getSimpleName(), "onServiceDisconnected" );
    }
};

```

The Context.BIND_AUTO_CREATE ensures that a service is created if one did not exist although the onStart() will be called only on explicit start of the service.

Once the client is bound to the service and the service has already started, we can invoke any of the methods that are exposed by the service. Here we have only one method and that is getMessage(). In this example, the invocation is done by clicking the invoke button. That would return the text message & update it below the button.

Let us see the invoke method:

Example 4-12.

```

private void invokeService() {
    if(conn == null) {
        Toast.makeText(RemoteServiceClient.this, "Cannot invoke - service not bound", Toast.LENGTH_SHORT
    } else {
        try {
            String message = remoteService.getCounter();
            TextView t = (TextView)findViewById(R.id.notApplicable);
            t.setText( "Message: "+message );
            Log.d( getClass().getSimpleName(), "invokeService()" );
        } catch (RemoteException re) {
            Log.e( getClass().getSimpleName(), "RemoteException" );
        }
    }
}
}

```

Once we use the service methods, we can release the service. This is done as follows (by clicking the release button):

Example 4-13.

```
private void releaseService() {
    if(conn != null) {
        unbindService(conn);
        conn = null;
        updateServiceStatus();
        Log.d( getClass().getSimpleName(), "releaseService()" );
    } else {
        Toast.makeText(RemoteServiceClient.this, "Cannot unbind - service not bound", Toast.LENGTH_SHORT).show();
    }
}
```

Finally we can stop the service by clicking the stop button. After this point no client can invoke this service.

Example 4-14.

```
private void stopService() {
    if (!started) {
        Toast.makeText(RemoteServiceClient.this, "Service not yet started", Toast.LENGTH_SHORT).show();
    } else {
        Intent i = new Intent();
        i.setClassName("com.demoapp.service", "com.demoapp.service.RemoteService");
        stopService(i);
        started = false;
        updateServiceStatus();
        Log.d( getClass().getSimpleName(), "stopService()" );
    }
}
```

These are the basics of working with a remote service on Android platform. All the best!

Note: If client & service are using different package structures then client has to include the .aidl file along with the package structure as in the service.

5.1 Introduction: Graphics

Ian Darwin

Discussion

Computer Graphics is any kind of display that there isn't a GUI component for. Charting, displaying pictures, and so on. Android is well provisioned for graphics, including a full implementation of the OpenGL EL, a subset of OpenGL intended for smaller devices.

5.2 Getting Screenshots

Ian Darwin

Problem

You need a standard way of getting screenshots for use in the Android Cookbook or any other documentation.

Solution

Use `ddms` Device->Screen Capture, and save to disk.

Discussion

The *Dalvik Debug Monitor Service* has a variety of uses. In this recipe we'll only talk about using it to get screen shots of your running device or emulator. The device does **not** have to be "rooted" for this to work, unlike many other methods of screen capture.

Start DDMS from the command line (it's in the tools directory of the SDK). After a startup delay you will see the home screen (if you also have Eclipse running (see [Rec-](#)

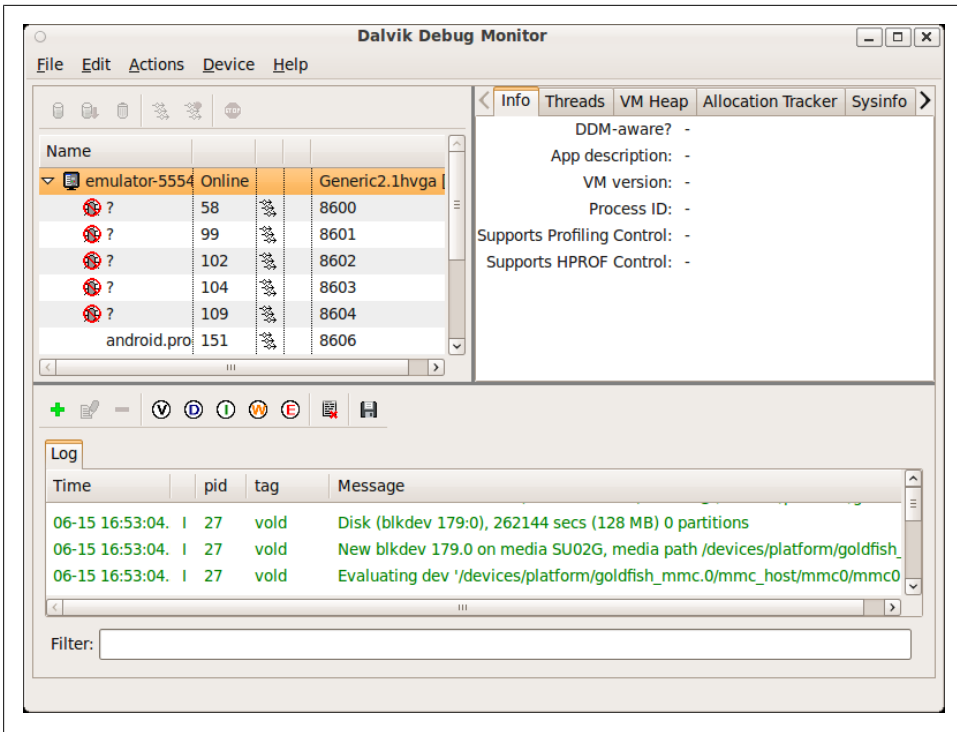


Figure 5-1.

ipe 1.4), there will be some conflict messages, which would be bad if you were debugging but are harmless here.

Select the Device menu and Screen Capture.

Ddms will communicate with the device and display the current screen contents in a dialog.

Click this button's Save button for a standard File Save dialog.

You can then upload the screen shots here, or use them in any other documentation tool. Here's how the same screen capture above would look.

5.3 Using a Custom Font

Ian Darwin

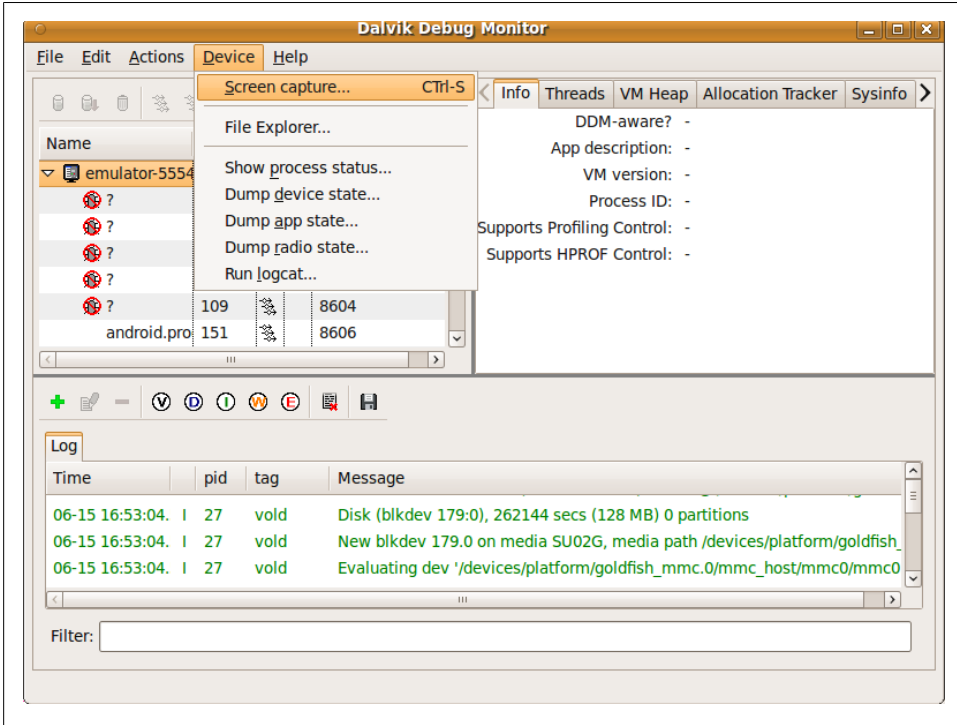


Figure 5-2.

Problem

The range of fonts that comes with Android 2.x is amazingly miniscule - three variants of the "Droid" font. You want something better.

Solution

Install a TTF or OTF version of your font in `assets/fonts` (creating this directory if needed). In your code, create a `Typeface` from the "asset", and call the `View`'s `setTypeface()` method. You're done!

Discussion

You can provide one or more fonts with your application. We have not yet seen a published way of installing system-wide fonts. So beware of huge fonts, as they will be downloaded with your application, increasing its size.

Your custom font's format should be TTF or OTF (TrueType or OpenTypeFace, a TTF extension). You need to create the "fonts" subdirectory under "assets" in your project, and install the font there.

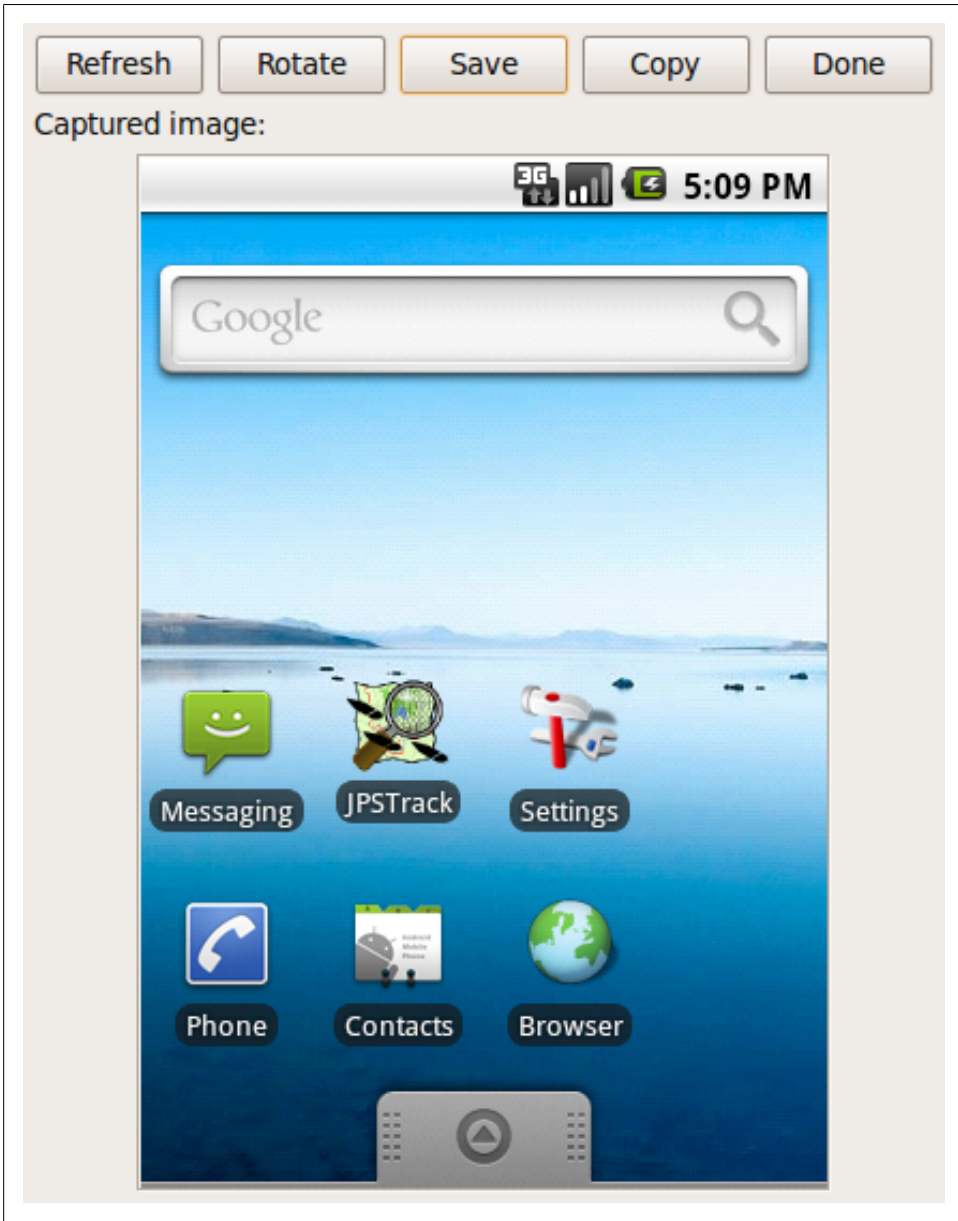


Figure 5-3.

While you can refer to the pre-defined fonts just using XML, you cannot refer to your own fonts using XML. This may change someday, but for now the content model of

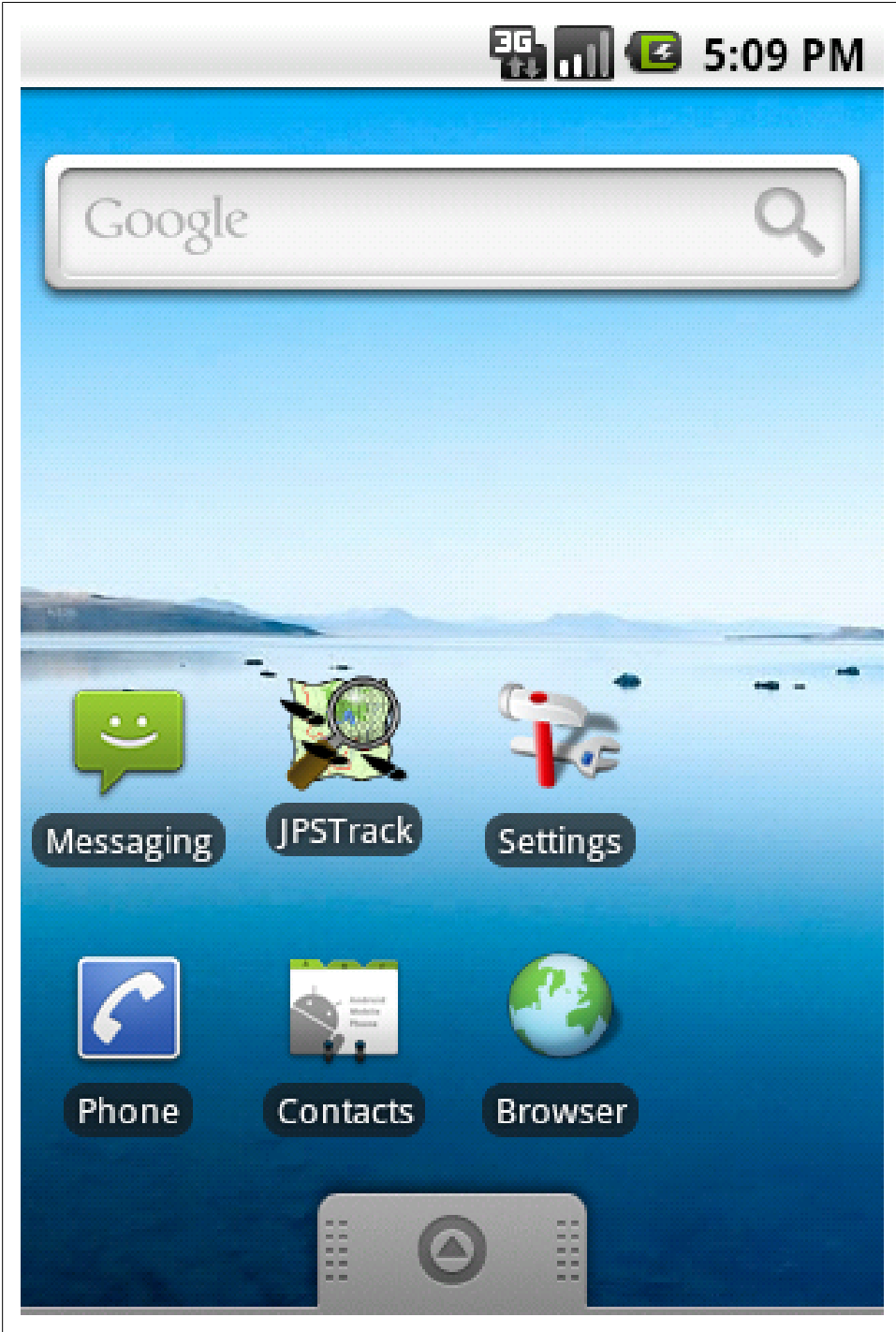


Figure 5-4.

the `android:typeface` attribute is an XML enumeration containing only "normal", "sans", "serif" and "monospace" - that's it!

So you have to use code. Basic steps are:

1. Find the View you want to use your font it;
2. Create a Typeface object from one of the Typeface class' static `create()` methods;
3. Message the Typeface into the View's `setTypeface` method.

There are several `Typeface.create()` methods, including:

- `create(String familyName, int style);`
- `create(TypeFace family, inst style);`
- `createFromAsset(AssetManager mgr, String path);`
- `createFromFile(File path);`
- `createFromFile(String path);`

You can pretty well see how most of these should work. The parameter "style" is, as in Java, one of several constants defined on the class representing fonts, here `Typeface`. Our code example uses the `createFromAsset()` method so we don't have to worry about font locations. You could probably provide a font shared by several locations using an absolute path into `"/sdcard"` using the latter two forms; remember to request permission in the `AndroidManifest` to read the SD Card! You can create representations of the built-in fonts, and variations on them, using the first two forms.

The font I used is the nice Iceberg(tm) font, from [SoftMaker Software GmbH](#). This font is copyright and I do not have permission to redistribute it, so when you download the project and want to run it, you will need to install a TrueType font file at `assets/fonts/fontdemo.ttf`. Note that if the font is invalid, Android will *silently ignore it* and use the built-in droid font.

In this demo we provide two text areas, one using the built-in "Serif" font and one using a custom font. They are defined, and various attributes added, in `mainx.xml`:

Example 5-1.

```
<?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:id="@+id/PlainTextView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/plain"
    android:textSize="36sp"
    android:typeface="serif"
```

```

        android:padding="10sp"
        android:gravity="center"
    />
<TextView
    android:id="@+id/FontView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/nicer"
    android:textSize="36sp"
    android:typeface="normal"
    android:padding="10sp"
    android:gravity="center"
    />
</LinearLayout>

```

Here is the source code:

Example 5-2.

```

public class FontDemo extends Activity {

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

        TextView v = (TextView) findViewById(R.id.FontView); // 1
        Typeface t = Typeface.createFromAsset(getAssets(), // 2
            "fonts/fontdemo.ttf");
        v.setTypeface(t, Typeface.BOLD_ITALIC); // 3
    }
}

```

If all is well, it should look like this:

Source Download URL

The source code for this example may be downloaded from this URL: <http://projects.darwinsys.com/FontDemo-src.zip>

5.4 Draw a spinning cube with OpenGL ES

Marco Dinacci

Problem

You want to create a basic OpenGL ES application.



Figure 5-5.

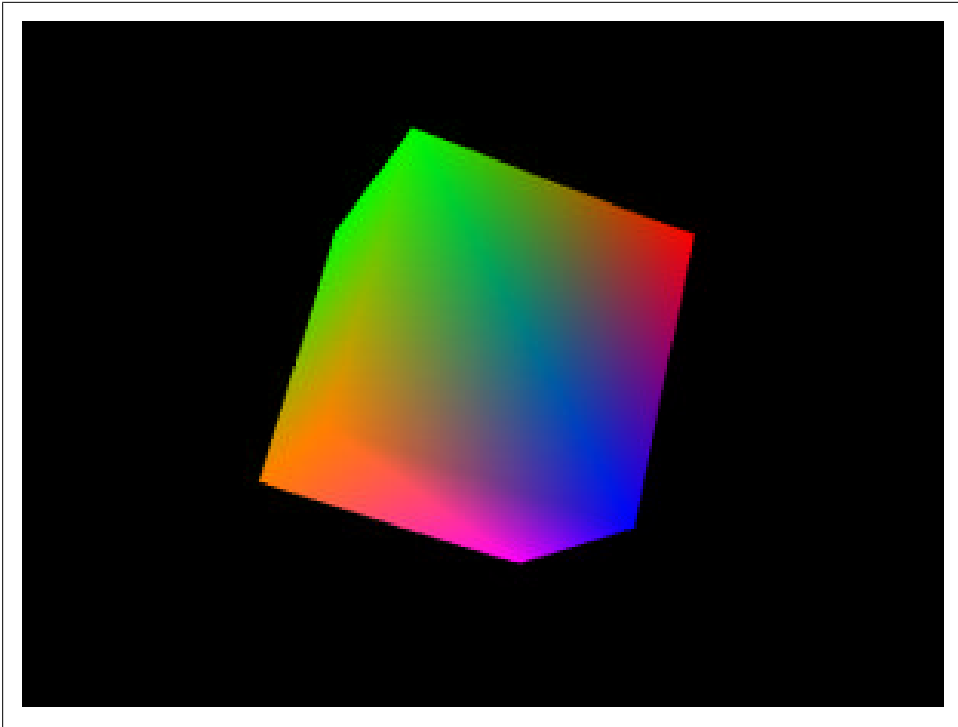


Figure 5-6.

Solution

We're going to create a `GLSurfaceView` and a custom `Renderer` that will draw a spinning cube.

Discussion

Android supports 3D graphics via the OpenGL ES API, a flavor of OpenGL specifically designed for embedded devices.

The recipe is not an OpenGL tutorial, it assumes the reader has already basic OpenGL knowledge.

The final result will look like this:

First we write a new `Activity` and in the `onCreate` method we create the two fundamental objects we need to use the OpenGL API: a `GLSurfaceView` and a `Renderer`.

Example 5-3.

```
public class OpenGLDemoActivity extends Activity {  
    @Override
```



```

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Go fullscreen
    requestWindowFeature(Window.FEATURE_NO_TITLE);
    getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
        WindowManager.LayoutParams.FLAG_FULLSCREEN);

    GLSurfaceView view = new GLSurfaceView(this);
    view.setRenderer(new OpenGLRenderer());
    setContentView(view);
}
}

```

And this is our `Renderer` that uses a simple `Cube` object we'll describe later to display a spinning cube:

Example 5-4.

```

class OpenGLRenderer implements Renderer {

    private Cube mCube = new Cube();
    private float mCubeRotation;

    @Override
    public void onSurfaceCreated(GL10 gl, EGLConfig config) {
        gl.glClearColor(0.0f, 0.0f, 0.0f, 0.5f);

        gl.glClearDepthf(1.0f);
        gl.glEnable(GL10.GL_DEPTH_TEST);
        gl.glDepthFunc(GL10.GL_LEQUAL);

        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,
            GL10.GL_NICEST);
    }

    @Override
    public void onDrawFrame(GL10 gl) {
        gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
        gl.glLoadIdentity();

        gl.glTranslatef(0.0f, 0.0f, -10.0f);
        gl.glRotatef(mCubeRotation, 1.0f, 1.0f, 1.0f);

        mCube.draw(gl);

        gl.glLoadIdentity();

        mCubeRotation -= 0.15f;
    }

    @Override
    public void onSurfaceChanged(GL10 gl, int width, int height) {
        gl.glViewport(0, 0, width, height);
    }
}

```

```

        gl.glMatrixMode(GL10.GL_PROJECTION);
        gl.glLoadIdentity();
        GLU.gluPerspective(gl, 45.0f, (float)width / (float)height, 0.1f, 100.0f);
        gl.glViewport(0, 0, width, height);

        gl.glMatrixMode(GL10.GL_MODELVIEW);
        gl.glLoadIdentity();
    }
}

```

Our `onSurfaceChanged` and `onDrawFrame` methods are basically the equivalent of the GLUT `glutReshapeFunc` and `glutDisplayFunc`. The first is called when the surface is resized, for instance when the phone switch between landscape and portrait mode, the second is called every frame and that's where we put the code to draw our cube.

Example 5-5.

```

class Cube {

    private FloatBuffer mVertexBuffer;
    private FloatBuffer mColorBuffer;
    private ByteBuffer mIndexBuffer;

    private float vertices[] = {
        -1.0f, -1.0f, -1.0f,
        1.0f, -1.0f, -1.0f,
        1.0f, 1.0f, -1.0f,
        -1.0f, 1.0f, -1.0f,
        -1.0f, -1.0f, 1.0f,
        1.0f, -1.0f, 1.0f,
        1.0f, 1.0f, 1.0f,
        -1.0f, 1.0f, 1.0f
    };

    private float colors[] = {
        0.0f, 1.0f, 0.0f, 1.0f,
        0.0f, 1.0f, 0.0f, 1.0f,
        1.0f, 0.5f, 0.0f, 1.0f,
        1.0f, 0.5f, 0.0f, 1.0f,
        1.0f, 0.0f, 0.0f, 1.0f,
        1.0f, 0.0f, 0.0f, 1.0f,
        0.0f, 0.0f, 1.0f, 1.0f,
        1.0f, 0.0f, 1.0f, 1.0f
    };

    private byte indices[] = {
        0, 4, 5, 0, 5, 1,
        1, 5, 6, 1, 6, 2,
        2, 6, 7, 2, 7, 3,
        3, 7, 4, 3, 4, 0,
        4, 7, 6, 4, 6, 5,
        3, 0, 1, 3, 1, 2
    };

    public Cube() {
        ByteBuffer byteBuf = ByteBuffer.allocateDirect(vertices.length * 4);
    }
}

```

```

byteBuf.order(ByteOrder.nativeOrder());
mVertexBuffer = byteBuf.asFloatBuffer();
mVertexBuffer.put(vertices);
mVertexBuffer.position(0);

byteBuf = ByteBuffer.allocateDirect(colors.length * 4);
byteBuf.order(ByteOrder.nativeOrder());
mColorBuffer = byteBuf.asFloatBuffer();
mColorBuffer.put(colors);
mColorBuffer.position(0);

mIndexBuffer = ByteBuffer.allocateDirect(indices.length);
mIndexBuffer.put(indices);
mIndexBuffer.position(0);
}

public void draw(GL10 gl) {
    gl.glFrontFace(GL10.GL_CW);

    gl.glVertexPointer(3, GL10.GL_FLOAT, 0, mVertexBuffer);
    gl.glColorPointer(4, GL10.GL_FLOAT, 0, mColorBuffer);

    gl.glEnableClientState(GL10.GL_VERTEX_ARRAY);
    gl.glEnableClientState(GL10.GL_COLOR_ARRAY);

    gl.glDrawElements(GL10.GL_TRIANGLES, 36, GL10.GL_UNSIGNED_BYTE,
        mIndexBuffer);

    gl.glDisableClientState(GL10.GL_VERTEX_ARRAY);
    gl.glDisableClientState(GL10.GL_COLOR_ARRAY);
}
}

```

The `Cube` uses two `FloatBuffer` objects to store vertex and color informations and a `ByteBuffer` to store the face indices. In order for the buffers to work it is important to set their order according to the endianness of the platform using the `order` method. Once the buffers have been filled with the values from the arrays, the internal cursor must be restored to the beginning of the data using `buffer.position(0)`.

See Also

<http://www.khronos.org/opengles>

5.5 Adding control to the OpenGL spinning cube

Marco Dinacci

Problem

You want to interact with an OpenGL polygon using the keyboard of your device.

Solution

We're going to create a custom `GLSurfaceView` and override the `onKeyUp` method to listens to `KeyEvent` created from the D-pad.

Discussion

This recipe extends on the "Create a spinning cube OpenGL" to show how to control the cube using a D-pad. We're going to increment the speed rotation along the X and Y axes using the directional keys of the D-pad.

The biggest change is that we now have our custom class that extends the `SurfaceView`. We do this so we can override the `onKeyUp` method and be notified when the user uses the D-pad.

The `onCreate` of our `Activity` looks like this:

Example 5-6.

```
public class SpinningCubeActivity2 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // go fullscreen
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);

        // create our custom view
        GLSurfaceView view = new OpenGLSurfaceView(this);
        view.setRenderer((Renderer)view);
        setContentView(view);
    }
}
```

Our new `GLSurfaceView` also implements the `Renderer` interface. The `onSurfaceCreated` and `onSurfaceChanged` method are exactly the same as in the previous recipe, most of the changes occur in the `onDrawFrame` as we introduce four new parameters: `mXrot` and `mYrot` to control the rotation of the cube along the X and Y axis and `mXspeed` and `mYspeed` to store the speed of the rotation along the X and Y axis.

Each time the user click on a D-pad button we alter the speed of the cube by modifying these parameters.

Here's the full code of our new class:

Example 5-7.

```
class OpenGLSurfaceView extends GLSurfaceView implements Renderer {

    private Cube mCube;
    private float mXrot;
```

```

private float mYrot;
private float mXspeed;
private float mYspeed;

public OpenGLSurfaceView(Context context) {
    super(context);

    // give focus to the GLSurfaceView
    requestFocus();
    setFocusableInTouchMode(true);

    mCube = new Cube();
}

@Override
public void onDrawFrame(GL10 gl) {
    gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);
    gl.glLoadIdentity();

    gl.glTranslatef(0.0f, 0.0f, -10.0f);

    gl.glRotatef(mXrot, 1.0f, 0.0f, 0.0f);
    gl.glRotatef(mYrot, 0.0f, 1.0f, 0.0f);

    mCube.draw(gl);

    gl.glLoadIdentity();

    mXrot += mXspeed;
    mYrot += mYspeed;
}

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if(keyCode == KeyEvent.KEYCODE_DPAD_LEFT)
        mYspeed -= 0.1f;
    else if(keyCode == KeyEvent.KEYCODE_DPAD_RIGHT)
        mYspeed += 0.1f;
    else if(keyCode == KeyEvent.KEYCODE_DPAD_UP)
        mXspeed -= 0.1f;
    else if(keyCode == KeyEvent.KEYCODE_DPAD_DOWN)
        mXspeed += 0.1f;

    return true;
}

// unchanged
@Override
public void onSurfaceCreated(GL10 gl, EGLConfig config) {
    gl.glClearColor(0.0f, 0.0f, 0.0f, 0.5f);

    gl.glClearDepthf(1.0f);
    gl.glEnable(GL10.GL_DEPTH_TEST);
    gl.glDepthFunc(GL10.GL_LEQUAL);
}

```

```

        gl.glHint(GL10.GL_PERSPECTIVE_CORRECTION_HINT,
                 GL10.GL_NICEST);
    }

    // unchanged
    @Override
    public void onSurfaceChanged(GL10 gl, int width, int height) {
        gl.glViewport(0, 0, width, height);
        gl.glMatrixMode(GL10.GL_PROJECTION);
        gl.glLoadIdentity();
        GLU.gluPerspective(gl, 45.0f, (float)width / (float)height, 0.1f, 100.0f);
        gl.glViewport(0, 0, width, height);

        gl.glMatrixMode(GL10.GL_MODELVIEW);
        gl.glLoadIdentity();
    }
}

```

The `Cube` is inherited from the previous recipe. Don't forget to call the `requestFocus()` and `setFocusableInTouchMode(true)` in the constructor of the view or else the key events will not be received.

See Also

Create a spinning cube with OpenGL ES recipe: <http://androidcookbook.com/Recipe.seam?recipeId=1418>

Source Download URL

The source code for this example may be downloaded from this URL: http://www.intransitione.com/intransitione.com/code/android/spinning_cube_controllable.zip

5.6 Taking a Picture Using an Intent

Ian Darwin

Problem

You want to take a picture from within your app and don't want to write a lot of code.

Solution

Create an Intent for `MediaStore.ACTION_IMAGE_CAPTURE`, tailor it a little, and call `startActivityForResult` on this Intent. Provide an `onActivityResult()` callback to get notified when the user is done with the camera.

Discussion

The code example below shows the complete Camera Activity from my jpstrack application.

Assuming that you want to save the image with your application's data (instead of in the Media Gallery location), you want to provide a file-based URI referring to the target location, using `intent.putExtra(MediaStore.EXTRA_OUTPUT, uri)`; . Note that, according to discussion on various forum sites, the intent handler may give significantly different results on different vendors' platforms. On the Motorola Milestone, using the Android 2.1 load from Telus Canada, with the code shown below, the defined directory gets a preview-scale image and the Media Gallery gets a 1/4-of-full-resolution (1280x960) copy. Hopefully this will be cleaned up and standardized in 2.2.

Example 5-8.

```
import jpstrack.android.MainActivity;
import jpstrack.android.FileNameUtils;

public class CameraNoteActivity extends Activity {

    private File imageFile;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // Use an Intent to get the Camera app going.
        Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        // Set up file to save image into.
        imageFile = new File(MainActivity.getDataDir(),
            FileNameUtils.getNextFilename("jpg"));
        Uri uri = Uri.fromFile(imageFile);
        intent.putExtra(MediaStore.EXTRA_OUTPUT, uri);
        intent.putExtra(MediaStore.EXTRA_VIDEO_QUALITY, 1);
        // And away we go!
        startActivityForResult(intent, 0);
    }

    @Override
    public void onActivityResult(int requestCode,
        int resultCode, Intent data) {
        switch(requestCode) {
            case 0: // take picture
                switch(resultCode) {
                    case Activity.RESULT_OK:
                        if (imageFile.exists())
                            Toast.makeText(this,
                                "Bitmap saved as " + imageFile.getAbsolutePath(),
                                Toast.LENGTH_LONG).show();
                        else {
                            AlertDialog.Builder alert =
                                new AlertDialog.Builder(this);
                            alert.setTitle("Error").setMessage(
```

```

        "Returned OK but image not created!").show();
    }
    break;
case Activity.RESULT_CANCELED:
    // no blather required!
    break;
default:
    Toast.makeText(this,
        "Unexpected resultCode: " + resultCode,
        Toast.LENGTH_LONG).show();
    }
    break;
default:
    Toast.makeText(this,
        "UNEXPECTED ACTIVITY COMPLETION",
        Toast.LENGTH_LONG).show();
    }
    finish();    // back to main app
}
}
}

```

See Also

[Recipe 5.7](#) - doing it this way gives you more control.

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.darwinsys.com/jpstrack/>

Binary Download URL

The executable code for this example may be downloaded from this URL: <http://www.darwinsys.com/jpstrack/jpstrack.android.apk>

5.7 Taking a Picture Using `android.media.Camera`

Marco Dinacci

Problem

You want to have more control on the various stages involved when taking a picture.

Solution

Create a `SurfaceView` and implement the callbacks fired when the user takes a picture in order to have control over the whole process of image capturing.

Discussion

Sometimes you may want more control over the stages involved when taking a picture or you may want to access and modify the raw image data acquired by the camera. In these cases, using a simple Intent to take a picture is not enough.

We're going to create a new Activity and customize the view to make it full screen inside the onCreate method.

Example 5-9.

```
public class TakePictureActivity extends Activity {
    private Preview mCameraView;

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

        // Force screen in landscape mode as showing a video in
        // potrait mode is not easily doable on all devices
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

        // Hide window title and go fullscreen
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN);

        mCameraView= new Preview(this);
        setContentView(mCameraView);
    }
}
```

The Preview class is the bulk of the recipe. It handle the Surface where the pixels are drawn and the Camera object.

We define a ClickListener in the constructor so the user can take a picture by just tapping once on the screen. Once we get the notification of the click we take a picture passing as parameters four (all optional) callbacks.

Example 5-10.

```
class Preview extends SurfaceView implements SurfaceHolder.Callback, PictureCallback {

    private SurfaceHolder mHolder;
    private Camera mCamera;
    private RawCallback mRawCallback;

    public Preview(Context context) {
        super(context);

        mHolder = getHolder();
        mHolder.addCallback(this);
        mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
        mRawCallback = new RawCallback();
    }
}
```

```

setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View v) {
        mCamera.takePicture(mRawCallback, mRawCallback, null,
            Preview.this);
    }
});
}

```

The `Preview` class implement the `SurfaceHolder.Callback` interface in order to be notified when underlying surface is created, changed and destroyed. We'll use these callbacks to properly handle the `Camera` object.

Example 5-11.

```

@Override
public void surfaceChanged(SurfaceHolder holder, int format, int width,
    int height) {

    Camera.Parameters parameters = mCamera.getParameters();
    parameters.setPreviewSize(width, height);
    mCamera.setParameters(parameters);

    mCamera.startPreview();
}

@Override
public void surfaceCreated(SurfaceHolder holder) {
    mCamera = Camera.open();

    configure(mCamera);

    try {
        mCamera.setPreviewDisplay(holder);
    } catch (IOException exception) {
        closeCamera();
    }
}

@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    closeCamera();
}

```

As soon as the camera is created we call `configure` in order to set the parameters the camera will use to take a picture. Things like flash mode, effects, picture format, picture size, scene mode and so on. Since not all devices support all kind of features always ask which features are supported before setting them.

Example 5-12.

```

private void configure(Camera camera) {
    Camera.Parameters params = camera.getParameters();

```

```

// Configure image format. RGB_565 is the most common format.
List<Integer> formats = params.getSupportedPictureFormats();
if (formats.contains(PixelFormat.RGB_565))
    params.setPictureFormat(PixelFormat.RGB_565);
else
    params.setPictureFormat(PixelFormat.JPEG);

// Choose the biggest picture size supported by the hardware
List<Size> sizes = params.getSupportedPictureSizes();
Camera.Size size = sizes.get(sizes.size()-1);
params.setPictureSize(size.width, size.height);

List<String> flashModes = params.getSupportedFlashModes();
if (flashModes.size() > 0)
    params.setFlashMode(Camera.Parameters.FLASH_MODE_AUTO);

// Action mode take pictures of fast moving objects
List<String> sceneModes = params.getSupportedSceneModes();
if (sceneModes.contains(Camera.Parameters.SCENE_MODE_ACTION))
    params.setSceneMode(Camera.Parameters.SCENE_MODE_ACTION);
else
    params.setSceneMode(Camera.Parameters.SCENE_MODE_AUTO);

// if you choose FOCUS_MODE_AUTO remember to call autoFocus() on
// the Camera object before taking a picture
params.setFocusMode(Camera.Parameters.FOCUS_MODE_FIXED);

camera.setParameters(params);
}

```

When the surface is destroyed we close the camera and free its resources:

Example 5-13.

```

private void closeCamera() {
    if (mCamera != null) {
        mCamera.stopPreview();
        mCamera.release();
        mCamera = null;
    }
}

```

The Jpeg callback is the last one called, this is where we restart the preview and save the file on disk.

Example 5-14.

```

@Override
public void onPictureTaken(byte[] jpeg, Camera camera) {
    // now that all the callbacks have been called it is safe to resume the preview
    mCamera.startPreview();

    saveFile(jpeg);
}

```

```
}  
}
```

Finally we implement the `ShutterCallback` and again `PictureCallback` to receive the uncompressed raw image data.

Example 5-15.

```
class RawCallback implements ShutterCallback, PictureCallback {  
  
    @Override  
    public void onShutter() {  
        // notify the user, normally with a sound, that the picture has  
        // been taken  
    }  
  
    @Override  
    public void onPictureTaken(byte[] data, Camera camera) {  
        // manipulate uncompressed image data  
    }  
}
```

See Also

[Recipe 5.6](#)

5.8 Using AndroidPlot to display charts and graphs in your Android application.

Rachee Singh

Problem

Depicting data graphically in an Android application.

Solution

There are many 3rd party graph libraries for Android available. In this example we make use of AndroidPlot library (open source) to depict a simple graph.

Discussion

Step 1: Download AndroidPlot library from here: <http://androidplot.com/wiki/Download> (any version).

Step 2: Create a new Android project and add the AndroidPlot library to the new project. To do this, create a new folder in the project folder by the name 'lib'. To this folder add the downloaded AndroidPlot jar file (named something like this: 'Android-

plot-core-0.4a-release.jar'). (At this stage you should be having directories like src, res, gen, lib.)

Step 3: To use the library, it must be added to the build path. For this, in Eclipse, right click the .jar you added and select the "Build Path -- Add to Build Path" option. This will show another directory called 'Referenced Libraries' in the Eclipse project.

Step 4: In our sample application, we are hard-coding some data and showing the plot corresponding to the data in the application. So, we require to add an XY plot to our XML layout (main.xml). Here's what main.xml looks like with a XYPlot component in a linear layout:

Example 5-16.

```
&lt;?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"
    >
    <com.androidplot.xy.XYPlot
        android:id="@+id/mySimpleXYPlot"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        title="Stats"/>
</LinearLayout>
```

Step 5: Get a reference to the XYplot defined in the XML.

Example 5-17.

```
mySimpleXYPlot = (XYPlot) findViewById(R.id.mySimpleXYPlot);
```

Step 6: Initialize two arrays of numbers for which the plot will be displayed.

Example 5-18.

```
// Create two arrays of y-values to plot:
Number[] series1Numbers = {1, 8, 5, 2, 7, 4};
Number[] series2Numbers = {4, 6, 3, 8, 2, 10};
```

Step 7: Turn the above arrays into XYSeries:

Example 5-19.

```
XYSeries series1 = new SimpleXYSeries(
    Arrays.asList(series1Numbers), // SimpleXYSeries takes a List so turn our array into a List
    SimpleXYSeries.ArrayFormat.Y_VALS_ONLY, // Y_VALS_ONLY means use the element index as the x value
    "Series1"); // Set the display title of the series
XYSeries series2 = new SimpleXYSeries(Arrays.asList(series2Numbers), SimpleXYSeries.ArrayFormat.Y_VALS_ONLY,
    "Series2");
```

Step 8: Create a formatter to use for drawing a series using LineAndPointRenderer:

Example 5-20.

```
LineAndPointFormatter series1Format = new LineAndPointFormatter(  
    Color.rgb(0, 200, 0),           // line color  
    Color.rgb(0, 100, 0),          // point color  
    Color.rgb(150, 190, 150));     // fill color (optional)
```

Step 9: Add series1 and series2 to the xyplot:

Example 5-21.

```
mySimpleXYPlot.addSeries(series1, series1Format);  
mySimpleXYPlot.addSeries(series2, new LineAndPointFormatter(Color.rgb(0, 0, 200), Color.rgb(0, 0, 100),
```

Step 10: Make it look cleaner:

Example 5-22.

```
// Reduce the number of range labels  
mySimpleXYPlot.setTicksPerRangeLabel(3);  
  
// By default, AndroidPlot displays developer guides to aid in laying out your plot.  
// To get rid of them call disableAllMarkup():  
mySimpleXYPlot.disableAllMarkup();  
  
mySimpleXYPlot.getBackgroundPaint().setAlpha(0);  
mySimpleXYPlot.getGraphWidget().getBackgroundPaint().setAlpha(0);  
mySimpleXYPlot.getGraphWidget().getGridBackgroundPaint().setAlpha(0);
```

Step 11: Run the application! This is how it looks:

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LNTjMDQ2MTktZjAzMi00ZjBkLWFhOTktZjA5OWY4YjE2MTRh&hl=en_US

Binary Download URL

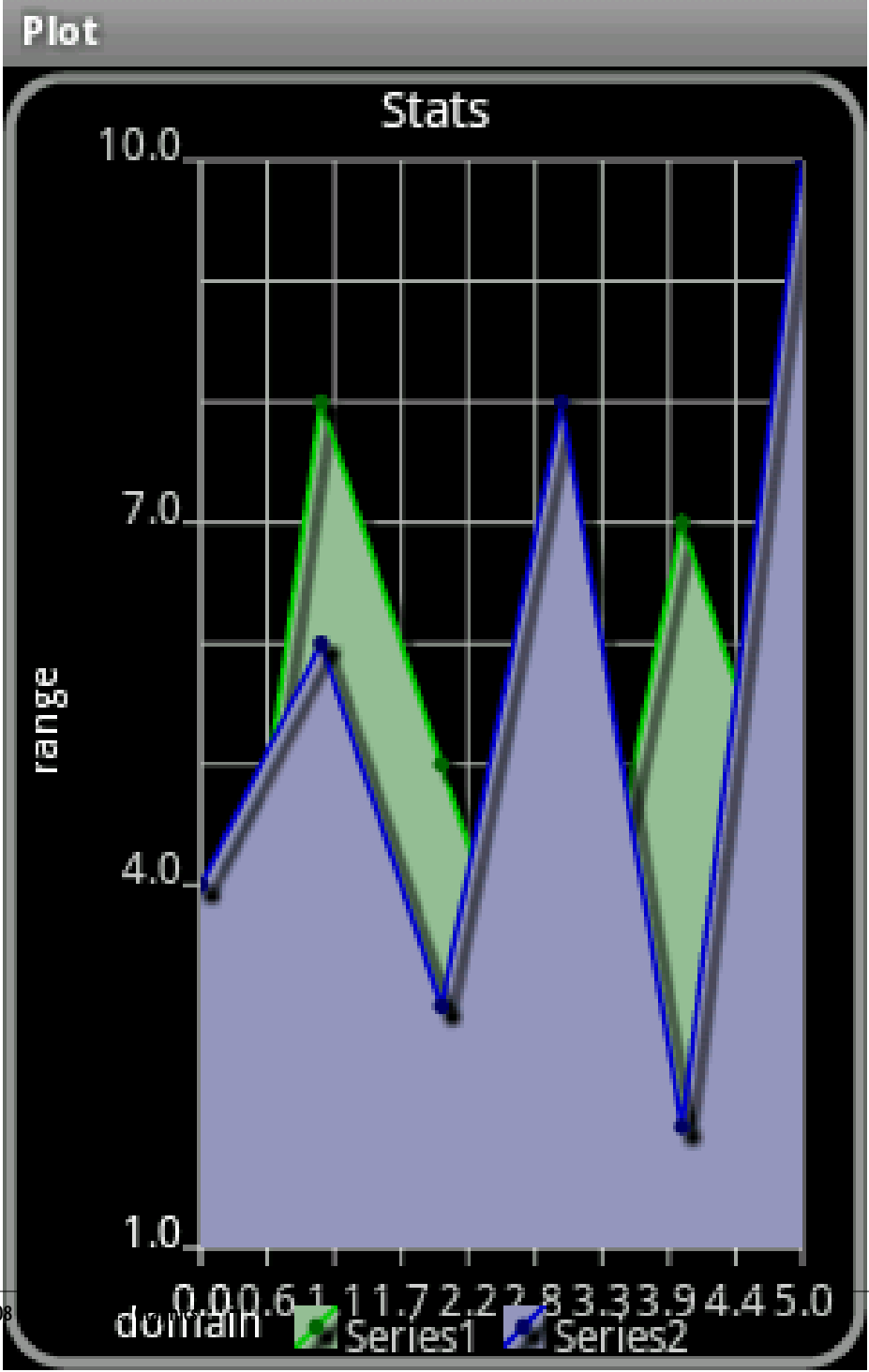
The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LZmI5ODEyZDgtNGNi00NTE1LWE3OWMtMjcxYTQwOTg1Y2Vh&hl=en_US

5.9 Use Inkscape to Create an Android Launcher Icon

Daniel Fowler

Problem

Every good Android App deserves a custom a launcher icon.



208

Figure 5-7.

Solution

Inkscape is a free and feature rich graphics program, it supports exporting to a bitmap file; this can be used to create the different size icons needed for an App.

Discussion

A graphics program is used to design graphical resources used in an Android application (App). Inkscape is a free multiplatform graphics program and has some very powerful features. It can be used to generate vector graphic images to a high standard. These images can then be exported to any required resolution. This is ideal for generating Android launcher icons (and other graphical resources). See the Inkscape web site at <http://inkscape.org/> for more information on the program and to download the latest version.

When a project is created in Eclipse a default icon is generated in the `res/drawable` folder. This default icon is 48x48 pixels. Icons are stored in the Portable Network Graphics (PNG) file format. Android supports different screen densities, measured in Dots Per Inch (dpi). Screen densities are grouped into low density (120 dpi), medium density (160 dpi), high density (240 dpi) and extra high density (320 dpi). The 48x48 pixel icon is suitable for medium density screens, for all other densities the 48x48 pixel icon is scaled up or down as required. Ideally for best results (sharp images with no pixelation) a project will include an icon for all the possible screen densities that an App will encounter. To do this four drawable folders are created under the `res` folder, one for each possible screen density; icon files of the correct size are placed into these directories:

- 36x36 pixel icon in `res/drawable-ldpi` for low density screens
- 48x48 pixel icon in `res/drawable-mdpi` for medium density screens
- 72x72 pixel icon in `res/drawable-hdpi` for high density screens
- 96x96 pixel icon in `res/drawable-xhdpi` for extra high density screens

Each icon must include a border around the central image, used for on screen spacing and minor image protrusions. The recommended border is one twelfth of the icon size. This means that the space the actual icon image occupies is smaller than the icon pixel size:

- 36x36 icon the image size is 30x30 pixels
- 48x48 icon the image size is 40x40 pixels
- 72x72 icon the image size is 60x60 pixels
- 96x96 icon the image size is 80x80 pixels

When designing an icon it is better to work with images that are larger than the required size. A larger image is easier to work with in a graphics package and easily scaled down when completed. An image that is 576x576 pixels is divisible equally by all the icon

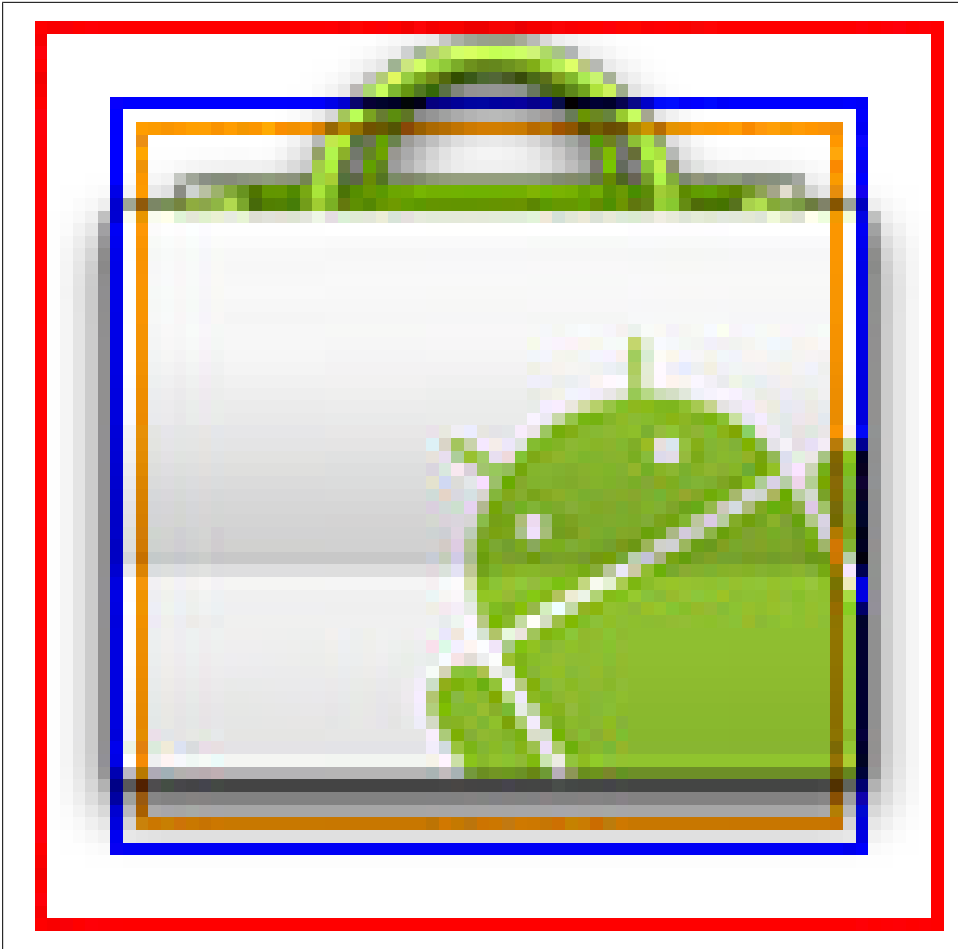


Figure 5-8.

sizes and is a reasonable size in which to design. For a vector based graphics package, such as Inkscape, the image size is irrelevant; it can be scaled up and down without losing quality. Inkscape uses the open Scalable Vector Graphics (SVG) format. Image detail is only lost when the final bitmap images are produced from the vector image.

Those wanting to learn to design images in Inkscape can use the many tutorials available. There are built in tutorials (via the Help menu) and many online, <http://inkscape.tutorials.wordpress.com/> is a good tutorial reference.

Once an image has been designed in Inkscape it can be exported to a PNG file for use as an App icon. In the following example the image that is to be converted to icons was generated from the tutorial at <http://vector.tutsplus.com/tutorials/illustration/creating-a-coffee-cup-with-inkscape/>. If the tutorial is followed this cup of coffee image is produced:



Figure 5-9.

This can be converted to an icon for a Coffee Ordering/Coffee Break Timer/Coffee Break Game or whatever coffee related App is currently in the pipeline. For those who do not want to follow the tutorial the image can be obtained from <http://openclipart.org>, a great source (over 33,000) of free images. Search for coffee and you will see various coffee related images including the one here, uploaded by this recipe's author. Click on the image, select the **View SVG** button and use the browser's **File->Save Page As** (Firefox) or **File->Save As** (Internet Explorer) menu to get the cup of coffee image.

The four required icon sizes are generated from the image using the Inkscape **Export Bitmap** option. The image is opened and then correctly proportioned for the export. This can be done for any image designed or opened in Inkscape. Remember that images for icons should not be overly detailed or have too many colors (detail is reduced during resizing), and try to fill a square area. Android icon guidelines also suggest images that are face on with minor drop shadows and a little top lighting, see http://developer.android.com/guide/practices/ui_guidelines/icon_design_launcher.html.

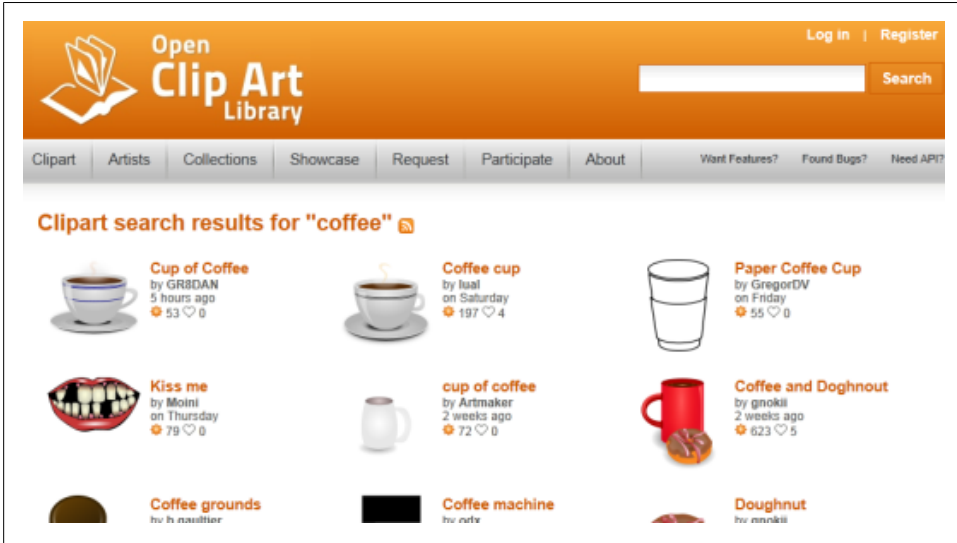


Figure 5-10.

With the image open resize the document to 576x576 pixels. To do this use the Document Properties option under the File menu. In Custom size set Width and Height to 576 and check Units are set to px (for pixels). Ensure that the Show page border check box is ticked.

Drag two vertical and two horizontal guides from the rulers (click and drag from any part of the page ruler). Drag them inside each page border approximately 1/12 of the width and height of the visible page border. The accurate position of the guides will be set using the guide properties. If the rulers are not visible use the View->Show/Hide->Rulers menu option to display them. Double click each guide and set the following positions accurately:

Guide	x	y
Top Horizontal	0	528
Bottom Horizontal	0	48
Left Vertical	48	0
Right Vertical	528	0

With the guides in place the image can be easily adjusted to fit within them. Minor protruding into the border area is allowable if required for image balance. Use the menu Edit->Select All or press Ctrl-A to select all the image, drag the image into position and resize as appropriate to fit within the box outlined by the guides.

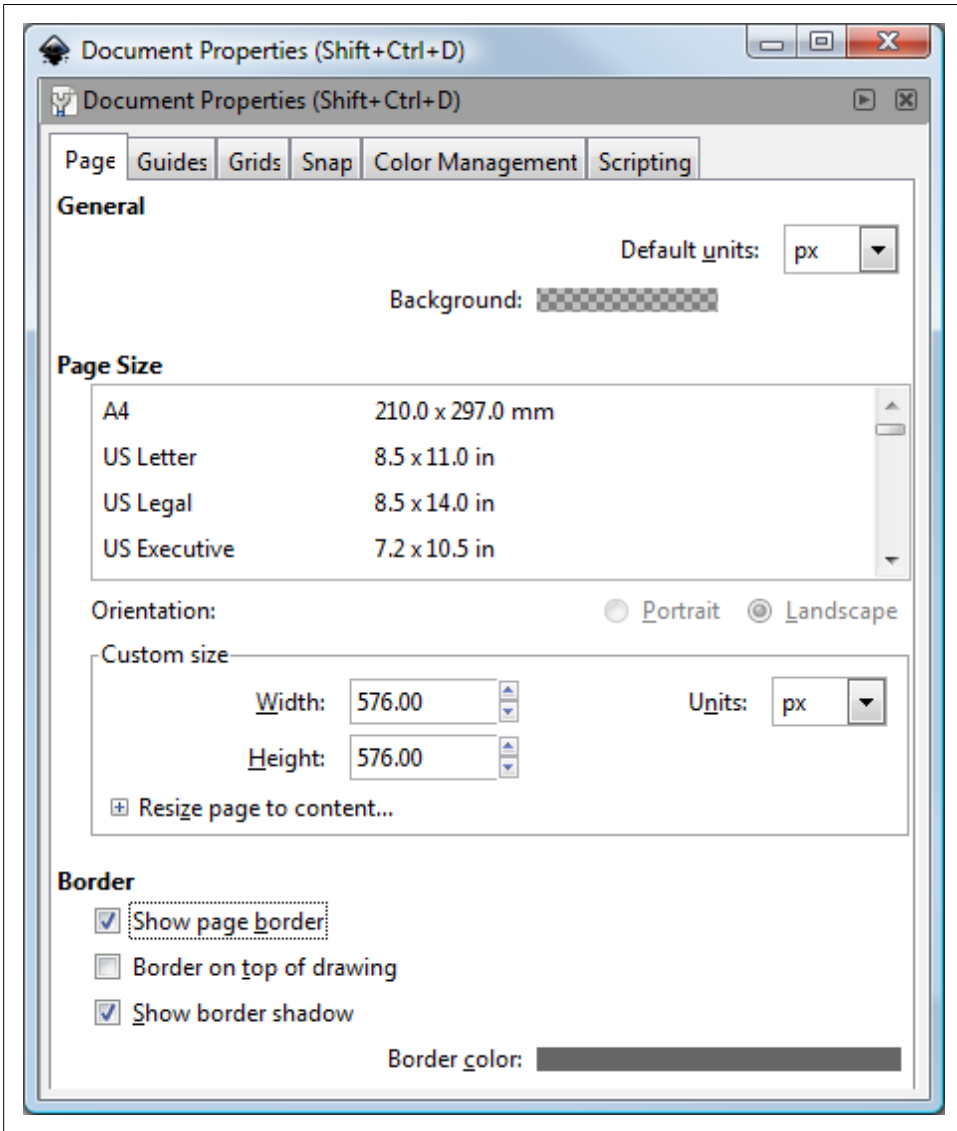


Figure 5-11.

With the image created and correctly proportioned the bitmaps for an Android project can be created. Using Eclipse open the project in which the icons are required. Select the `res` folder and create four new folders (menu option `File->New->Folder` or context menu `New->Folder`):

- `res/drawable-ldpi`

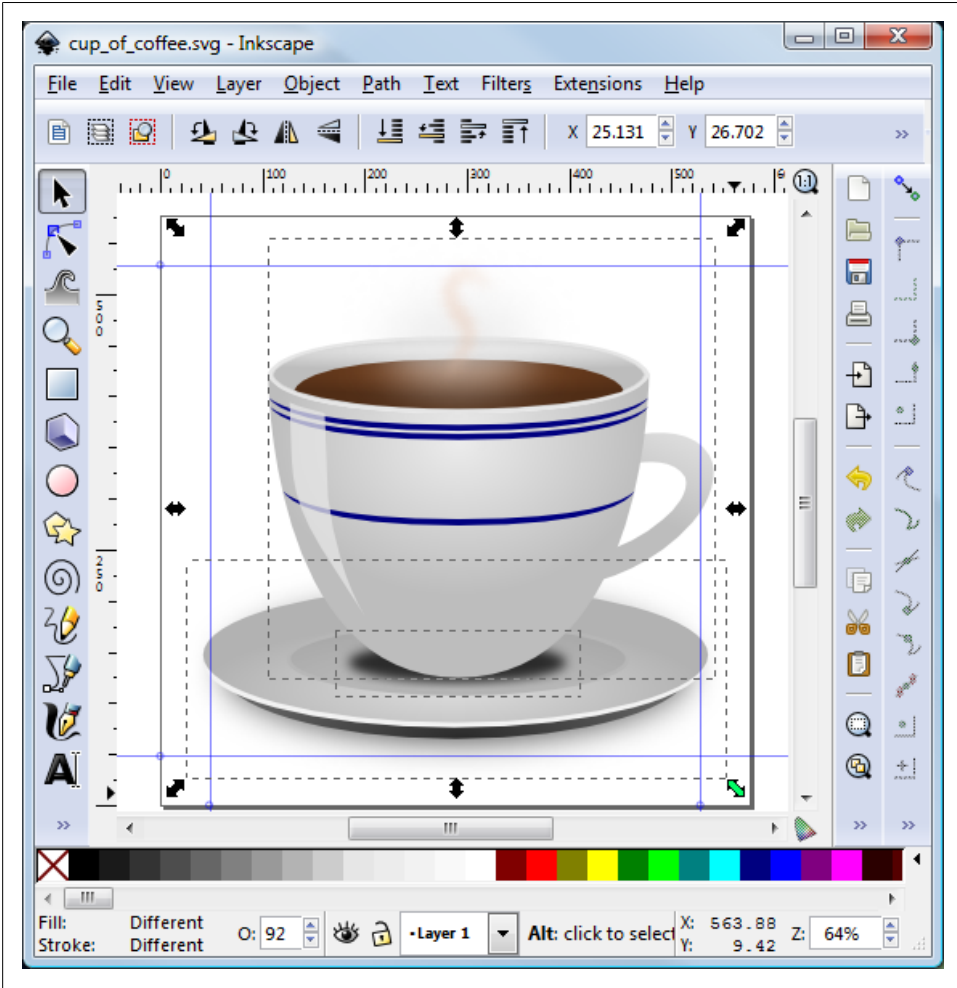


Figure 5-12.

- res/drawable-mdpi
- res/drawable-hdpi
- res/drawable-xhdpi

The existing `drawable` folder is used as fallback if an icon cannot be found or for Apps that can run on Android 1.5.

Back in Inkscape ensure that the image is not selected (click outside the image). Use the **File-Export Bitmap** menu option to bring up the **Export Bitmap** dialog. Select **Page**, then under **Bitmap Size** set **Width** and **Height** to 96, the dpi setting does not need to be changed (it will change as **Width** and **Height** are changed). Under **Filename** browse

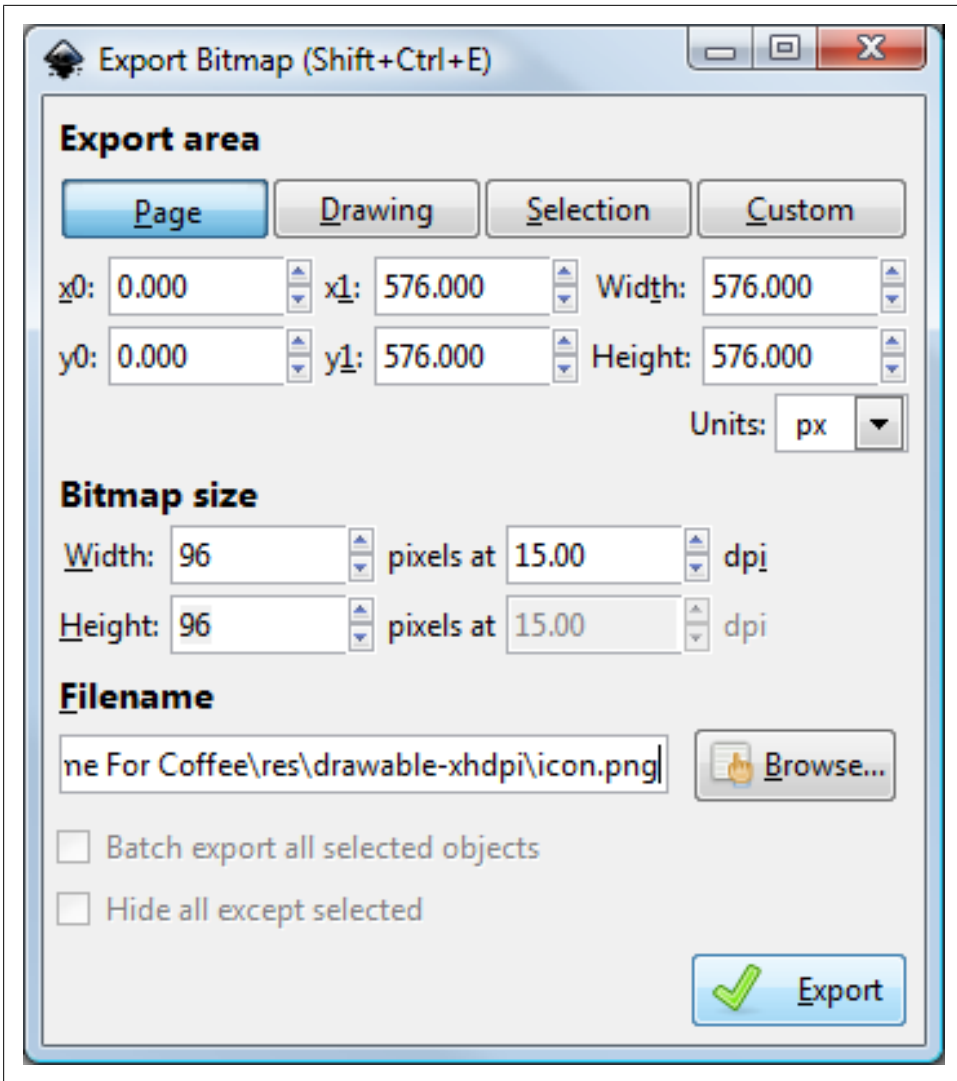


Figure 5-13.

to the project directory for the xhdpi icon (res/drawable-xhdpi) and enter icon.png for the file name. Press the **Export** button to generate the icon.

For the other three icon resolutions set **Width** and **Height** appropriately (72 then 48 and finally 36) and browse to the correct folder to export each icon. Finally copy the icon from the `res/drawable-mdpi` folder into the `drawable` folder to replace the default icon. This process will have generated the different size icons required to support different device screens.



Figure 5-14.

If Eclipse was open when the icons are generated the open project will need to be refreshed to see the new icons in the folders, select **File->Refresh** or press **F5**.

The Application can be tested on physical and virtual devices to ensure the icons appear as expected.

The icon files do not need to be called `icon.png`, see the recipe [Recipe 5.10](#) for information on changing the launcher icon file name.

See Also

<http://inkscape.org/>

<http://linkscapetutorials.wordpress.com/>

<http://vector.tutsplus.com/tutorials/illustration/creating-a-coffee-cup-with-inkscape/>

<http://openclipart.org>

http://developer.android.com/guide/practices/ui_guidelines/icon_design_launcher.html

[Recipe 5.10](#)

5.10 Easy Launcher Icons from OpenClipArt.org using Paint.NET

Daniel Fowler

Problem

A good icon for an App is important. It helps set an App apart from others and is a must for an App to appear professional.

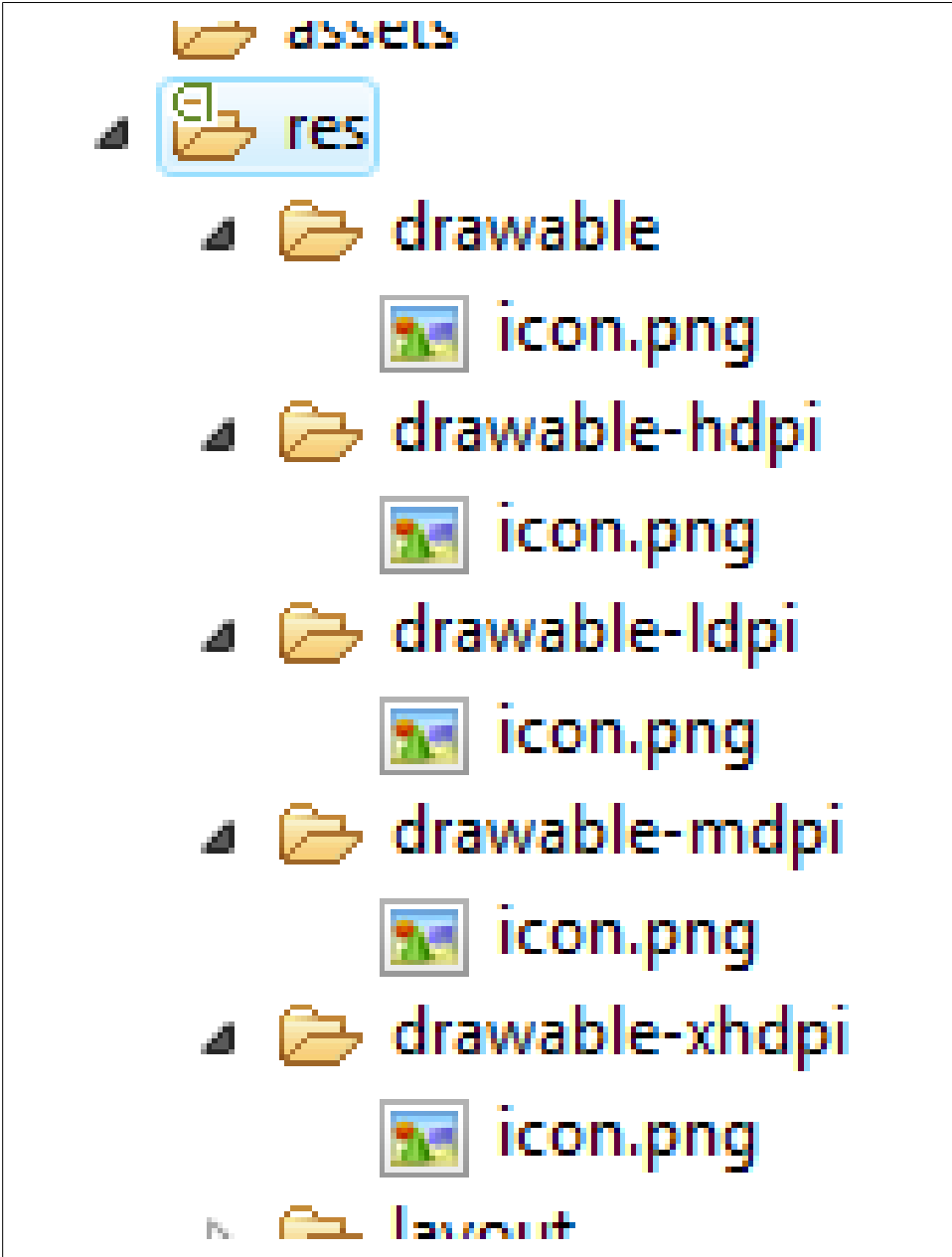


Figure 5-15.

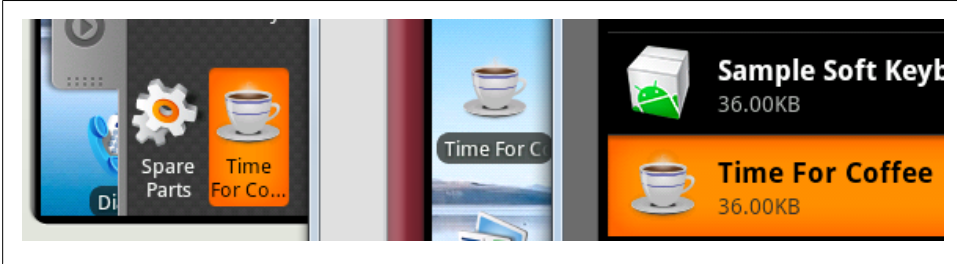


Figure 5-16.

Solution

Developers sometimes struggle to produce good graphics, such as icons, fortunately a good source of free graphics that can be adapted for icons is available.

Discussion

When an Application is close to being released consideration needs to be given to the tasks to get it ready for the Android Market. One of those tasks is to provide a good icon. The icon will usually be the most common graphical representation of the App that a user encounters. It will represent the App on the Applications screen, in Manage Applications and as a shortcut if added to the Home Screen. A good icon helps the initial impression of the App and helps with getting it to stand out from the crowd. Developers with access to a graphical artist, either professionally or through friends, or are good artists themselves will have finer control on the graphics within their Application. However, there are many who find that doing the graphics in an App is a chore. This recipe shows how to generate a good icon quickly, though compromising the fine control provided by a dedicated artist.

The website Open Clip Art Library at <http://www.openclipart.org> is a great source for free graphics, over 33,000. The graphics provided are in vector format which make them great for scaling to icon size. Icons are a raster format so once a suitable graphic has been chosen it needs to be converted into the Android icon format which is Portable Network Graphics (PNG).

Taking the [Recipe 1.4](#) as an example App and adding an icon.

First find a suitable free graphic as a starting point. Go to <http://www.openclipart.org> and use the **Search** box. The search results may include graphics that do not always appear logical. This is because the search not only includes the name of the graphic, but also tags and descriptions, and partial words; therefore graphics unrelated to the major search term will appear. As well as contributions with misspellings or named in a different language. However, this also means that occasionally an unexpected but suitable graphic will be found. Page through the search results which are provided as thumbnails with title, contributor and date of submission, and number of downloads.

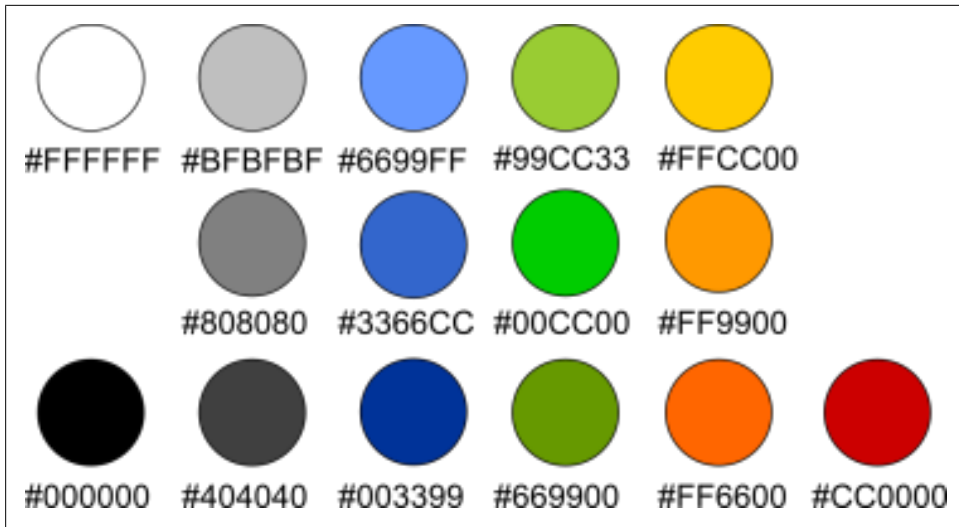


Figure 5-17.

When looking for a graphic to use as an icon there are some pointers to keep in mind:

- There is a recommended colour palette to fit in with the Android theme, this is only a recommendation but a useful guide. Avoid color that is too extreme.
- The graphic will be scaled down dramatically, so not too much detail, the search thumbnail itself is a good indicator.
- Clear and simple designs with smooth lines and bright, neutral colors will scale well and look good on a device screen.
- Keep in mind Android design guidelines at http://developer.android.com/guide/practices/ui_guidelines/icon_design_launcher.html, this means graphical representations that are face on, with a small drop shadow and top lighting.
- Icons are square, so look for an image that if bounded by a square it would fill most of that square.

For the *Hello World App* the search term **earth** was used.

The graphic titled *A simple globe* was chosen as the basis for the icon from the second page of search results. Click on the graphic to bring up its details. The graphic can be saved to the local machine by clicking on it (or click on the **View SVG** button) and using the browser's **File** menu. In Firefox select **Save Page As** and select its location. In Internet Explorer select **Save as...**, or both browsers support **Ctrl-S**. This will save the file as a vector file, SVG, no good as an icon. Fortunately the images's Open Clip Art page also has an option to obtain the file as a PNG file.



Figure 5-18.

Android icons need to be provided in four different sizes. These different sizes are to allow Android to display the best possible icon for the device's screen density. It is recommended that an App supplies all the size icons required thus preventing poor icons being displayed on some devices. The four icon sizes are:

- 36 by 36 pixels for low density displays (120 dpi)
- 48 by 48 pixels for medium density displays (160 dpi)
- 72 by 72 pixels for high density displays (240 dpi)
- 96 by 96 pixels for extra high density displays (320 dpi)

There is also a border to take into consideration, the border area allows for spacing and image overrun and is recommended to be one twelfth of the icon width.

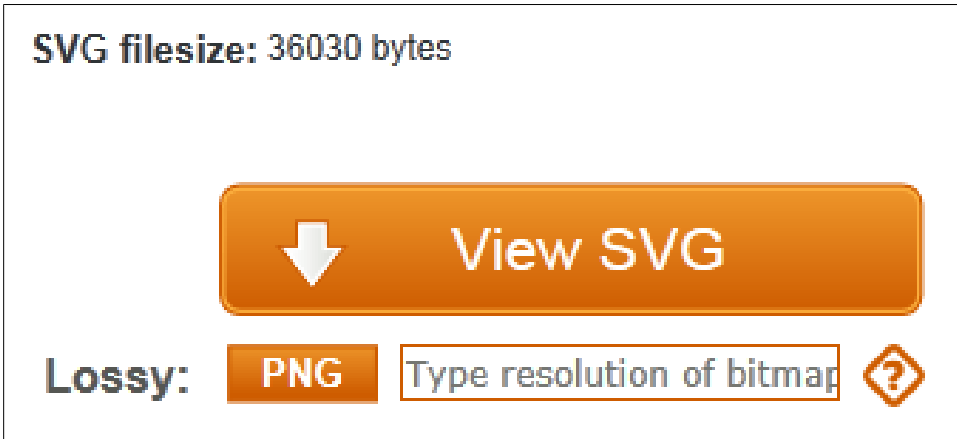


Figure 5-19.

This means that the practical image size for the icon graphic is smaller than the stated icon size.

- 30 by 30 pixels for low density
- 40 by 40 pixels for medium density
- 60 by 60 pixels for high density
- 80 by 80 pixels for extra high density

On the Open Clip Art page for the required graphic we can use the PNG button to obtain a PNG in the four image sizes required. In the box next to the PNG button type in the first image size required, 80 (for the extra high density icon). We cannot put in the icon size, 96, because that would not leave any border.

Click on the PNG button and then use the browser's File menu (or Ctrl-S) to save the generated PNG file. Hit the browser's back button to return to the image's web page. Clear the box next to the PNG button and enter the size of the next icon graphic required, in this case 60 for the high density icon. Again click the PNG button and save the generated file. Do the same with the values 40 and 30 to generate the other two graphics.

A couple of problems may occur. Sometimes the conversion will still produce the previous size graphic. If this happens reload the image's Open Clip Art page (click on the address bar and with the cursor at the end of the address hit enter, using F5 will not clear the problem). A graphic may also fail to convert to PNG. In Mozilla a message will be displayed stating that the graphic contained errors, in Internet Explorer a small box with a cross in it will be displayed. If the graphic fails to convert either select another image, or download the SVG file and use a graphics application that supports SVG. Alternatively on the image's Open Clip Art page bring up the context menu on the

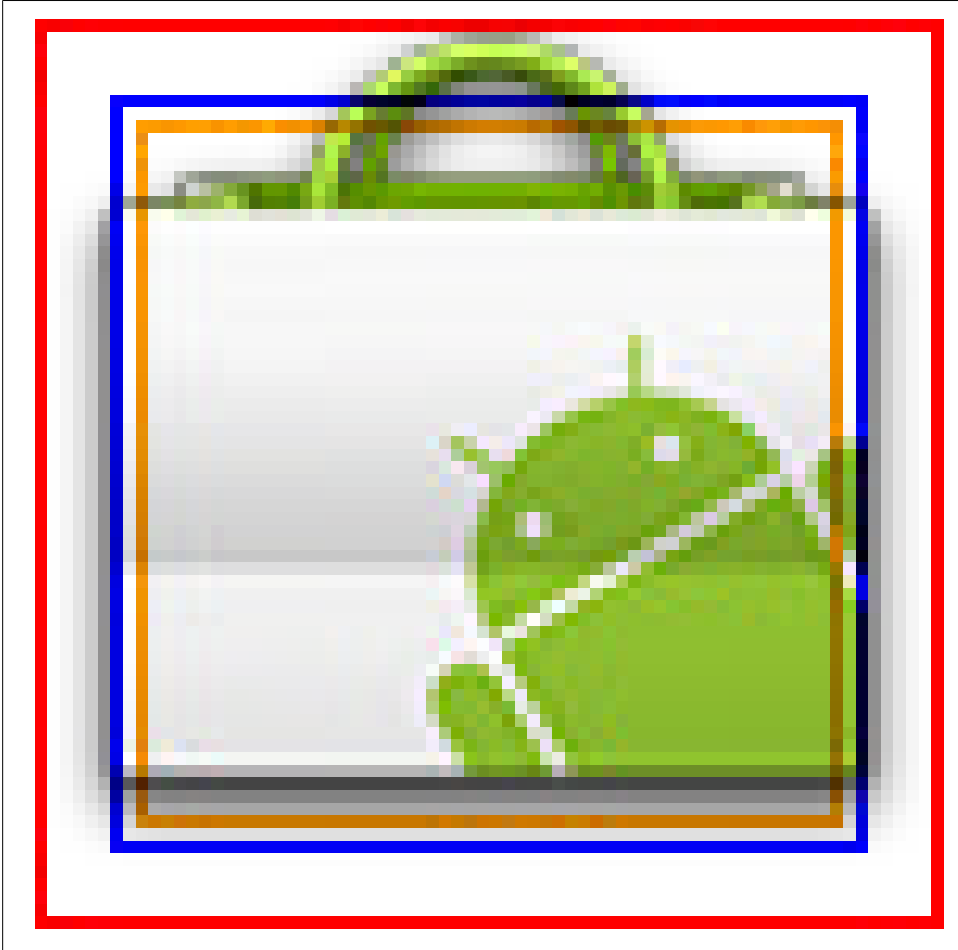


Figure 5-20.

graphic itself and save it as a full size PNG (it can then be resized in a graphics application and the transparency reset).

After using the PNG button on the selected graphic there will be four files each containing the same image at four resolutions. The graphics files may not be perfectly square, for example 39 by 40 instead of 40 by 40, but the small difference does not matter.

The files need to be resized into the correct icon size by adding the empty border. This is done in a graphics application, such as GIMP (<http://www.gimp.org>), Inkscape (<http://www.inkscape.org>) or Paint.NET (<http://www.getpaint.net> - Windows only).

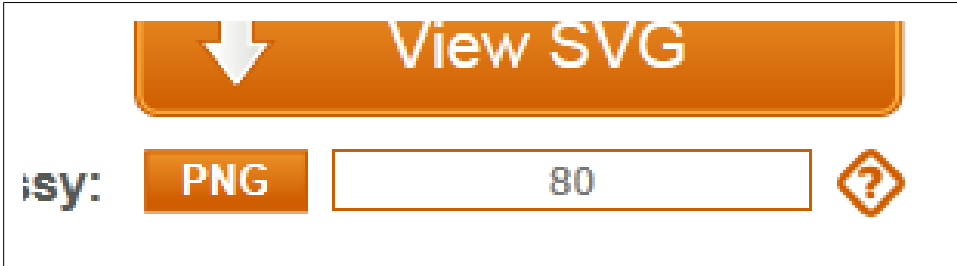


Figure 5-21.

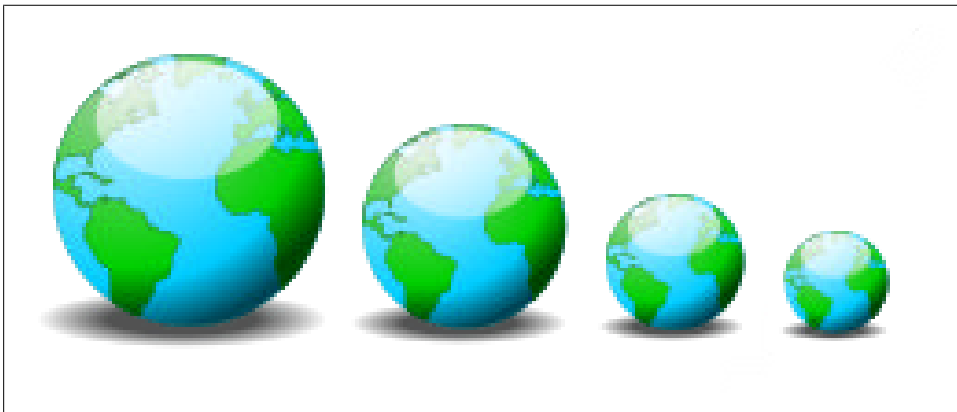


Figure 5-22.

In Paint.NET open the first graphics file. First the secondary (background) color needs to be set to transparency. This is done with the Colors dialog, using the Window menu option select Colors (or press F8). On the Colors dialog ensure that Secondary is selected in the drop down, then click the More button to see the advanced options. Set the Transparency option in the bottom right of the Colors dialog to zero.

Next open the Canvas Size dialog by using the Image menu option and selecting Canvas Size (or press Ctrl-Shift-R). Select the By absolute size radio button, ignore the Maintain aspect ratio check box, if the graphic is square it can be checked, if not it should be unchecked. In the Pixel size options set the correct Width and Height for the icon for the given graphic, both 36 for the 30 by 30 graphic, both 48 for the 40 by 40, both 72 for the 60 by 60, and both 96 for the 80 by 80. Set the Anchor option to middle. Select OK.

Save the resized image and repeat for the other three graphics to finish up with four PNG icon files at sizes 36, 48, 72 and 96.

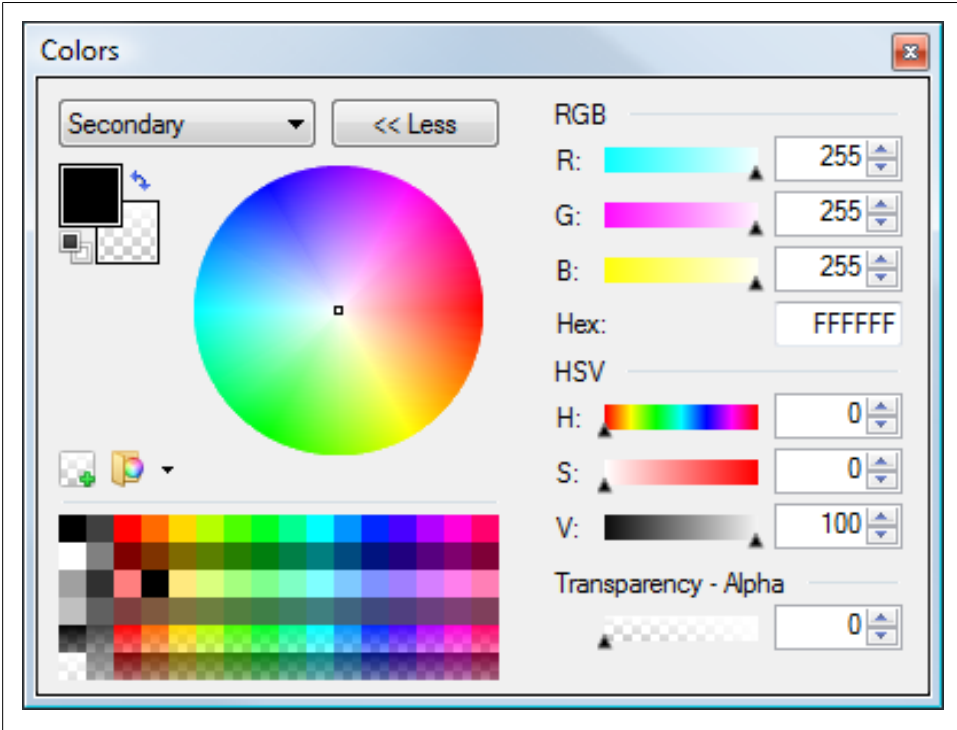


Figure 5-23.

The four files need to be copied into the project where the icons are to be used. In the project directories each icon is placed into a folder under the **res** folder for each dpi setting. If the project is in Eclipse then it is likely that under **res** there are already folders **drawable-hdpi**, **drawable-ldpi** and **drawable-mdpi**, all with the default icon.

The existing icons are replaced with the newly created ones; in the process the folder for **xhdpi** is added called **drawable-xhdpi**. If the App supports Android version 1.5 then a folder simply called **drawable** containing the 48 by 48 icon is also required. The following table is a summary:

Table 5-1. Icon Formatting Summary

Folder	Icon Size	Image Size	dpi	Android Density	Example Screen	Notes
drawable-ldpi	36x36	30x30	120	ldpi	small QVGA	
drawable-mdpi	48x48	40x40	160	mdpi	normal HVGA	default icon in absence of anything else
drawable-hdpi	72x72	60x60	240	hdpi	normal WVGA800	
drawable-xhdpi	96x96	80x80	320	xhdpi	custom	

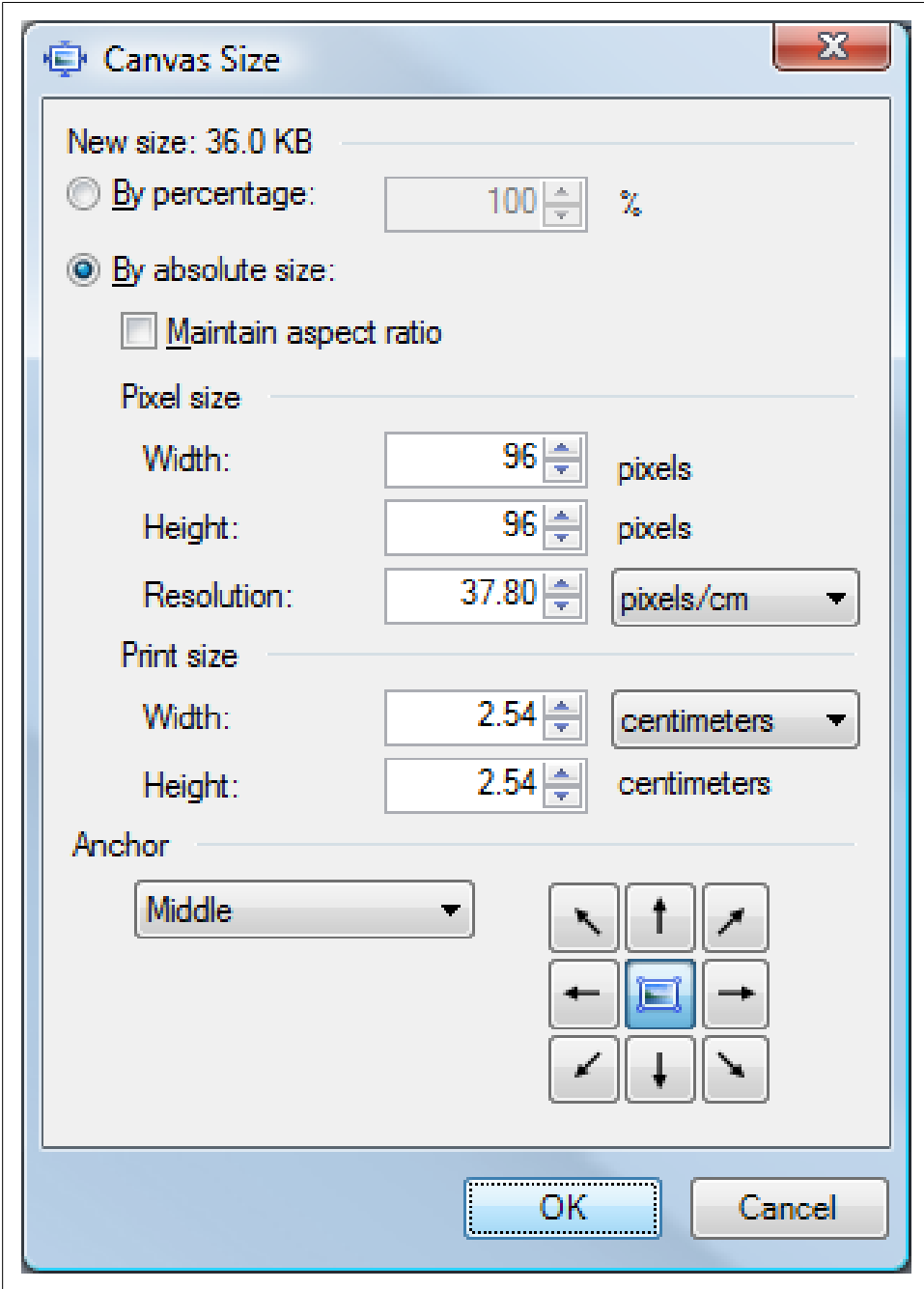


Figure 5-24.

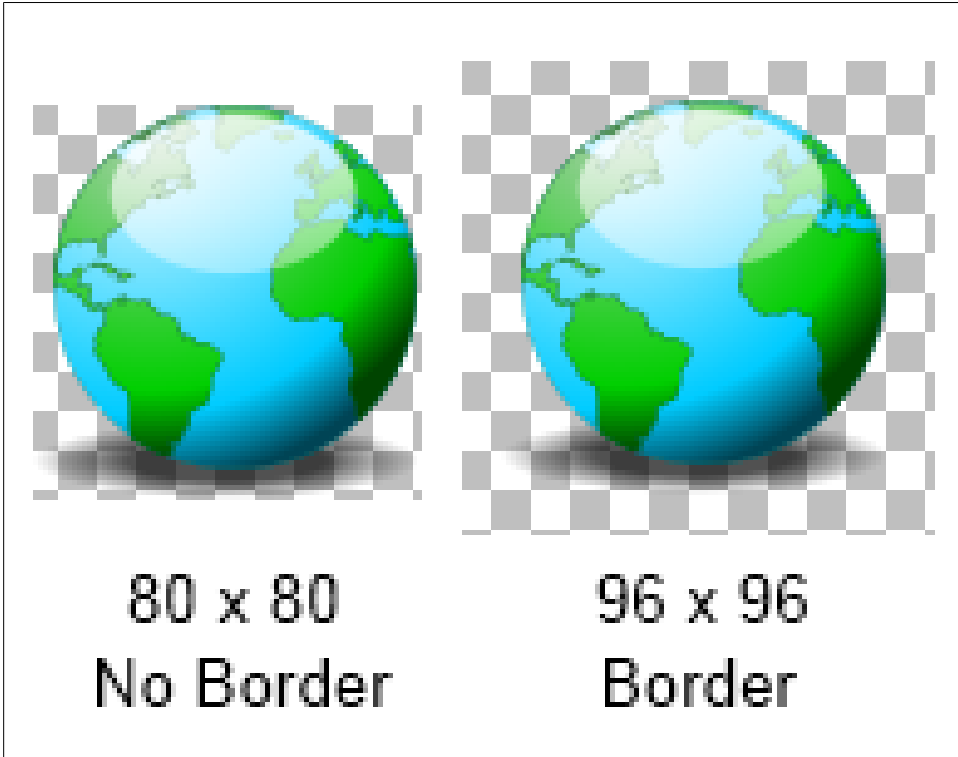


Figure 5-25.

Folder	Icon Size	Image Size	dpi	Android Density	Example Screen	Notes
drawable	48x48	40x40	160	mdpi	normal HVGA	default icon in absence of anything else

The icon file does not need to be called `icon.png`. As long as all the file names in all the 'drawable' folders are valid and the same they can be named something else. For example the icon files could be called `globe.png`. If the file name is changed then the `android:icon` attribute in the `application` element in the manifest file will also need `icon` changing to `globe`. Open the `AndroidManifest.xml` file. Locate the `application` element and change `android:icon="@drawable/icon"` to `android:icon="@drawable/globe"`.

Remember to give thanks for free stuff, in this case I thank Open Clip Art Library contributor [jhnri4](#).

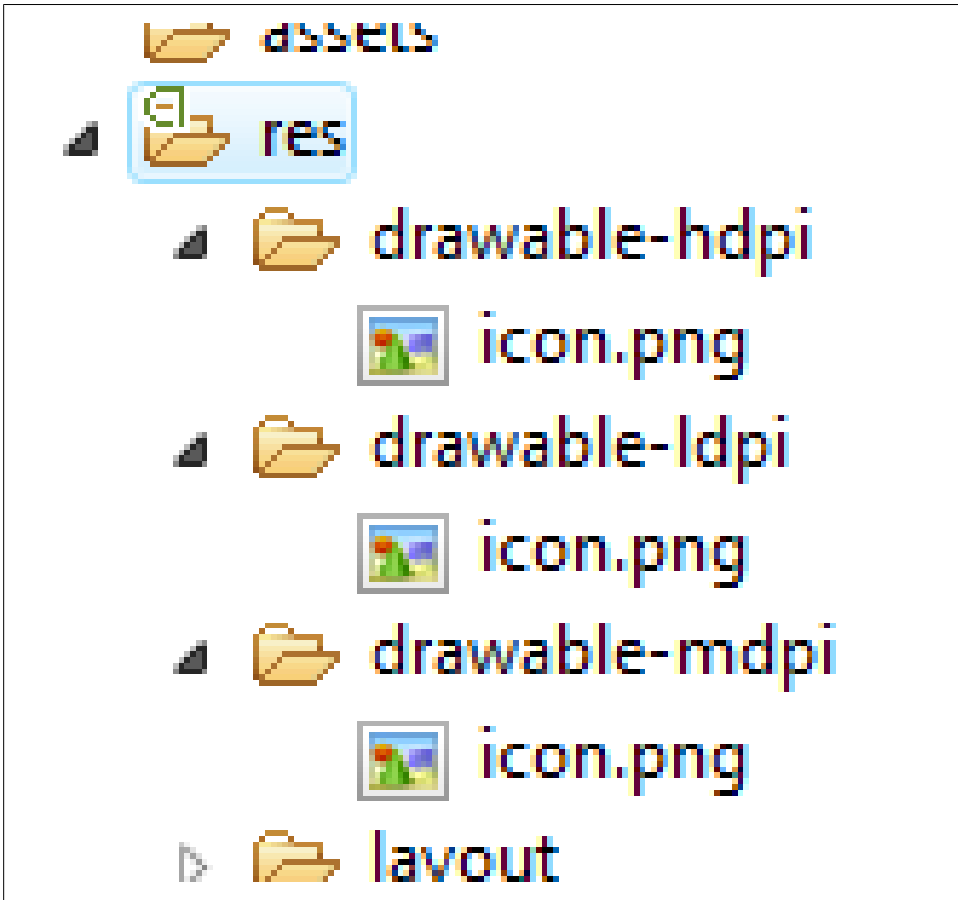


Figure 5-26.

See Also

Recipe 1.4

http://developer.android.com/guide/practices/ui_guidelines/icon_design_launcher.html

<http://www.openclipart.org>

<http://www.getpaint.net>

<http://www.inkscape.org>

<http://www.gimp.org>

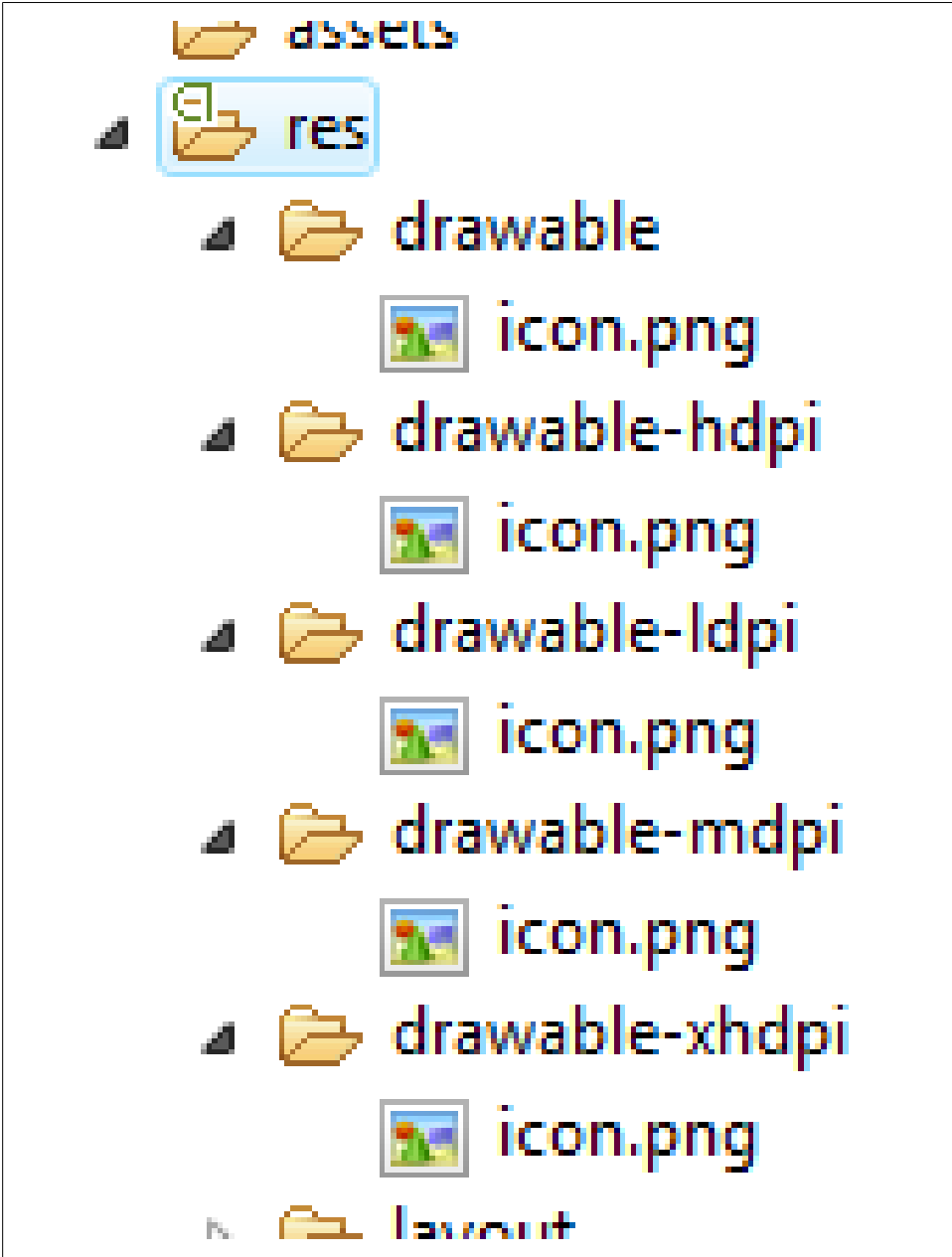


Figure 5-27.



Figure 5-28.

5.11 Android HTML5 RGraph Charting

Wagied Davids

Problem

Need to visualize data in a chart and ability to interaction with the plot/chart via JavaScript.

Solution

Example 5-23.

```
<html>
<head>
<title>RGraph: HTML5 canvas graph library - pie chart</title>
```

```

<script src="RGraph/libraries/RGraph.common.core.js" ></script>
<script src="RGraph/libraries/RGraph.common.annotate.js" ></script>
<script src="RGraph/libraries/RGraph.common.context.js" ></script>
<script src="RGraph/libraries/RGraph.common.tooltips.js" ></script>
<script src="RGraph/libraries/RGraph.common.zoom.js" ></script>
<script src="RGraph/libraries/RGraph.common.resizing.js" ></script>
<script src="RGraph/libraries/RGraph.pie.js" ></script>

<script>
  window.onload = function ()
  {
    /**
     * These are not angles - these are values. The appropriate angles are calculated
     */
    var pie1 = new RGraph.Pie('pie1', [41,37,16,3,3]); // Create the pie object
    pie1.Set('chart.labels', ['MSIE 7 (41%)', 'MSIE 6 (37%)', 'Firefox (16%)', 'Safari (3%)', 'Other (3%)']);
    pie1.Set('chart.gutter', 30);
    pie1.Set('chart.title', "Browsers (tooltips, context, zoom)");
    pie1.Set('chart.shadow', false);
    pie1.Set('chart.tooltips.effect', 'contract');
    pie1.Set('chart.tooltips', [
      'Internet Explorer 7 (41%)',
      'Internet Explorer 6 (37%)',
      'Mozilla Firefox (16%)',
      'Apple Safari (3%)',
      'Other (3%)'
    ]
    );
    pie1.Set('chart.highlight.style', '3d'); // Defaults to 3d anyway; can be 2d or 3d

    if (!RGraph.isIE8()) {
      pie1.Set('chart.zoom.hdir', 'center');
      pie1.Set('chart.zoom.vdir', 'up');
      pie1.Set('chart.labels.sticks', true);
      pie1.Set('chart.labels.sticks.color', '#aaa');
      pie1.Set('chart.contextmenu', [['Zoom in', RGraph.Zoom]]);
    }

    pie1.Set('chart.linewidth', 5);
    pie1.Set('chart.labels.sticks', true);
    pie1.Set('chart.strokestyle', 'white');
    pie1.Draw();

    var pie2 = new RGraph.Pie('pie2', [2,29,45,17,7]); // Create the pie object
    pie2.Set('chart.gutter', 45);
    pie2.Set('chart.title', "Some data (context, annotatable)");
    pie2.Set('chart.linewidth', 1);
    pie2.Set('chart.strokestyle', '#333');
    pie2.Set('chart.shadow', true);
    pie2.Set('chart.shadow.blur', 3);
    pie2.Set('chart.shadow.offsetx', 3);
    pie2.Set('chart.shadow.offsety', 3);
    pie2.Set('chart.shadow.color', 'rgba(0,0,0,0.5)');
    pie2.Set('chart.colors', ['red', 'pink', '#6f6', 'blue', 'yellow']);
    pie2.Set('chart.contextmenu', [['Clear', function () {RGraph.Clear(pie2.canvas); pie2.Draw();}]]);
  }
</script>

```

```

        pie2.Set('chart.key', ['John (2%)', 'Richard (29%)', 'Fred (45%)', 'Brian (17%)', 'Peter (7%)']);
        pie2.Set('chart.key.background', 'white');
        pie2.Set('chart.key.shadow', true);
        pie2.Set('chart.annotatable', true);
        pie2.Set('chart.align', 'left');
        pie2.Draw();
    }
</script>
</head>
<body>

    <div style="text-align: center">
        <canvas id="pie1" width="420" height="300">[No canvas support]</canvas>
        <canvas id="pie2" width="440" height="300">[No canvas support]</canvas>
    </div>

</body>
</html>

```

File: main.xml

Example 5-24.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#FFFFFF">

    <WebView
        android:id="@+id/webview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
    </WebView>
</LinearLayout>

```

File: Main.java

Example 5-25.

```

import android.app.Activity;
import android.os.Bundle;
import android.webkit.WebChromeClient;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;

public class Main extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {

```

```

super.onCreate(savedInstanceState);
setContentView(R.layout.main);

// Obtain reference to the WebView holder
WebView webview = (WebView) this.findViewById(R.id.webview);

// Get the settings
WebSettings webSettings = webview.getSettings();

// Enable Javascript for user interaction clicks
webSettings.setJavaScriptEnabled(true);

// Display Zoom Controles
webSettings.setBuiltInZoomControls(true);
webview.requestFocusFromTouch();

// Set the client
webview.setWebViewClient(new WebViewClient());
webview.setWebChromeClient(new WebChromeClient());

// Load the URL
webview.loadUrl("file:///android_asset/rgraphview.html");
}
}

```

Discussion

As an alternative to creating Android charts in pure Java, an interesting possibility exists for creating charts using the handy old WebView and HTML/Javascript approach.

Note: RGraph uses the HTML5 Canvas component, which is not accommodated in the webkit packaged with Android 1.5. RGraph works nicely and tested with Android 2.1 and 2.2.

Steps involved:

1. Create an assets directory for HTML files; Android internally maps it to file:///android_asset/ (note triple slash and singular spelling of "asset")
2. Copy rgraphview.html into it: res/assets/rgraphview.html
3. Create a javascript directory: res/assets/RGraph

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.filefactory.com/file/b3f2ffc/n/TestAndroidRGraph.zip>

5.12 Simple Raster Animation

Daniel Fowler

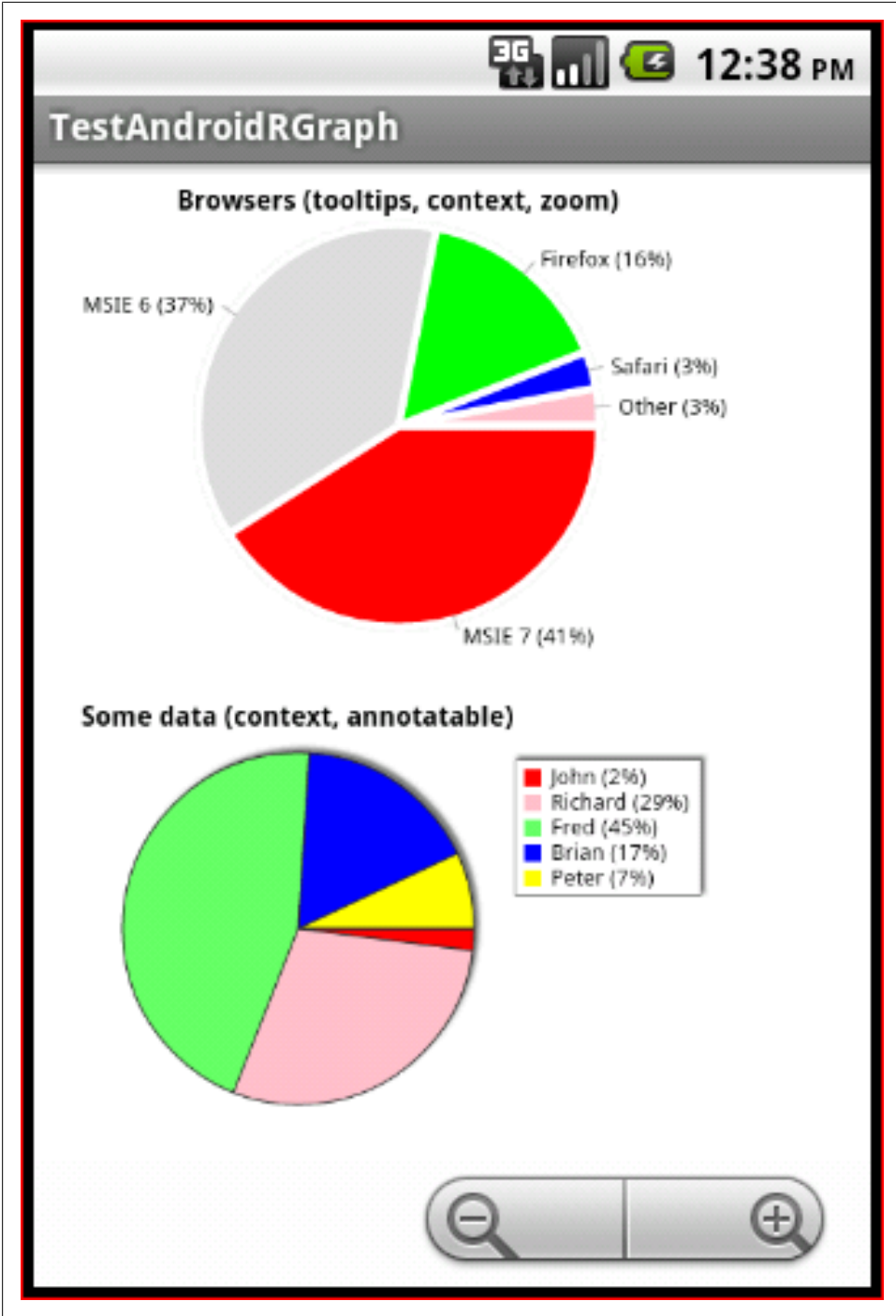


Figure 5-29.

Problem

There is a need to add an animated image to a screen.

Solution

Android has good support for user interface animation; it is easy to sequence images using the `AnimationDrawable` class.

Discussion

To achieve the animation first images to be sequenced are generated using a graphics program. Each image represents one frame of the animation, they will usually be the same size with changes between each frame as required.

This animation recipe will sequence some traffic light images. The images can be generated using the open source vector graphics program Inkscape (see <http://inkscape.org>). A copy of the image used is available from the Open Clip Art Library (<http://www.openclipart.org/>), searching for *Traffic Lights Turned Off*, select the image, click on the *View SVG* button and save the file from your browser. Open the file in Inkscape.

Four images will make up the animation, they will show the sequence of traffic lights as used in the United Kingdom, red, red and yellow, green, yellow and back to red. The SVG image has all the lights available, they are just hidden behind a translucent circle. To generate the first image select the circle covering the red light and delete it. Then from the **Edit** menu use **Select All** to highlight the whole image. Using the **File** menu select **Export Bitmap**. In the Export Bitmap dialog under **Bitmap size** enter *150* in the **Height** box, choose a directory and file name for the file to be generated, e.g. `red.png`. Click the **Export** button to export the bitmap. Delete the circle covering the yellow light, select all again and export as before to a file, e.g. `red_yellow.png`. Use the **Edit** menu and choose **Undo** (twice) to cover the red light and yellow light and then delete the circle covering the green light. Export to `green.png`. Again use undo to cover the green light and delete the circle covering the yellow light. Export the bitmap to `yellow.png`.

Four files are now ready for the animation.

Start an Android project. Copy the four generated files into the **res/drawable** directory. An **animation-list** needs to be defined in the same directory. Create a new file in **res/drawable** called *uktraffilight.xml*. In this new file add the following.

Example 5-26.

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="false">
    <item android:drawable="@drawable/red" android:duration="2000" />
    <item android:drawable="@drawable/red_yellow" android:duration="2000" />
</animation-list>
```

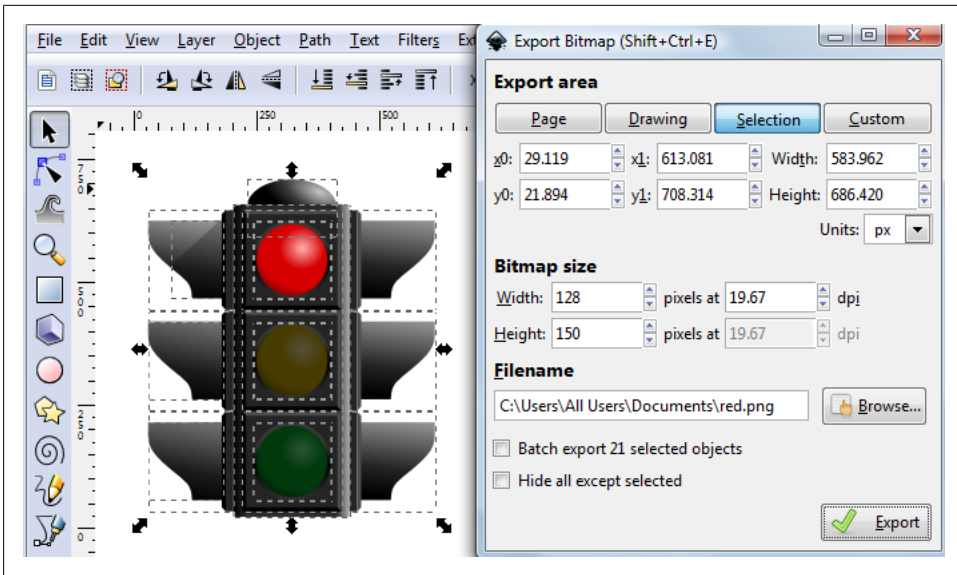


Figure 5-30.

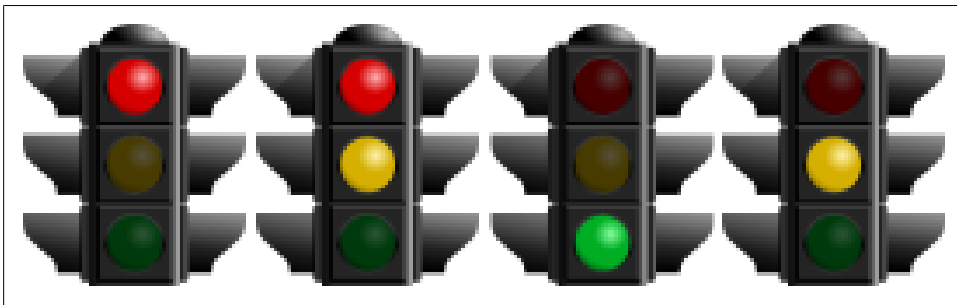


Figure 5-31.

```
<item android:drawable="@drawable/green" android:duration="2000" />
<item android:drawable="@drawable/yellow" android:duration="2000" />
</animation-list>
```

This lists the images to be animated in the order of the animation and how long each one needs to be displayed (in milliseconds). If the animation needs to stop after running through the images then the attribute **android:oneshot** is set to *true*.

In the layout file for the program a `ImageView` is added whose source is given as `@drawable/uktrafflights` (i.e. pointing to the created file).

Example 5-27.

```
&lt;?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">
        <ImageView android:id="@+id/imageView1"
            android:src="@drawable/uktrafficklites"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"/>
    </LinearLayout>

```

In the Activity class an `AnimationDrawable` (the Android class that performs the animation) is declared. In `onCreate` it is assigned to the drawable that the `ImageView` uses. Finally the animation is started by calling the `AnimationDrawable start()` method (there is a `stop()` method available to end the animation if required). The start method is called in `onWindowFocusChanged` to ensure everything has loaded before the animation starts (if could easily have been started with a Button or other type of input).

Example 5-28.

```

public class main extends Activity {
    AnimationDrawable lightsAnimation;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ImageView lights = (ImageView) findViewById(R.id.imageView1);
        lightsAnimation=(AnimationDrawable) lights.getDrawable();
    }
    @Override
    public void onWindowFocusChanged(boolean hasFocus) {
        super.onWindowFocusChanged(hasFocus);
        lightsAnimation.start();
    }
}

```

Image animations can be useful to add interest to screens and can be used in games or cartoons.

See Also

<http://inkscape.org>

<http://www.openclipart.org>



Figure 5-32.

Graphical User Interface

6.1 Introduction: GUI

Ian Darwin

Discussion

When Android was being invented, its designers faced many choices whose outcome would determine the success or failure of their project. Once they had rejected all the other smartphone operating systems, both closed and open source, and decided to build their own atop the Linux kernel, they were faced with somewhat of a blank canvas. One important choice was which user interface technology to deploy: JavaME, Swing, SWT, or none of the above.

JavaME is the Java Micro Edition, Sun/Oracle's official standard API for cell phones and other small devices. JavaME is actually a pretty big success story: tens or hundreds of millions of cell phones have a Java Micro Edition runtime inside. And every Blackberry made since around 2000, and all Blackberry applications in the world, are based on JavaME. But the JavaME GUI was regarded as too limiting, having been designed for the days when cell phones had really tiny screens.

Swing is the Java Standard Edition ("Desktop Java", JavaSE, a.k.a. JDK or JRE) GUI. It is based atop the earlier AWT. It can make some *beautiful GUI music in the right hands*, but is just too large and uses too much overhead for Android.

SWT is the GUI layer developed for use in the *Eclipse IDE* itself and in Eclipse Rich Clients. It is an abstraction layer, and depends on the underlying operating-specific toolkit (e.g., Win32 in the Microsoft arena, GTK under Unix/Linux, etc.).

The final option, and the one ultimately chosen, was to go it alone. The Android designers thus built their own Graphical User Interface toolkit designed specifically for smartphones. But they took many good ideas from the other toolkits, and learned from the mistakes that had been made along the way.

To learn any new GUI framework is, necessarily, a lot of work. One set of guidelines that can help is *the Android Patterns site*, which is not about coding but about showing designers *how* the Android visual experience is supposed to work, Illustrated, crowd-sourced, and recommended!

See Also

Designing Visual Interfaces: Communication Oriented Techniques by Muller and Sano is a thorough discussion of the design issues. The examples are from mostly desktop applications (Mac, Unix, Windows) but the principles spelt out here will be useful in dealing with human-computer interaction issues.

6.2 User Interface Guidelines (placeholder)

Ian Darwin

Problem

Lots of developers, even good ones, are very bad at user interface design.

Solution

This needs a real recipe; this is just a placeholder to save a few notes, which can be copied into the final recipe (or not...) and then I will delete this one.

Discussion

UI Guidelines have been around almost since Xerox PARC invented graphical user interfaces in the 1980's and gave them to Microsoft and Apple. A given set of guidelines must be appropriate to the platform. General guidelines for mobile devices are available from several sources. Android.com publishes advice too.

...

For some thoughtful UI pattern notes, see <http://android-developers.blogspot.com/2010/05/twitter-for-android-closer-look-at.html>.

See Also

There is an article from Research in Motion that is somewhat specific to the Blackberry platform but may be useful: see http://na.blackberry.com/eng/developers/resources/Newsletter/2010/Featured_Story_Jan_2010.jsp?html

6.3 SlidingDrawer Overlapping other UI components

Wagied Davids

Problem

You want the SlidingDrawer to overlap other UI components eg. ListView

Solution

File: main.xml

Example 6-1.

```
<?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="wrap_content">

    <ListView
        android:id="@+id/list_journal"
        android:layout_width="fill_parent"
        android:layout_height="365dip" />

    <SlidingDrawer
        android:id="@+id/slidingDrawer"
        android:handle="@+id/drawerHandle"
        android:content="@+id/contentLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <ImageView
            android:id="@+id/drawerHandle"
            android:src="@drawable/help_tab_selector"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content">
        </ImageView>

        <LinearLayout
            xmlns:android="http://schemas.android.com/apk/res/android"
            android:id="@+id/contentLayout"
            android:gravity="center"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:background="@drawable/bg">
            <ImageView
                android:src="@drawable/icon"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="center">
            </ImageView>
        </LinearLayout>
    </SlidingDrawer>
</LinearLayout>
```

File: list_item.xml

Example 6-2.

```
&lt;?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/list_item"
    android:gravity="center"
    android:textColor="@color/white"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
</TextView>
```

File: Main.java

Example 6-3.

```
import java.util.ArrayList;
import java.util.List;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.SlidingDrawer;
import android.widget.SlidingDrawer.OnDrawerCloseListener;
import android.widget.SlidingDrawer.OnDrawerOpenListener;

public class Main extends Activity implements OnDrawerOpenListener,
    OnDrawerCloseListener {

    private ListView listView;
    private SlidingDrawer slidingDrawer;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Set the View Layer
        setContentView(R.layout.main);

        List<String> data= getData();

        // Get reference to ListView
        listView = (ListView) this.findViewById(R.id.list_journal);
        ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this, R.layout.list_item, data);
        listView.setAdapter(arrayAdapter);

        // Get reference to SlidingDrawer
        slidingDrawer = (SlidingDrawer) this.findViewById(R.id.slidingDrawer);
        slidingDrawer.setOnDrawerOpenListener(this);
        slidingDrawer.setOnDrawerCloseListener(this);
    }

    /**
```

```

    * Get some data
    * @return
    */
private List<String> getData()
{
    List<String> data= new ArrayList<String>();
    for( int i= 0; i < 20; i++)
    {
        data.add( String.valueOf(i));
    }
    return data;
}

@Override
public void onDrawerOpened() {
    listView.setVisibility(ListView.GONE);
}

@Override
public void onDrawerClosed() {
    listView.setVisibility(ListView.VISIBLE);
}
}
}

```

Discussion

The default behaviour of the SlidingDrawer component is to maximize to a height of the position of the last component on the screen. But if the last component is at the very bottom, then the SlidingDrawer will not be apparently visible!

The Solution: Manipulating Androids View layer to hide/reveal components! listView - a reference to the ListView component declared in your XML-layout file. slidingDrawer - a reference to the SlidingDrawer component in our XML-layout file.

```
// Get reference to SlidingDrawer slidingDrawer = (SlidingDrawer) this.findViewById(R.id.slidingDrawer);
slidingDrawer.setOnDrawerOpenListener(this);
slidingDrawer.setOnDrawerCloseListener(this);
```

```
@Override public void onDrawerOpened() { listView.setVisibility(ListView.GONE); }
```

```
@Override public void onDrawerClosed() { listView.setVisibility(ListView.VISIBLE); }
```

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.filefactory.com/file/b49a47h/n/TestSlidingDrawerOverList.zip>

6.4 Android 3.0 Photo Gallery

Wagied Davids

Problem

Display a photo gallery

Solution

File: main.xml

Example 6-4.

```
<?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"
    android:gravity="center"
    >

    <Gallery
        android:id="@+id/gallery1"
        android:layout_height="wrap_content"
        android:layout_width="match_parent"
        android:spacing="10dip"
        >
    </Gallery>
</LinearLayout>
```

File: Main.java

Example 6-5.

```
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.Gallery;
import android.widget.Toast;

public class Main extends Activity implements OnItemClickListener
{
    private static final String tag = "Main";
    private Gallery _gallery;
    private ImageAdapter _imageAdapter;

    /** Called when the activity is first created. */
```

```

@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    setTitle("Android Honeycomb Photo Gallery Example");

    _gallery = (Gallery) this.findViewById(R.id.gallery1);
    _imageAdapter = new ImageAdapter(this);
    _gallery.setAdapter(_imageAdapter);
    _gallery.setOnItemClickListener(this);
}

@Override
public void onItemClick(AdapterView<?> arg0, View view, int position, long duration)
{
    int resourceId = (Integer) _imageAdapter.getItem(position);
    Drawable drawable = getResources().getDrawable(resourceId);
    Bitmap bitmap = BitmapFactory.decodeResource(getResources(), resourceId);

    Toast.makeText(this, "Selected Image: " + getResources().getText(resourceId) + "\nHeight: " +
}
}

```

File: ImageAdapter.java

Example 6-6.

```

public class ImageAdapter extends BaseAdapter
{
    private Context _context = null;
    private final int[] imageIds = { R.drawable.formula, R.drawable.hollywood, R.drawable.mode1, R.drawable.relation2, R.drawable.renaissance, R.drawable.renaissance_zoom };
    public ImageAdapter(Context context)
    {
        this._context = context;
    }

    @Override
    public int getCount()
    {
        return imageIds.length;
    }

    @Override
    public Object getItem(int index)
    {
        return imageIds[index];
    }

    @Override
    public long getItemId(int index)
    {
        return index;
    }
}

```

```

    }

    @Override
    public View getView(int position, View view, ViewGroup group)
    {
        ImageView imageView = new ImageView(_context);
        imageView.setImageResource(imageIds[position]);
        imageView.setScaleType(ScaleType.FIT_XY);
        imageView.setLayoutParams(new Gallery.LayoutParams(400, 400));
        return imageView;
    }
}

```

Discussion

1. Download the preview release of Android 3.0 using either the SDK download manager (preferred) or from within the Eclipse IDE using Android SDK and AVD manager.
2. Create an AVD to run the emulator
3. Create an Android project (*Important*: set the Min. SDK Version to "Honeycomb") and Click Finish!
4. Create a main entry point java file eg. Main.java
5. Create an ImageAdapter.java file
6. Create an XML layout file: main.xml
7. Package and Run the Android app.

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.filefactory.com/file/b58cf27/n/TestHoneycombGallery.zip>

6.5 Building a UI using Fragments API of Android 3.0 in Android 2.2

Saketkumar Srivastav

Problem

You want to add Fragments to the UI in Android 2.0+ versions, though they were originally only available in 3.0+. Fragments, as the name suggests, are nothing but the small chunks of UI which constitutes a single activity. It can be treated as individual portlets of a portal page. It is very similar to activity in terms of its looks, lifecycle, etc but it is also different than Activity in the sense that 'A fragment should always reside in an Activity'; fragments cannot exist independently as Activities.

Solution

The Fragments API was not supported by previous versions to Android 3.0. When Google released the compatibility package, it was possible to build applications using this Fragments API.

Discussion

To create a Fragment, we need to extend the class with Fragment. There are different kinds of Fragments available such as: ListFragment (ListActivity) DialogFragment (Dialog Interface) PreferenceFragment (PreferenceActivity)

Lets start with the FragmentTestActivity class. In onCreate() method we set the list adapter to hold a string array of magazine titles of EFY group. We also set the listener on the list items so that we can perform some action when an item from the list is clicked.

In the onItemClickListener() method we perform the main task of managing the Fragment. We obtain the instance of the fragment passing the position of the clicked item. Now we need to replace the fragment element that we have in main.xml with the new fragment TestFragment which has a meaning full UI associated with it. To accompolish this we get the instance of FragmentTransaction class, this API allows us to add, remove and replace a fragment programmatically. We replace the R.id.the_frag which corresponds to the <fragment> element of main.xml with the newly created fragment 'f' set-Transition() method signifies the kind of transition that happens with the fragment. addToBackStack() method adds the fragment transaction to back of the fragment stack so that when the back button is pressed on the device, you go to the last transaction of the fragment and not exiting the application. After all the transaction is being made we commit the transaction.

Now let us setup the fragment class, the TestFragment class. We initialize the position of the clicked item from the list to a variable magznumber. As we discussed earlier if a fragment is being associated with a UI then onCreateView() method is used to inflate the view to the fragment. Here, we create a linear layout for the fragment and then load it with the appropriate image of the magazine in an ImageView and this ImageView is added to the linear layout.

FragmentTestActivity.java

Example 6-7.

```
public class FragmentTestActivity extends FragmentActivity implements OnItemClickListener {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        ListView l = (ListView) findViewById(R.id.number_list);
        ArrayAdapter<String> magzTitles = new ArrayAdapter<String>(getApplicationContext(),
            android.R.layout.simple_list_item_1, new String[]{"Electronics For You",
                "Linux For You",
                "Facts For you"});
        /* ArrayAdapter<String> magzTitles = new ArrayAdapter<String>(getApplicationContext(),
            android.R.layout.simple_list_item_1, R.array.magz_titles);*/
    }
}
```

```

        l.setAdapter(magzTitles);
        l.setOnItemClickListener(this);
    }

    /**
     * Called when a number gets clicked
     */
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

        Fragment f = new TestFragment(position+1);

        FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
        ft.replace(R.id.the_frag, f);
        ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
        ft.addToBackStack(null);
        ft.commit();
    }
}

```

TestFragment.java

Example 6-8.

```

public class TestFragment extends Fragment {

    private int magznumber;

    public TestFragment() {

    }

    /**
     * Constructor for being created explicitly
     */
    public TestFragment(int position) {
        this.magznumber = position;
    }

    /**
     * If we are being created with saved state, restore our state
     */
    @Override
    public void onCreate(Bundle saved) {
        super.onCreate(saved);
        if (null != saved) {
            magznumber = saved.getInt("magznumber");
        }
    }

    /**
     * Save the number of Androids to be displayed
     */
    @Override
    public void onSaveInstanceState(Bundle toSave) {
        toSave.putInt("magznumber", magznumber);
    }
}

```

```

}

/**
 * Make a grid to view the magazines
 */
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

    Context c = getActivity().getApplicationContext();

    LinearLayout l = new LinearLayout(c);
    LayoutParams params = new LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.MATCH_PARENT, 0);

    l.setLayoutParams(params);

    ImageView i = new ImageView(c);

    switch(magznumber){
    case 1:
        i.setImageResource(R.drawable.efymag);
        break;
    case 2:
        i.setImageResource(R.drawable.lfymag);
        break;
    case 3:
        i.setImageResource(R.drawable.ffymag);
        break;
    }

    l.addView(i);

    return l;
}
}

```

See Also

<http://developer.android.com/guide/topics/fundamentals/fragments.html>

Source Download URL

The source code for this example may be downloaded from this URL: <https://github.com/SaketSrivastav/AndroidFragmentDemo>

Binary Download URL

The executable code for this example may be downloaded from this URL: <https://sites.google.com/site/iamsaketsrivastav/publications/FragmentTest.apk?attredirects=0&d=1>



Figure 6-1.

6.6 Haptic Feedback

Adrian Cowham

Problem

Building user confidence that their actions had an effect is a requirement for any app on any platform. The canonical example is displaying a progress bar to let users know their action took effect and it's being processed. For touch interfaces this technique still applies, but the advantage of a touch interface is that developers have the opportunity to provide physical feedback, users are capable of actually feeling the device react to their actions.

Solution

I've played with many apps on Android phones and tablets, and the thing I appreciate most is knowing that touching the screen had an effect. I like to know immediately that the app recognized and is reacting to my touch. This reaction comes in three forms, visual, audio, or physical. This recipe discusses how to increase use confidence in your app by providing instant physical feedback through the use of Android's haptic controls.

Discussion

Android has some stock haptic controls, but if these don't satisfy your needs you can gain control of the device's vibrator for custom feedback.

Custom control of the device's vibrator requires permission, this is something you'll have to explicitly list in your `AndroidManifest.xml` (example below). If you're paranoid about asking for permission or if you already have a long list of permissions, you may want to use the stock Android haptic feedback options.

Please note, the Motorola Xoom doesn't have a vibrator, therefore the examples below will compile and run, but you will not receive haptic feedback.

I'll start by showing the more complicated example first, custom haptic feedback.

Custom haptic feedback using the device's vibrator

1.) First Step, request the permission. Add the following line to you `AndroidManifest.xml`.

Example 6-9.

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

2.) Now define a listener to respond to touch events. It's not shown here but the `CustomHapticListener` class is actually a private non-static inner class of my `Activity`. This is because it needs access to the `Context.getSystemService(...)` method.

Example 6-10.

```
private class CustomHapticListener implements OnTouchListener {

    // Duration in milliseconds to vibrate
    private final int durationMs;

    public CustomHapticListener( int ms ) {
        durationMs = ms;
    }

    @Override
    public boolean onTouch( View v, MotionEvent event ) {
        if( event.getAction() == MotionEvent.ACTION_DOWN ){
            Vibrator vibe = ( Vibrator ) getSystemService( VIBRATOR_SERVICE );
```

```

        vibe.vibrate( durationMs );
    }
    return true;
}
}

```

Lines 13 and 14 are the important ones. Line 13 gets a reference to the Vibrator service and line 14 vibrates the device. If you have requested the vibrate permission, line 14 will throw an exception.

3.) Lastly, register the listener. In your Activity's onCreate(...) method. You'll need to get a reference to the GUI element you want to attach haptic feedback to and then register the OnTouchListener we defined above.

Example 6-11.

```

@Override
public void onCreate( Bundle savedInstanceState ) {
    Button customBtn = ( Button ) findViewById( R.id.btn_custom );
    customBtn.setOnTouchListener( new CustomHapticListener( 100 ) );
}

```

That's it you're now in control of the haptic feedback, now onto using stock Android haptic feedback.

Stock Haptic Feedback Events

First things first: to use stock Android haptic feedback events you must enable this on View-by-View basis. That is, you must explicitly enable haptic feedback for each View. Enabling haptic feedback can be done declaratively in your layout file or programmatically in Java. To enable haptic feedback in your layout, simply add the **android:hapticFeedbackEnabled="true"** attribute to your View(s). Here's an abbreviated example:

Example 6-12.

```

<button android:hapticFeedbackEnabled="true">
</button>

```

Here's how you do the same thing in code:

Example 6-13.

```

Button keyboardTapBtn = ( Button ) findViewById( btnId );
keyboardTapBtn.setHapticFeedbackEnabled( true );

```

Now that haptic feedback has been enabled, the next step is to register an OnTouchListener and then perform the actual feedback. Below is an example of registering an OnTouchListener and performing haptic feedback when a user touches the View.

Example 6-14.

```
// Initialize some buttons with the stock Android haptic feedback values
private void initializeButtons() {
    // initialize the buttons with the standard Haptic feedback options
    initializeButton( R.id.btn_keyboardTap, HapticFeedbackConstants.KEYBOARD_TAP );
    initializeButton( R.id.btn_longPress,   HapticFeedbackConstants.LONG_PRESS );
    initializeButton( R.id.btn_virtualKey,  HapticFeedbackConstants.VIRTUAL_KEY );
}

// helper method to initialize single buttons and register an onTouchListener
// to perform the haptic feedback
private void initializeButton( int btnId, int hapticId ) {
    Button btn = ( Button ) findViewById( btnId );
    btn.setOnTouchListener( new HapticTouchListener( hapticId ) );
}

// Class to handle touch events and respond with haptic feedback
private class HapticTouchListener implements onTouchListener {

    private final int feedbackType;

    public HapticTouchListener( int type ) { feedbackType = type; }

    public int feedbackType() { return feedbackType; }

    @Override
    public boolean onTouch( View v, MotionEvent event ) {
        // only perform feedback when the user touches the view, as opposed
        // to lifting a finger off the view
        if( event.getAction() == MotionEvent.ACTION_DOWN ){
            // perform the feedback
            v.performHapticFeedback( feedbackType() );
        }
        return true;
    }
}
```

You'll notice on lines 3 - 5 I'm initializing 3 different buttons with three different haptic feedback constants. These are Android's stock values, 2 of the 3 seem to provide exactly the same feedback. The example code above is part of a test app I wrote to demonstrate haptic feedback and I could not tell the difference between `HapticFeedbackConstants.LONG_PRESS` and `HapticFeedbackConstants.KeyboardTap`. Also, `HapticFeedbackConstants.VIRTUAL_KEY` do not appear to provide any feedback when tested.

Line 31 is where the haptic feedback is performed. All in all, providing haptic feedback is pretty simple, if you want control of the device's vibrator make sure you request permission in your `AndroidManifest.xml`. If you choose to use the stock Android haptic feedback options, make sure you enable haptic feedback for your Views either in the layout or programmatically.

See Also

<http://mytensions.blogspot.com/2011/03/androids-haptic-feedback.html>

Source Download URL

The source code for this example may be downloaded from this URL: <https://docs.google.com/leaf?id=0BwH86cQEzwiZZjZiMThmM2EtZDk3Zi00NTViLTk0NjYtNDU2YzI5MjVmMzYw&hl=en&authkey=CJu58JcL>

6.7 Handling Configuration Changes by Decoupling View from Model

Alex Leffelman

Problem

When your device's configuration changes (most frequently due to an orientation change), your Activity is destroyed and recreated, making state information difficult to maintain.

Solution

Decouple your user interface from your data model so that the destruction of your Activity doesn't affect your state data.

Discussion

It's a situation that every Android developer runs into with their very first application: "My application works great, but when I change my phone's orientation everything resets!"

By design, when a device's configuration (read: orientation) changes, the Android UI Framework destroys the current **Activity** and recreates it for the new configuration. This enables the designer to optimize the layout for different screen orientations and sizes. However, this causes a problem for the developer who wishes to maintain the state of the **Activity** as it was before the orientation change destroyed the screen. Attempting to solve this problem can lead to many complicated solutions, some more graceful than others. But if we take a step back and design our application wisely, we can write cleaner, more robust code that makes life easier for everyone.

A Graphical User Interface (GUI) is exactly what its name describes. It is a graphical representation of an underlying data model that allows the user to interface with and manipulate the data. It is NOT the data model itself. Let's talk our way through an example to illustrate why that is an important point to make.

Consider a Tic-Tac-Toe application. A simple main `Activity` for this would most likely include *at bare minimum* a `GridView` (with appropriate `Adapter`) to display the board and a `TextView` to tell the user whose turn it is. When the user clicks a square in the grid, an appropriate X or O is placed in that grid cell. As a new Android developer, we find it logical to also include a 2-dimensional array containing a representation of the board to store its data so that we can determine if the game is over, and if so, who won.

Example 6-15.

```
public class TicTacToeActivity extends Activity {

    private TicTacToeState[][] mBoardState;

    private GridView mBoard;
    private TextView mTurnText;

    @Override
    public void onCreate(Bundle savedInstanceState) {

        setContentView(R.layout.main);

        mBoardState = new TicTacToeState[3][3];

        mBoard = (GridView)findViewById(R.id.board);
        mTurnText = (TextView)findViewById(R.id.turn_text);

        // ... Set up Adapter, OnClickListener, etc, for mBoard.
    }
}
```

This is easy enough to imagine and implement, and everything works great. Except that when you turn your phone sideways in the middle of an intense round of Tic-Tac-Toe, you have a fresh board staring you in the face and your inevitable victory is postponed. As described earlier, the UI Framework just destroyed your `Activity` and recreated it, calling `onCreate()` and resetting the board data.

While reading the above code, you might have said to yourself, "Hey, that '`Bundle savedInstanceState`' looks promising!" And you'd be right. For this painfully, almost criminally simple example, you could stick your board data into a `Bundle` and use it to reload your screen. There's even a pair of methods, `onRetainNonConfigurationInstance()` and `getLastNonConfigurationInstance()`, that let you pass any `Object` you want from your old, destroyed `Activity`, to your newly created one. For this example you could just pass your `mBoardState` array to your new `Activity` and you'd be all set. But we're going to write big, successful, amazing apps any day now, and that just doesn't scale well with complicated interfaces. We can do better!

This is why separating your GUI from your data model is so handy. Your GUI can be destroyed, recreated, and changed, but the underlying data can survive unharmed through as many UI changes as you can throw at it. Let's separate our game state out into a separate data class.

Example 6-16.

```
public class TicTacToeGame {  
  
    private TicTacToeState[][] mBoardState;  
  
    public TicTacToeGame() {  
        mBoardState = new TicTactoeState[3][3];  
        // ... Initialize  
    }  
  
    public TicTacToeState getCellState(int row, int col) {  
        return mBoardState[row][col];  
    }  
    public void setCellState(int row, int col, TicTacToeState state) {  
        mBoardState[row][col] = state;  
    }  
  
    // ... Other utility methods to determine whose turn it is, if the game is over, etc.  
}
```

This will not only help us maintain our application state, it's generally just good Object Oriented Design.

Now that we have our data safely outside of the volatile `Activity`, how do we access it to build our interface? There are two common approaches: 1) Declare all variables in `TicTacToeGame` as `static`, and access them through static methods. 2) Design `TicTacToeGame` as a Singleton, allowing access to one global instance to be used throughout our application.

I prefer the second option purely from a design preference perspective. We can turn `TicTacToeGame` into a Singleton by making the constructor `private` and adding the following lines to the top of the class:

Example 6-17.

```
private static TicTacToeGame instance = new TicTacToeGame();  
public static TicTacToeGame getInstance() {  
    return instance;  
};
```

Now all we have to do is obtain the game data, and set our UI elements to appropriately display the data. It's most useful to wrap this in its own function - `refreshUI()`, perhaps - so that it can be used whenever your `Activity` makes a change to the data. For example, when a user clicks a cell of the board, there need only be two lines of code in the listener: one call to modify the data model (via our `TicTacToeGame` singleton), and one call to refresh the UI.

It may be obvious, but it is worth mentioning that your data classes survive only as long as your application's process is running. If it is killed by the user or the system, naturally the data is lost. That situation necessitates more persistent storage through the file system or databases and is outside the scope of this recipe.

This approach very effectively decouples your visual representation of the data from the data itself, and makes orientation changes trivial. Simply calling `refreshUI()` in your `onCreate(Bundle)` method is enough to ensure that whenever your `Activity` is destroyed and recreated, it can access the data model and display itself correctly. And as an added bonus, you're now practicing better Object Oriented Design and will see your code base become cleaner, more scalable, and easier to maintain.

6.8 Let Them See Stars: Using RatingBar

Ian Darwin

Problem

You want the user to choose from a number of identical GUI elements in a group to indicate a value such as a "rating" or "evaluation"

Solution

Use the `RatingBar` widget; it lets you specify the number of stars to appear, the default rating, be notified when the user changes the value, and retrieve the rating.

Discussion

`RatingBar` provides the newly-familiar "Rating" user interface experience, where a user is asked to rank or rate something by a number of stars (the `RatingBar` doesn't display the thing to be rated; that's up to the rest of your app). `RatingBar` is a subclass of `ProgressBar`, extended to display a whole number of icons ("the star") in the bar. Its primary properties are:

- `numStars` - the number of stars to display (int)
- `rating` - the user's chosen rating (float, because of `stepSize`)
- `stepSize` - the increment for selection (float, common values are 1.0 and 0.5, depending on how fine-grained you want the rating to be).
- `isIndicator` - boolean, set to true to make this read-only

These are normally set in the XML:

Example 6-18.

```
<RatingBar
    android:id="@+id/serviceBar"
    android:gravity="center"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:numStars="5"
    android:rating="3"
    android:stepSize="1.0"
```



```

    android:isIndicator='false'
/>

```

The RatingBar maintains its rating value internally. You can find out how the user has rated the item in two ways:

- invoke the `getRating()` method, or
- provide a change notification listener of type `OnRatingBarChangeListener`.

The `OnRatingBarChangeListener` has a single method, `onRatingChanged`, called with three arguments:

- `RatingBar rBar` - the event source, a reference to the particular `RatingBar`;
- `float fRating` - the rating that was set;
- `boolean fromUser` - true if set by a user, false if set programmatically

The example program simulates a Customer Survey; it creates two `RatingBars`, one to rate Service and another to rate Price (the XML for both is identical except for the `android:id`). In the main program, an `OnRatingBarChangeListener` is created, to display touchy-feely--sounding feedback for the given rating (the rating is converted to int and a switch statement is used to generate a message for Toast).

Example 6-19.

```

public class Main extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        OnRatingBarChangeListener barChangeListener = new OnRatingBarChangeListener() {
            @Override
            public void onRatingChanged(RatingBar rBar, float fRating, boolean fromUser) {
                int rating = (int) fRating;
                String message = null;
                switch(rating) {
                    case 1: message = "Sorry you're really upset with us"; break;
                    case 2: message = "Sorry you're not happy"; break;
                    case 3: message = "Good enough is not good enough"; break;
                    case 4: message = "Thanks, we're glad you liked it."; break;
                    case 5: message = "Awesome - thanks!"; break;
                }
                Toast.makeText(Main.this,
                    message,
                    Toast.LENGTH_LONG).show();
            }
        };
        final RatingBar sBar = (RatingBar) findViewById(R.id.serviceBar);
        sBar.setOnRatingBarChangeListener(barChangeListener);
        final RatingBar pBar = (RatingBar) findViewById(R.id.priceBar);
        pBar.setOnRatingBarChangeListener(barChangeListener);
    }
}

```

```

Button doneButton = (Button) findViewById(R.id.doneButton);
doneButton.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View arg0) {
        String message = String.format(
            "Final Answer: Price %.0f/%d, Service %.0f/%d\nThank you!",
            sBar.getRating(), sBar.getNumStars(),
            pBar.getRating(), pBar.getNumStars()
        );
        // Thank the user
        Toast.makeText(Main.this,
            message,
            Toast.LENGTH_LONG).show();
        // And upload the numbers to a database, hopefully...

        // That's all for this Activity, hence this App.
        finish();
    }
});
}
}

```

Since there is more than one `RatingBar`, we don't save the value in the listener, since an incomplete survey is not useful in our scenario; in the Done Button action listener, we fetch both values, display them, and this would be the place to save them. Your mileage may vary: it may make more sense to save them in the `OnRatingBarChangeListener`.

If you're not used to `printf`-like formatting, the `String.format` call uses `%.0f` to format the float as an int, instead of casting it (since we have to do nice formatting anyway). Ideally the format message should be from the XML strings, but it's only a demo program.

The main UI looks like this: displaying a feedback rating:

.

When the user clicks the Done button, they will see the Farewell message displayed on the desktop window:

XXX TODO - discuss behavior with fractional increment!

When you wish both to display the current "average" or similar measure ratings from a community **and** allow the user to enter their own rating, it is customary to display the current ratings read-only, and to create a pop-up dialog to enter their particular rating. This is described at the [Android Patterns Site](#).

See Also

[RatingBar in the 'Form Stuff' tutorial on Android.com](#)

[An MVC tutorial that also shows how to construct your own RatingBar-like View component.](#)

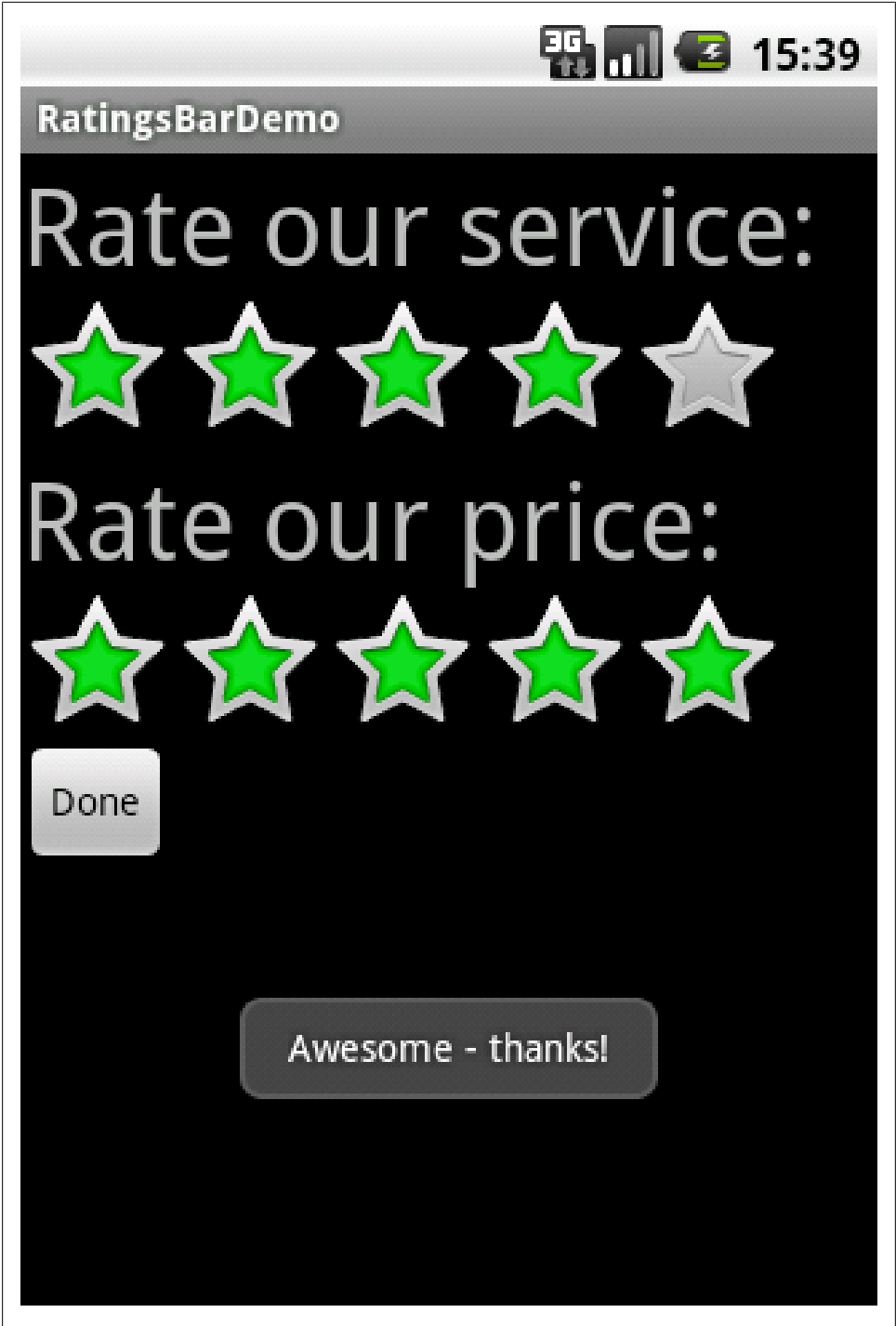


Figure 6-2.

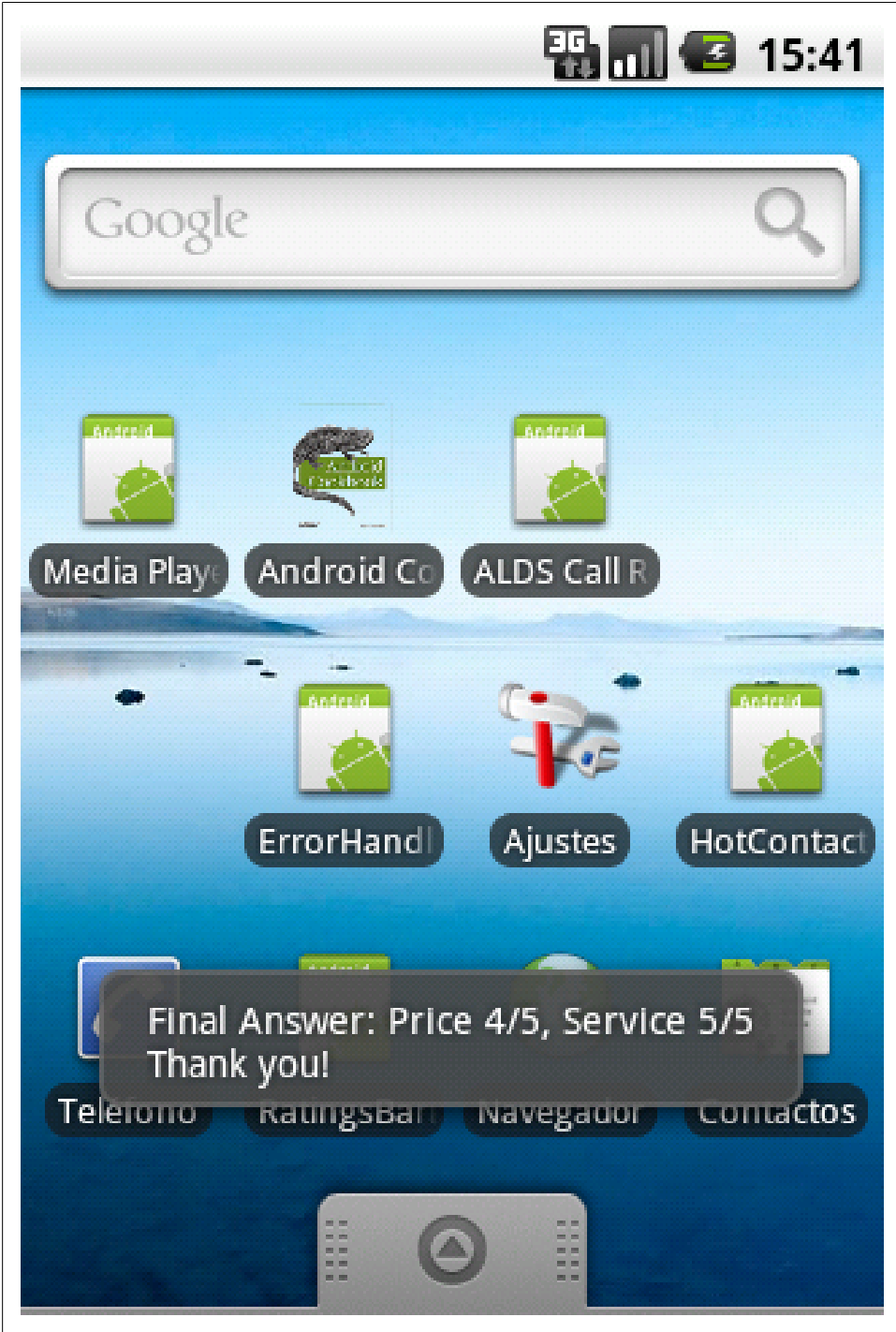


Figure 6-3.

6.9 Invoke an action handler when a Button is pressed

Ian Darwin

Problem

You need to do something when the user presses a Button.

Solution

Create a Button in your layout. In `onCreate()`, find it by ViewID. Call its `setOnClickListener()`. In the `OnClickListener` implementation, check for the ViewID and perform the relevant action.

Discussion

Creating a Button in your layout is simple. Assuming XML layout:

Example 6-20.

```
<Button android:id="@+id/start_button"
        android:text="@string/start_button_label"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```

In your activity's `onCreate()`, find the button by its ViewIDm (in this example, `R.id.start_button`). Call its `setOnClickListener()` method with an `OnClickListener`.

In the `OnClickListener` implementation, check for the ViewID and perform the relevant action.

Example 6-21.

```
public class Main extends Activity implements OnClickListener {
    public void onCreate() {
        startButton = findViewById(R.id.start_button);
        startButton.setOnClickListener(this);
        ...
    }

    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.start_button:
                // Start whatever it is the start button starts...
                ...
            case R.id.some_other_button:
                // etc
        }
    }
}
```

Any experienced Java programmer would expect to use an anonymous inner class for the `onClickListener`, as has been done in AWT and Swing since Java 1.1. Due to efficiency, early Android documentation recommended against this, simply having the Activity implement `OnClickListener` and checking the ViewID (i.e., the Java 1.0 way of doing things). As with Swing, however, the power of devices has gotten much faster, and such old-style ways of doing things are becoming less popular, though you will still see both styles in use for some time.

6.10 Creating an Alert Dialog.

Rachee Singh

Problem

The programmer wishes to prompt the user of certain unsaved changes in the application through an alert sending mechanism.

Solution

Use of Alert Dialog in Android. It permits giving suitable options to the user, in case of unsaved changes scenario the options would be:

1. Save
2. Discard Changes
3. Cancel.

Discussion

Through the `AlertDialog` class, the user can be provided with 2 options that can be used in any scenario:

1. Positive Reaction
2. Neutral Reaction
3. Negative Reaction. If the user has entered some data in an `EditText` and is then attempting the cancel that activity, the application should prompt the user to either Save his changes, Discard them or cancel the alert dialog itself.

Here is the code that would implement this kind of an `AlertDialog` along with appropriate click listeners on each button on the dialog.

Example 6-22.

```
AlertDialog = new AlertDialog.Builder(this)
.setTitle(R.string.unsaved)
.setMessage(R.string.unsaved_changes_message)
.setPositiveButton(R.string.save_changes, new AlertDialog.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
```

```

        saveInformation();
    }
})
.setNeutralButton(R.string.discard_changes, new AlertDialog.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        finish();
    }
})
.setNegativeButton(android.R.string.cancel_dialog, new AlertDialog.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        alertDialog.cancel();
    }
})
.create();
alertDialog.show();

```

6.11 Customize the SlidingDrawer component to animate/transition from the top down.

Wagied Davids

Problem

You want to customize the SlidingDrawer component to animate/transition from the top down.

Solution

Android's SlidingDrawer component's default behavior is to transition from the bottom and then move upwards on a user clicking the panel handle. To accommodate the transition from the top moving downwards, an animation is required. Use is made of the open-source org.panel package to accomplish this task.

Discussion

Steps:

- Include the org.panel easing interpolator package (source included in the given download link)
- Include as a new namespace such as panel in your Android view XML, e.g., `xmlns:panel="http://schemas.android.com/apk/res/org.panel"`
- Use the tag set instead of the Android SlidingDrawer component!

File: main.xml

Example 6-23.

```

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout

```

```

xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:panel="http://schemas.android.com/apk/res/org.panel"
android:layout_width="fill_parent"
android:layout_height="fill_parent">

<LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <org.panel.Panel
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/topPanel"
        android:paddingBottom="20dip"
        panel:position="top"
        panel:animationDuration="1000"
        panel:linearFlying="true"
        panel:openedHandle="@drawable/top_switcher_expanded_background"
        panel:closedHandle="@drawable/top_switcher_collapsed_background">
        <Button
            android:id="@id/panelHandle"
            android:layout_width="fill_parent"
            android:layout_height="33dip" />
        <LinearLayout
            android:id="@id/panelContent"
            android:orientation="vertical"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content">
            <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:text="From the Top -> Down"
                android:textSize="16dip"
                android:padding="4dip"
                android:textStyle="bold" />

            <ImageView
                android:src="@drawable/android_skateboard"
                android:layout_gravity="center"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />

        </LinearLayout>
    </org.panel.Panel>

</LinearLayout>
</FrameLayout>

```

File: Test.java

Example 6-24.

```

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

```



```

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

```

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.filefactory.com/file/b48823e/n/Android-SlidingDrawer-TopDown.zip>

6.12 Use a Timepicker widget

Pratik Rupwal

Problem

Sometimes we need to ask the user to enter the time for processing some element in the application. Accepting time in Text boxes is not graceful as well as requires validation.

Solution

Timepicker widget can be used for accepting time from the user. It makes the appearance graceful and reduces the requirement of validation.

Discussion

Below code explains how to show the current time on the screen and gives a button, clicking on which produces the timepicker widget to the user for accepting the time.

Example 6-25.

```

public class Main extends Activity
{

    private TextView mTimeDisplay;
    private Button mPickTime;

    private int mHour;
    private int mMinute;

    static final int TIME_DIALOG_ID = 0;

    /** Called when the activity is first created. */

```

```

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

    // capture our View elements
    mTimeDisplay = (TextView) findViewById(R.id.timeDisplay);
    mPickTime = (Button) findViewById(R.id.pickTime);

    // add a click listener to the button
    mPickTime.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            showDialog(TIME_DIALOG_ID);
        }
    });

    // get the current time
    final Calendar c = Calendar.getInstance();
    mHour = c.get(Calendar.HOUR_OF_DAY);
    mMinute = c.get(Calendar.MINUTE);

    // display the current date
    updateDisplay();
}

// The below overridden method gets invoked when 'showDialog()' is called inside 'onClick()' method defi
// for handling the click event of button 'change the time'

@Override
protected Dialog onCreateDialog(int id) {
    switch (id) {
        case TIME_DIALOG_ID:
            return new TimePickerDialog(this,
                mTimeSetListener, mHour, mMinute, false);
    }
    return null;
}

// updates the time we display in the TextView
private void updateDisplay() {
    mTimeDisplay.setText(
        new StringBuilder()
            .append(pad(mHour)).append(":")
            .append(pad(mMinute)));
}

// the callback received when the user "sets" the time in the dialog
private TimePickerDialog.OnTimeSetListener mTimeSetListener =
    new TimePickerDialog.OnTimeSetListener() {
        public void onTimeSet(android.widget.TimePicker view, int hourOfDay, int minute) {
            mHour = hourOfDay;
            mMinute = minute;
            updateDisplay();
        }
    }
}

```

```

};

private static String pad(int c)
{
    if (c >= 10)
        return String.valueOf(c);
    else
        return "0" + String.valueOf(c);
}
}

```

The below screenshot shows the timepicker appearing on the screen after clicking on 'change the time' button.

6.13 Formatting with Correct Plurals

Ian Darwin

Problem

You're displaying something like "Found "+ n + " items", but in English, "Found 1 reviews" is ungrammatical. You want "Found 1 review" for the case n == 1.

Solution

For simple, English-only results, use a conditional statement. For better results, that can be internationalized, use a `ChoiceFormat`. On Android, you can use `<plural>` in an XML Resources file.

Discussion

The "quick and dirty" answer is to use Java's ternary operator (`cond ? trueval : falseval`) in a string concatenation. Since in English, for most nouns, both zero and plurals get an 's' appended to the noun in English ("no books, one book, two books"), we need only test for `n==1`.

Example 6-26.

```

// FormatPlurals.java
public static void main(String argv[]) {
    report(0);
    report(1);
    report(2);
}
/** report -- using conditional operator */
public static void report(int n) {
    System.out.println("Found " + n + " item" + (n==1?"":"s"));
}

```

Running this on JavaSE as a main program shows the following output:

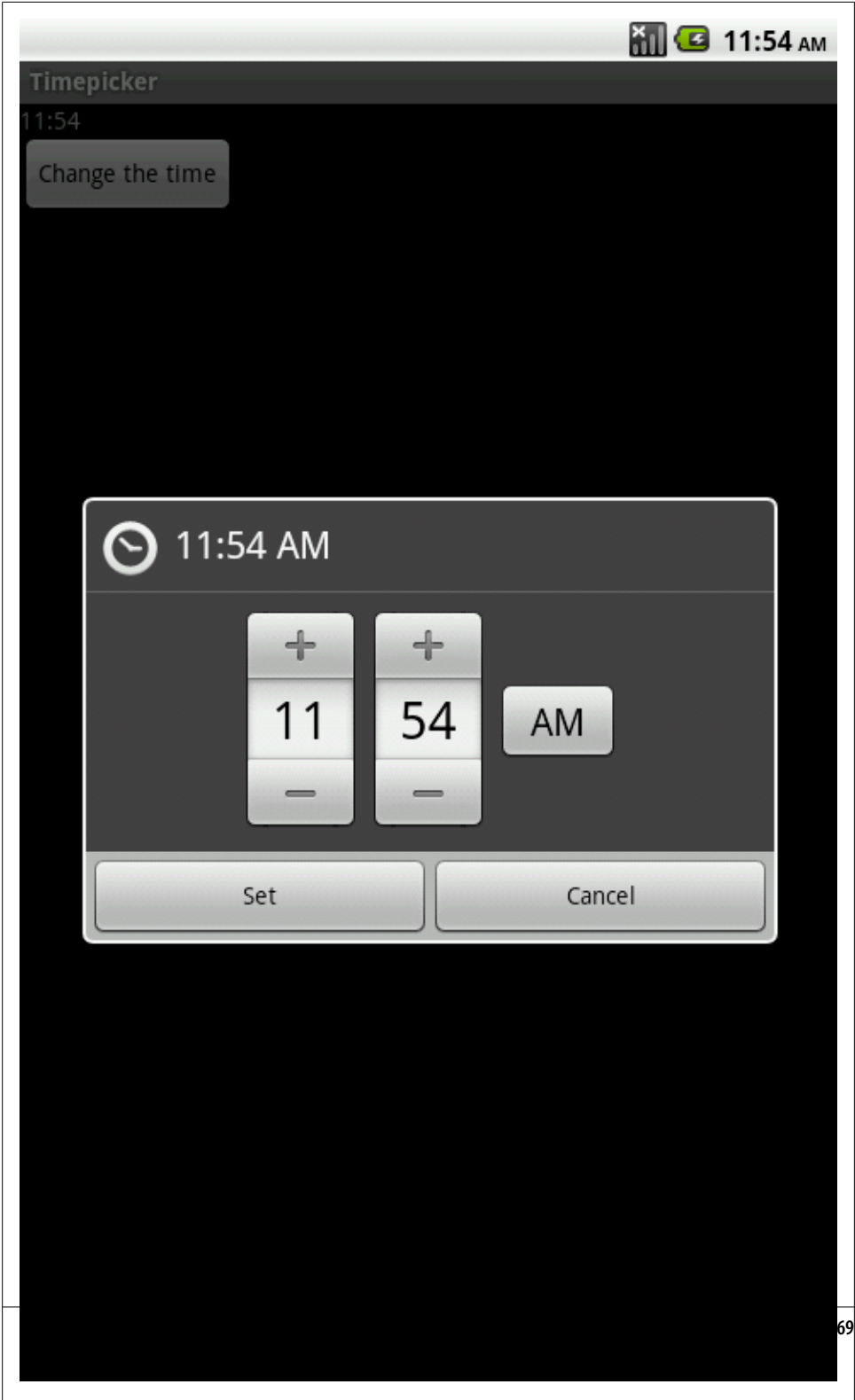


Figure 6-4.

Example 6-27.

```
$ java FormatPlurals
Found 0 items
Found 1 item
Found 2 items
$
```

The final `println` statement is short for:

Example 6-28.

```
if (n==1)
    System.out.println("Found " + n + " item");
else
    System.out.println("Found " + n + " items");
```

This is a lot longer, in fact, so Java's ternary conditional operator is worth learning.

Of course you can't use this arbitrarily, because English is a strange and somewhat idiosyncratic language. Some nouns like *bus* require 'es', while others like "cash" are collective nouns with no plural (you can have two flocks of geese or two stacks of cash, but you cannot have "two cashes"). Some nouns, like "fish", can be considered plural as-is, although "fishes" is also a correct plural.

A Better Way

The `ChoiceFormat` class from `java.text` is ideal for handling plurals; it lets you specify singular and plural (or, more generally, range) variations on the noun. It is capable of more, but here I'll show only a couple of the simpler uses. I specify the values 0, 1, and 2 (or more), and the string values to print corresponding to each number. The numbers are then formatted according to the range they fall into:

Example 6-29.

```
import java.text.*;

/**
 * Format a plural correctly, using a ChoiceFormat.
 * @author Ian F. Darwin, http://www.darwinsys.com/
 * @version $Id: FormatPluralsChoice.java,v 1.7 2010/06/22 16:31:20 ian Exp $
 */
public class FormatPluralsChoice extends FormatPlurals {

    // ChoiceFormat to just give pluralized word
    static double[] limits = { 0, 1, 2 };
    static String[] formats = { "reviews", "review", "reviews" };
    static ChoiceFormat pluralizedFormat =
        new ChoiceFormat(limits, formats);

    // ChoiceFormat to give English text version, quantified
    static ChoiceFormat quantizedFormat = new ChoiceFormat(
        "0#no reviews|1#one review|1<many reviews");
```

```

// Test data
static int[] data = { -1, 0, 1, 2, 3 };

public static void main(String[] argv) {
    System.out.println("Pluralized Format");
    for (int i : data) {
        System.out.println("Found " + i + " " +
            pluralizedFormat.format(i));
    }

    System.out.println("Quantized Format");
    for (int i : data) {
        System.out.println("Found " +
            quantizedFormat.format(i));
    }
}
}

```

Either of these loops generates similar output to the basic version. The code using the `ChoiceFormat` is slightly longer, but more general, and lends itself better to internationalization. Put the string for the "quantized" form constructor into `strings.xml` and it will be part of your localization actions.

Best Way of All (Android-only)

Create a file in `/res/values/somefilename.xml` containing something like:

Example 6-30.

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
<plurals name="numberOfSongsAvailable">
<item quantity="one">One item found.</item>
<item quantity="other">%d items found.</item>
</plurals>
</resources>

```

In your code you then use the following:

Example 6-31.

```

int count = getNumberOfSongsAvailable();
Resources res = getResources();
String songsFound = res.getQuantityString(R.plurals.numberOfSongsAvailable, count);

```

(This part suggested by Tomas Persson.)

See Also

For the Android-only way, see <http://developer.android.com/guide/topics/resources/string-resource.html#Plurals>.

Source Download URL

The source code for this example may be downloaded from this URL: <http://javacook.darwinsys.com/javasrc/numbers/FormatPluralsChoice.java>

6.14 Feed AutoCompleteTextView using a SQLite database query

Jonathan Fuerth

Problem

Although the Android documentation contains a complete working example of using AutoCompleteTextView with an ArrayAdapter, just substituting a SimpleCursorAdapter into the example does not work.

Solution

There are two extra twists to using SimpleCursorAdapter instead of ArrayAdapter:

1. You need to tell the adapter which column to use for filling the text view after the user selects a completion.
2. You need to tell the adapter how to requery based on the user's latest input in the text field. Otherwise, it shows all rows returned by the cursor and the list never shrinks to include the items of actual interest.

Discussion

The following example code would typically be found in the onCreate() method of the Activity that contains the AutoCompleteTextView. It retrieves the AutoCompleteTextView from its activity's layout, creates a SimpleCursorAdapter, configures that SimpleCursorAdapter to work with the AutoCompleteTextView, then assigns the adapter to the view.

The two important differences from the ArrayAdapter example in the Android dev guide are marked "important difference 1" and "important difference 2" in the code example. They are each covered by a short discussion following the example.

Example 6-32.

```
final AutoCompleteTextView itemName = (AutoCompleteTextView) findViewById(R.id.item_name_view);

SimpleCursorAdapter itemNameAdapter = new SimpleCursorAdapter(
    this, R.layout.completion_item, itemNameCursor, fromCol, toView);

// important difference 1
itemNameAdapter.setStringConversionColumn(
    itemNameCursor.getColumnIndexOrThrow(GroceryDBAdapter.ITEM_NAME_COL));
```

```

// important difference 2
itemNameAdapter.setFilterQueryProvider(new FilterQueryProvider() {

    public Cursor runQuery(CharSequence constraint) {
        String partialItemName = null;
        if (constraint != null) {
            partialItemName = constraint.toString();
        }
        return groceryDb.suggestItemCompletions(partialItemName);
    }
});

itemName.setAdapter(itemNameAdapter);

```

Important difference 1: With ArrayAdapter, there is no need to specify how to convert the user's selection into a String. However, SimpleCursorAdapter supports using one column for the text of the suggestion, and a different column for the text that's fed into the text field after the user selects a suggestion. Although the most common case is to use the same text for the suggestion as you get in the text field after picking it, this is *not* the default. The default is to fill the text view with the toString() representation of your cursor—something like `android.database.sqlite.SQLiteCursor@f00f00d0`.

Important difference 2: With ArrayAdapter, the system takes care of filtering the alternatives to display only those strings that start with what the user has typed into the text field so far. The SimpleCursorAdapter is more flexible, but again, the default behaviour is not useful. If you fail to write a FilterQueryProvider for your adapter, the AutoCompleteTextView will simply show the initial set of suggestions no matter what the user types. With the FilterQueryProvider, the suggestions work as expected.

TODO: there is a potential cursor leak in the code example. Update with fix and add a discussion item.

6.15 Change The Enter Key to "Next" on the Soft Keyboard

Jonathan Fuerth

Problem

Several apps, including the Web Browser and the Contacts app, replace the "Enter" key on the on-screen keyboard with a "Next" key that gives focus to the next data entry view. How do you add this kind of polish to your own apps?

Solution

Set the appropriate Input Method Editor (IME) attribute on the views in question.

Discussion

Here is a simple layout with three text fields (EditText views) and a submit button:

Note the enter key in the bottom right. Pressing it causes the currently-focused text field to expand vertically to accommodate another line of text. Not what you normally want!

Here is the code for that layout:

Example 6-33.

```
<?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">
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Field 1" />
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Field 2" />
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Field 3" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:text="Submit" />
</LinearLayout>
```

Here's a better version of the same UI, with a Next button where Enter used to be:

Besides being more convenient for users, this also prevents people from entering multiple lines of text into a field that was only intended to hold a single line.

Here's how to tell Android to display a Next button on your keyboard. Note the `android:imeOptions` attributes on each of the three `EditText` views:

Example 6-34.

```
<?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">
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
```

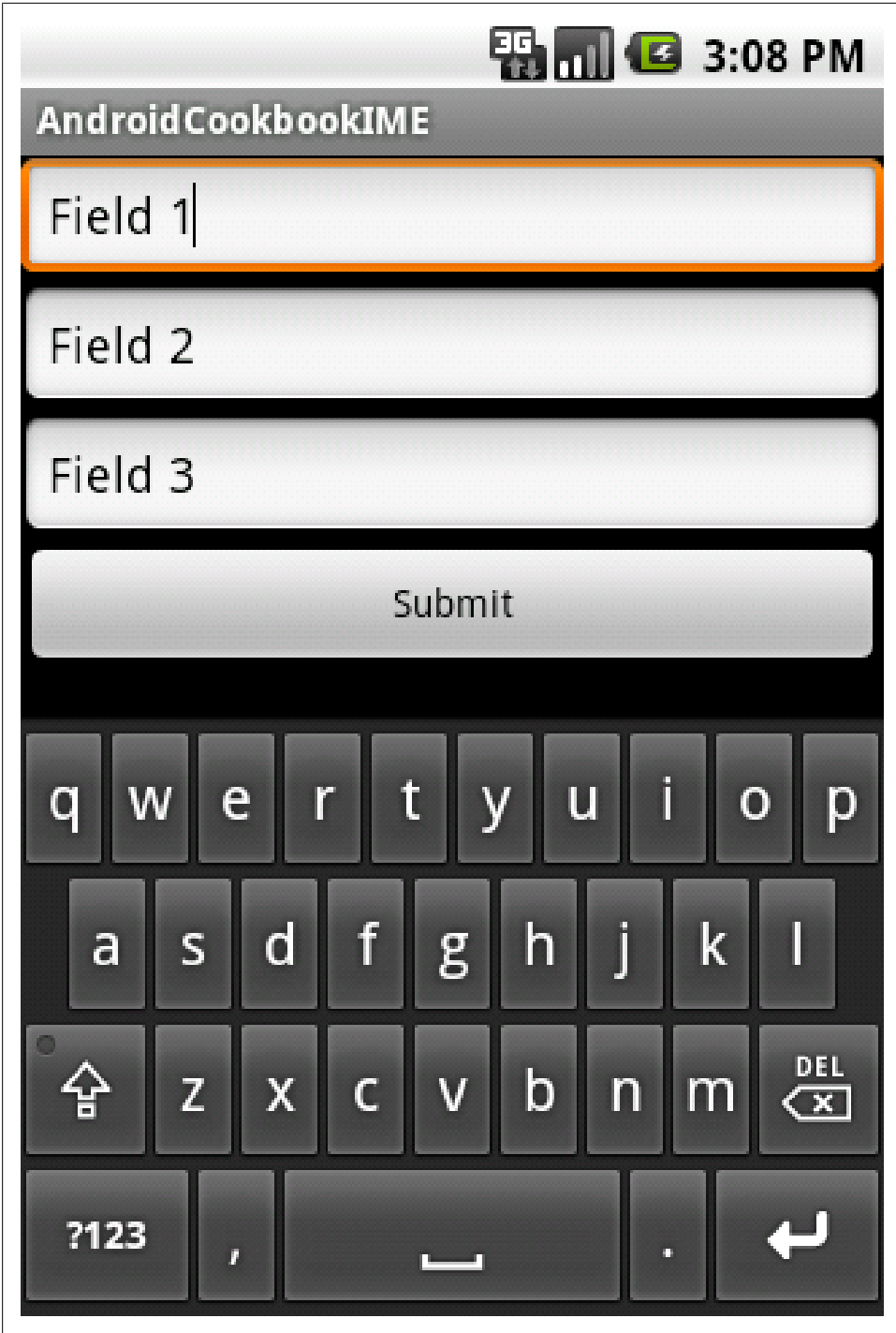


Figure 6-5.

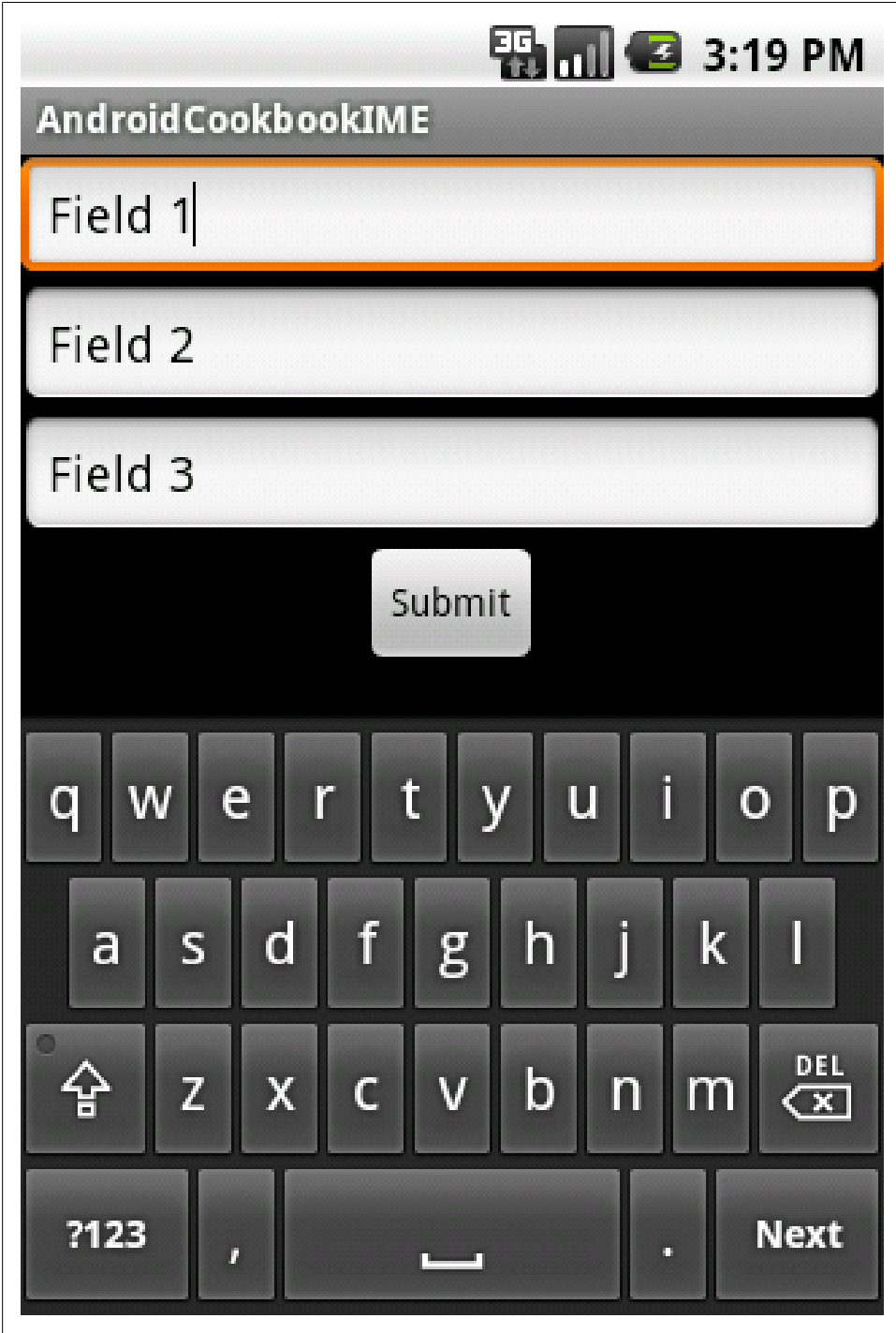


Figure 6-6.
276 | Chapter 6: Graphical User Interface

```

        android:text="Field 1"
        android:imeOptions="actionNext" />
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Field 2"
    android:imeOptions="actionNext" />
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Field 3"
    android:imeOptions="actionDone" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal"
    android:text="Submit" />
</LinearLayout>

```

Finally, notice the `actionDone` on the third text field: the button that follows is not focusable in touch mode, and if it was, it wouldn't display a keyboard anyway. As you might guess, `actionDone` puts a Done button where the enter key normally goes. Pressing the Done button simply hides the keyboard.

There are a number of refinements you can make to the appearance of the software keyboard, including hints about the input type, suggested capitalization, and even select-all-on-focus behaviour. They are all worth investigating. Every little touch can make your app more of a pleasure to use.

See Also

[The Android API documentation from `TextView`](#), especially [the section on `ImeOptions`](#).

6.16 How to Create a Simple Widget

Catarina Reis

Problem

Widgets are simple graphical user interfaces that allow users to easily interact with an existing application (activity and/or service). It's rather simple to create one. Just try it!

Solution

A guided list of steps with a simple example that allows you to create a widget that starts a service that updates its visual components.

Example: `CurrentMoodWidget`. A simple solution that presents the current mood in the form of a smiley text in a widget. The current mood smiley changes to a random mood smiley whenever the user clicks the update button (smiley image button).

(BROKEN XREF TO RECIPE -1 'File:device_0.png|thumb|alt=Initial|Initial Mood')

(BROKEN XREF TO RECIPE -1 'File:device_1.png|thumb|alt=Mood Updated|Current Mood')

Discussion

Following these steps you will be able to create a widget that calls a service and updates its visual components.

1. Create a new Android Project (`CurrentMoodWidgetProject`) : * "Current Mood" as the application name; : * `oreillymedia.cookbook.android.spikes` for the package name; : * do not create an activity; : * and min SDK version: 8 (for Android 2.2).

2. Add the text support required for the widget under the resources files (`res/values/string.xml`), according to the following name-value pairs. : * `widgettext - "current mood:"` : * `widgetmoodtext - ":"`

3. Add the image(s) that will appear in the widget's button under the `res/drawable` structure (`smile_icon.png`). (BROKEN XREF TO RECIPE -1 'File:smile_icon.png|thumb|alt=smile_icon.png|')

4. Create a new layout file inside `res/layout`, under the project structure, that will define the widget layout (`widgetlayout.xml`) according to the following structure.

Example 6-35.

```
<TextView android:text="@string/widgettext"
    android:layout_width="Odp"
    android:layout_height="wrap_content"
    android:layout_weight="0.8"
    android:layout_gravity="center_vertical"
    android:textColor="#000000"></TextView>
<TextView android:text="@string/widgetmoodtext"
    android:id="@+id/widgetMood" android:layout_width="Odp"
    android:layout_height="wrap_content"
    android:layout_weight="0.3"
    android:layout_gravity="center_vertical"
    android:textColor="#000000"></TextView>
<ImageButton android:id="@+id/widgetBtn" android:layout_width="Odp"
    android:layout_height="wrap_content"
    android:layout_weight="0.5" android:src="@drawable/smile_icon"
    android:layout_gravity="center_vertical"></ImageButton>
```

5. Now, you should provide the widget provider setup configuration:

- Create the `res/xml` folder under the project structure
- Create a xml file (`widgetproviderinfo.xml`) with the following parameters:

Example 6-36.

```
<appwidget-provider
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```

        android:minWidth="220dp"
        android:minHeight="72dp"
        android:updatePeriodMillis="86400000"
        android:initialLayout="@layout/widgetlayout">
</appwidget-provider>

```

6. Now you should create the service that reacts to the user interaction with the smiley image button (`CurrentMoodService.java`).

Example 6-37.

```

@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    super.onStart(intent, startId);
        updateMood(intent);
    stopSelf(startId);
    return START_STICKY;
}

private void updateMood(Intent intent) {
    if (intent != null){
        String requestedAction = intent.getAction();
        if (requestedAction != null && requestedAction.equals(UPDATEMOOD)){
            this.currentMood = getRandomMood();
            int widgetId = intent.getIntExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, 0);
            AppWidgetManager appWidgetMan = AppWidgetManager.getInstance(this);
            RemoteViews views = new RemoteViews(this.getPackageName(),R.layout.widgetlayout);
            views.setTextViewText(R.id.widgetMood, currentMood);
            appWidgetMan.updateAppWidget(widgetId, views);
        }
    }
}

```

7. After defining the service, it is time to implement the widget provider class (`CurrentMoodWidgetProvider.java`).

Example 6-38.

```

@Override
public void onUpdate(Context context, AppWidgetManager appWidgetManager,
    int[] appWidgetIds) {
    super.onUpdate(context, appWidgetManager, appWidgetIds);

    for (int i=0; i<appWidgetIds.length; i++) {
        int appWidgetId = appWidgetIds[i];
        RemoteViews views = new RemoteViews(context.getPackageName(), R.layout.widgetlayout);
        Intent intent = new Intent(context, CurrentMoodService.class);
        intent.setAction(CurrentMoodService.UPDATEMOOD);
        intent.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID, appWidgetId);
        PendingIntent pendingIntent = PendingIntent.getService(context, 0, intent, 0);
        views.setOnClickPendingIntent(R.id.widgetBtn, pendingIntent);
        appWidgetManager.updateAppWidget(appWidgetId, views);
    }
}

```

8. Finally it is necessary to declare the Service and the App Widget Provider in the Manifest (`AndroidManifest.xml`).

Example 6-39.

```
<service android:name=".CurrentMoodService">
</service>
<receiver android:name=".CurrentMoodWidgetProvider">
  <intent-filter>
    <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
  </intent-filter>
  <meta-data android:name="android.appwidget.provider"
    android:resource="@xml/widgetproviderinfo" />
</receiver>
```

Source Download URL

The source code for this example may be downloaded from this URL: <http://sites.google.com/site/androidsourcecode/src/CurrentMoodWidgetProject.rar>

6.17 Make a View Shake

Ian Darwin

Problem

You want a View component to shake for a few seconds to catch the user's attention.

Solution

Create an animation in the XML, then call the View object's `startAnimation()`, using the convenience routing `loadAnimation()` to load the XML.

Discussion

The Animation specification is created in XML files in the `anim` directory. In this example I want the text entry field to be able to shake either left-to-right (to emulate a person shaking their head side-to-side, meaning "no" or "I disagree" in many parts of the world) or up and down (a person nodding agreement). So I create two animations, `horizontal.xml` and `vertical.xml`. Here is `horizontal.xml`:

Example 6-40.

```
<?xml version="1.0" encoding="utf-8"?>
<translate
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:fromXDelta="0"
  android:toXDelta="10"
  android:duration="1000"
  android:interpolator="@anim/cycler"
/>
```

The file `vertical.xml` is identical except it uses `fromYDelta` and `toYDelta`.

The Interpolator - the function that drives the animation - is contained in another file, `cycler.xml`, shown below.

Example 6-41.

```
<?xml version="1.0"?>
<cycleInterpolator
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:cycles="5"/>
```

To apply one of the two animations to a View component, you need a reference to it. You can of course use the common `findViewById(R.id.XXX)`. You can also use the Activity method `getCurrentFocus()` if you are dealing with the current input (focus) view component; this avoids coupling to the name of a particular component, if you know that your animation will always apply to the current input object. In my code I know this is true because the animation startup is done in an `onClick()` method. Thirdly, you could use the View that is passed into the `onClick()` method, but that would make the Button shake, not the text field.

I won't show the whole application, but here is the `onClick()` method which contains all the animation code:

Example 6-42.

```
@Override
public void onClick(View v) {
    String answer = answerEdit.getText().toString();
    if ("yes".equalsIgnoreCase(answer)) {
        getCurrentFocus().startAnimation(
            AnimationUtils.loadAnimation(getApplicationContext(),
                R.anim.vertical));
        return;
    }
    if ("no".equalsIgnoreCase(answer)) {
        getCurrentFocus().startAnimation(
            AnimationUtils.loadAnimation(getApplicationContext(),
                R.anim.horizontal));
        return;
    }
    Toast.makeText(this, "Try to be more definite, OK?",
        Toast.LENGTH_SHORT).show();
}
```

The shaking effect is convenient for drawing the user's attention to an input that is incorrect, but it can easily be overdone. Use judiciously!

6.18 Using CheckBoxes and RadioButtons

Blake Meike

Problem

You want to offer the user a set of choices that is more limited than a list.

Solution

Use CheckBoxes, RadioButtons or Spinners as appropriate

Discussion

These Views are probably familiar to you from other user interfaces. They allow the user to choose from multiple options. CheckBoxes are typically used when you want to offer multiple selections with a yes/no or true/false choice for each. RadioButtons are used when only one choice is allowed at a time.

Spinners are similar to 'combo boxes' in some GUI frameworks, and are now covered in the Recipe (BROKEN XREF TO RECIPE -1 'Dropdown Choooser with the Spinner class'). Android has adapted these familiar components to make them more useful in a touchscreen environment. The screenshot above shows the three types of multiple-choice Views laid out on an Android application, with the Spinner pulled down to show the options. The layout XML file that created the screen in the figure looks like this:

Example 6-43.

```
<?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"
    >
<CheckBox
    android:id="@+id/cbxBox1"
    android:layout_width="20dp"
    android:layout_height="20dp"
    android:checked="false"
    />
<TextView
    android:id="@+id/txtCheckBox"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="CheckBox: Not checked"
    />
<RadioGroup
    android:id="@+id/rgGroup1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton android:id="@+id/RB1" android:text="Button1" />
    <RadioButton android:id="@+id/RB2" android:text="Button2" />
    <RadioButton android:id="@+id/RB3" android:text="Button3" />
</RadioGroup>
<TextView
```

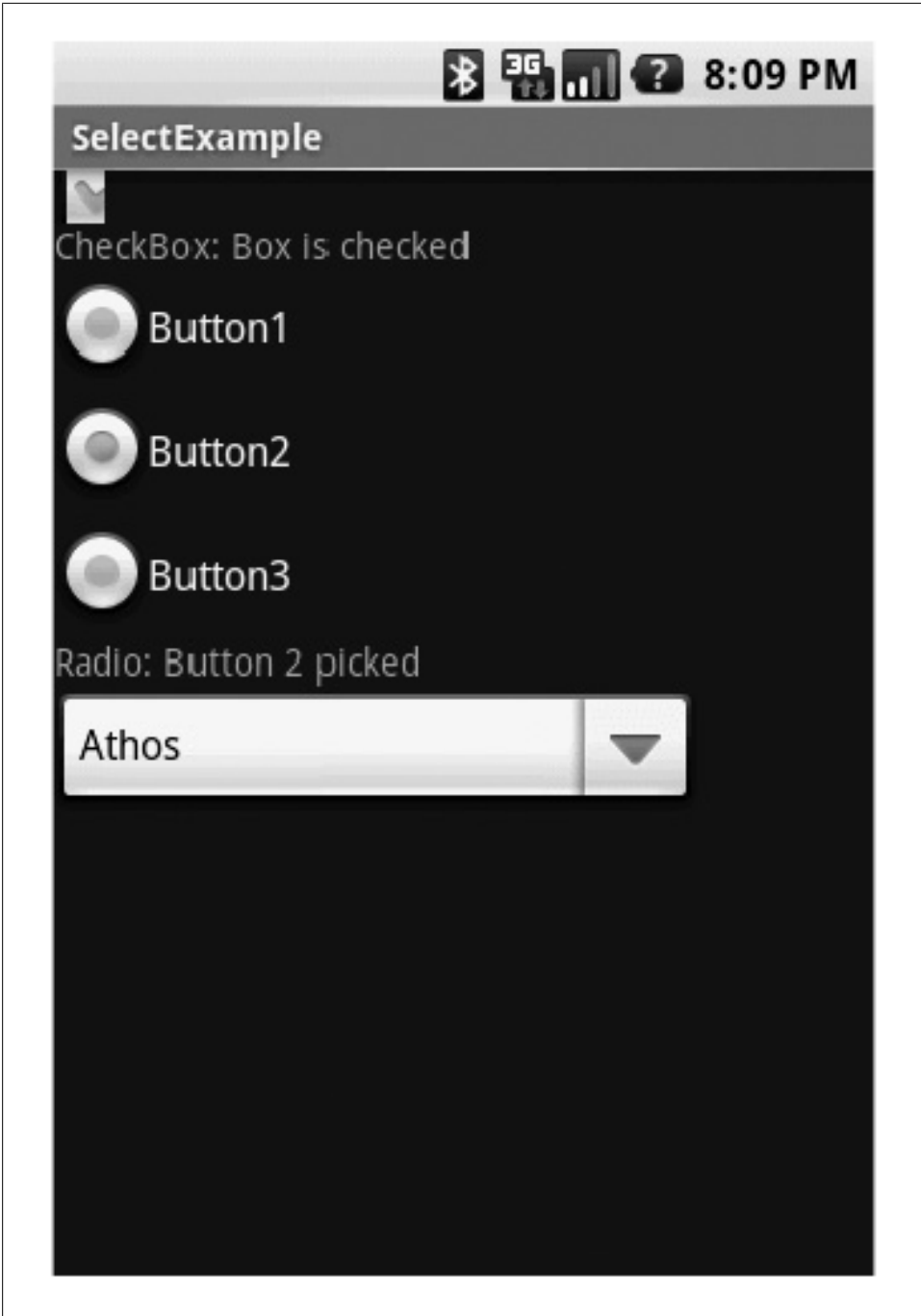


Figure 6-7.

```

        android:id="@+id/txtRadio"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="RadioGroup: Nothing picked"
    />
<Spinner
    android:id="@+id/spnMusketeers"
    android:layout_width="250dp"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="2dp"
    />
</LinearLayout>

```

The XML file just lists each View we want on the screen along with the attributes we want. A RadioGroup is really a ViewGroup, so it contains the appropriate RadioButton Views. Here is the Java file that responds to user clicks.

Example 6-44.

```

package com.oreilly.select;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import com.google.android.maps.GeoPoint;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.CheckBox;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;

public class SelectExample extends Activity {
    private CheckBox checkBox;
    private TextView txtCheckBox, txtRadio;
    private RadioButton rb1, rb2, rb3;
    private Spinner spnMusketeers;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        checkBox = (CheckBox) findViewById(R.id.cbxB1);
        txtCheckBox = (TextView) findViewById(R.id.txtCheckBox);
        txtRadio = (TextView) findViewById(R.id.txtRadio);
        rb1 = (RadioButton) findViewById(R.id.RB1);
        rb2 = (RadioButton) findViewById(R.id.RB2);
        rb3 = (RadioButton) findViewById(R.id.RB3);
        spnMusketeers = (Spinner) findViewById(R.id.spnMusketeers);
    }
}

```

```

// React to events from the CheckBox
checkBox.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener() {
    public void onClick(View v){
        if (checkBox.isChecked()) {
            txtCheckBox.setText("CheckBox: Box is checked");
        }
        else
        {
            txtCheckBox.setText("CheckBox: Not checked");
        }
    }
});
// React to events from the RadioGroup
rb1.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
    public void onClick(View v){
        txtRadio.setText("Radio: Button 1 picked");
    }
});
rb2.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
    public void onClick(View v){
        txtRadio.setText("Radio: Button 2 picked");
    }
});
rb3.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
    public void onClick(View v){
        txtRadio.setText("Radio: Button 3 picked");
    }
});
// Set up the Spinner entries
List<String> lsMusketeers = new ArrayList<String>();
lsMusketeers.add("Athos");
lsMusketeers.add("Porthos");
lsMusketeers.add("Aramis");
ArrayAdapter<String> aspnMusketeers =
    new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item,
        lsMusketeers);
aspnMusketeers.setDropDownViewResource
    (android.R.layout.simple_spinner_dropdown_item);
spnMusketeers.setAdapter(aspnMusketeers);
// Set up a callback for the spinner
spnMusketeers.setOnItemSelectedListener(
    new OnItemSelectedListener() {
        public void onNothingSelected(AdapterView<?> arg0) { }
        public void onItemSelected(AdapterView<?> parent, View v,
            int position, long id) {
            // Code that does something when the Spinner value changes
        }
    }
);
}
}

```

These three Views work as follows:

CheckBox

The CheckBox View takes care of flipping its state back and forth and displaying the appropriate checkmark when the state is true. All you have to do is create an OnClickListener to catch click events, and you can add whatever code you want to react.

RadioGroup

As mentioned earlier, the RadioGroup View is really a ViewGroup that contains any number of RadioButton Views. The user can select only one of the buttons at a time, and you capture the selections by setting OnClickListener for each RadioButton. Note that clicking on one of the RadioButtons does not fire a click event for the RadioGroup.

Taken together, these three Views let you provide a short set of choices and have the user select one or multiple choices from the choices offered.

6.19 Creating a Notification in the Status Bar

Ian Darwin

Problem

You want to place a notification icon in the notice bar at the top of the screen, to call the user's attention to an event that occurred *or* to remind them of a service that is running in the background.

Solution

Create a Notification object, provide it with a PendingIntent which wraps a real Intent for what to do when the user selects the Notification. At the same time you pass in the PendingIntent you also pass a title and text to be displayed in the notification area. You should set the AUTO_CANCEL flag unless you want to remove the notification from the notice bar manually. Finally, you find and ask the NotificationManager to display (notify) your Notification, associating with it an ID so that you can refer to it later, e.g to remove it.

Discussion

Notifications are normally used from a running Service class to notify (hence the name) the user of some fact. Either an event has occurred (receipt of a message, loss of contact with a server, or whatever), or, you just want to remind the user that a long-running Service is still running.

Create a Notification object; the constructor takes an Icon id, the text to display briefly in the Notice bar, and the time at which the event occurred (timestamp in milliseconds). Before you can show the Notification you have to provide it with a PendingIntent for what to do when the user selects the Notification, and ask the NotificationManager to display your Notification.

The following shows doing the right thing in the wrong place; Notifications are normally shown from Services. This Recipe is just focusing on the Notification API.

Example 6-45.

```
public class Main extends Activity {

    private static final int NOTIFICATION_ID = 1;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        int icon = R.drawable.icon; // Preferably a distinct icon

        // Create the notification itself
        String noticeMeText = getString(R.string.noticeMe);
        Notification n =
            new Notification(
                icon, noticeMeText, System.currentTimeMillis());

        // And the Intent of what to do when user selects notification
        Context applicationContext = getApplicationContext();
        Intent notIntent = new Intent(this, NotificationTarget.class);
        PendingIntent wrappedIntent =
            PendingIntent.getActivity(this, 0,
                notIntent, Intent.FLAG_ACTIVITY_NEW_TASK);

        // Condition the Notification
        String title = getString(R.string.title);
        String message = getString(R.string.message);
        n.setLatestEventInfo(applicationContext, title,
            message, wrappedIntent);
        n.flags |= Notification.FLAG_AUTO_CANCEL;

        // Now invoke the Notification Service
        String notifService = Context.NOTIFICATION_SERVICE;
        NotificationManager mgr =
            (NotificationManager) getSystemService(notifService);
        mgr.notify(NOTIFICATION_ID, n);
    }
}
```

The following is the file strings.xml:

Example 6-46.

```
<resources>
    <string name="app_name">NotificationDemo</string>
    <string name="hello">Hello World, Main!</string>
    <string name="noticeMe">Lookie Here!!</string>
    <string name="title">My Notification</string>
</resources>
```

```
<string name="message">This is my message</string>
<string name="target_name">Notification Target</string>
<string name="thanks">Thank you selecting the notification.</string>
</resources>
```

The "noticeMe" string appears briefly (a few seconds only) in the status bar:

Then the main view will appear;

When the user drags the status bar down, it expands to show the details, which includes the icons and the title and message strings (unless you are using a custom view, see below).

If you have set auto-clear, the notification will no longer appear in the status bar. If the user selects the notification box, the PendingIntent becomes current. Ours simply shows a simple Thank You notification. If the user clicks the Clear button, however, the Intent does not get run (even with Auto-Clear, which can leave you in a bit of a lurch).

TODO updating the status.

TODO Other flags, e.g., LED colors.

TODO Custom View.

See Also

The official tutorial is at <http://developer.android.com/guide/topics/ui/notifiers/notifications.html>.

6.20 Autocompletion with Icons/Images

Wagied Davids

Problem

Searching for items/entries, that can occur in more than a single category. Perform autocompletion and simultaneously make the difference in categories immediately apparent using icons. For example, the term Pain (in bold) occurs in both Symptoms category as well as Reactions category, but also text in the Notes category. Categories:

Symptoms: Pain, Depression, Weakness Reactions: Pain, Inflammation, Itchiness

Notes: I had a pain in my buttocks.

Solution

Visualize the categories and items to be searched simultaneously using category icons.

File: search_bar_fragment.xml

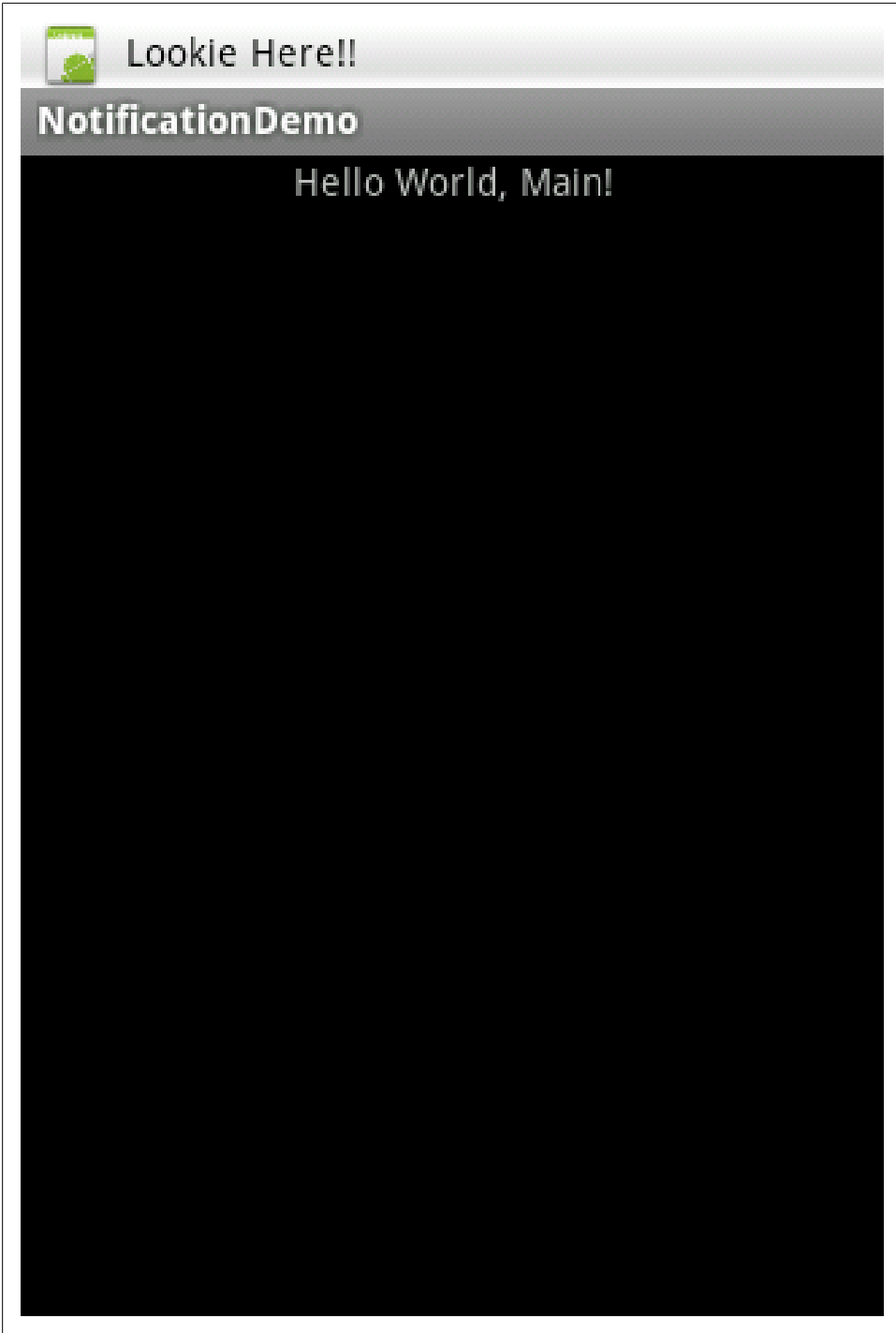


Figure 6-8.

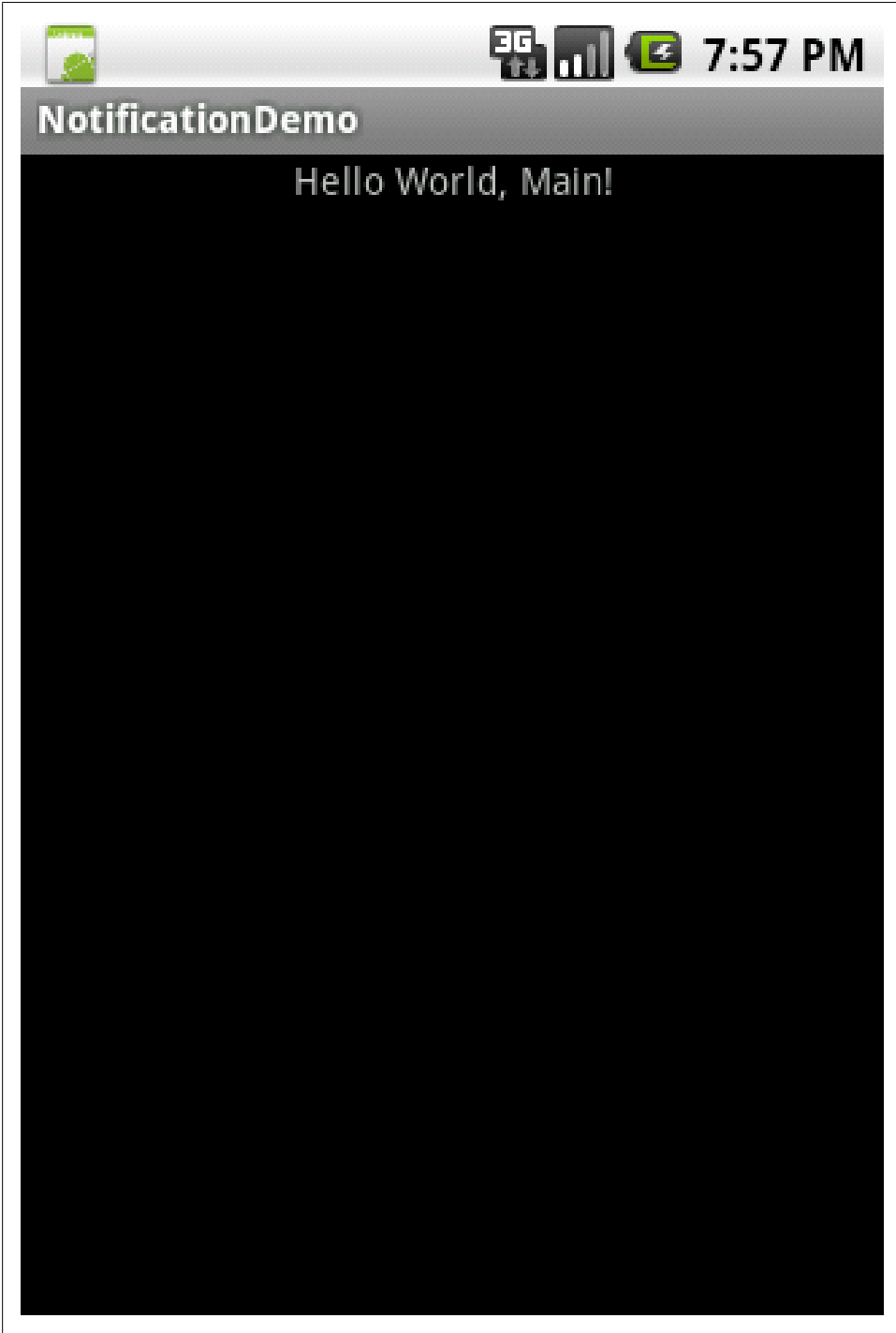


Figure 6-9.
290 | Chapter 6: Graphical User Interface

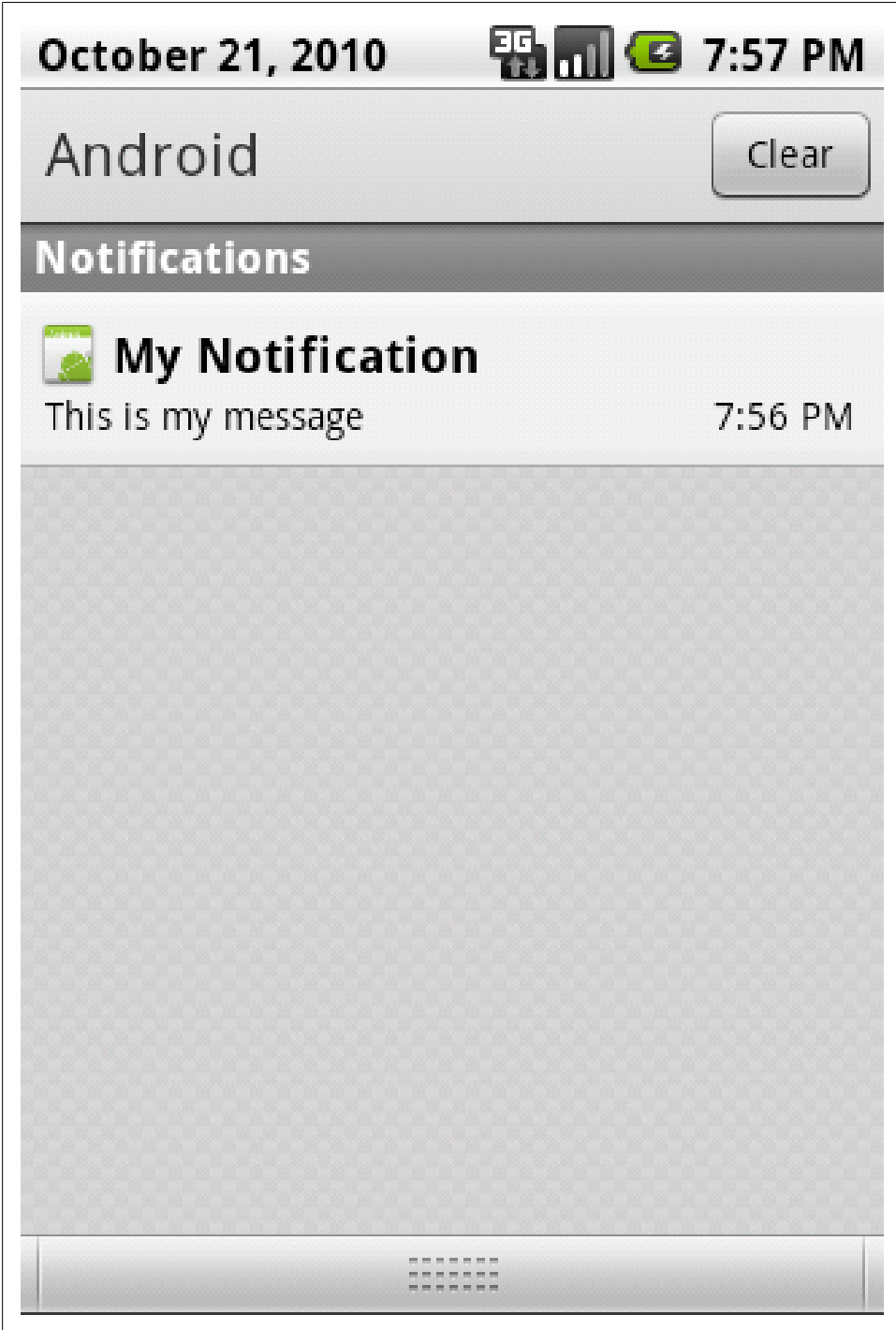


Figure 6-10.

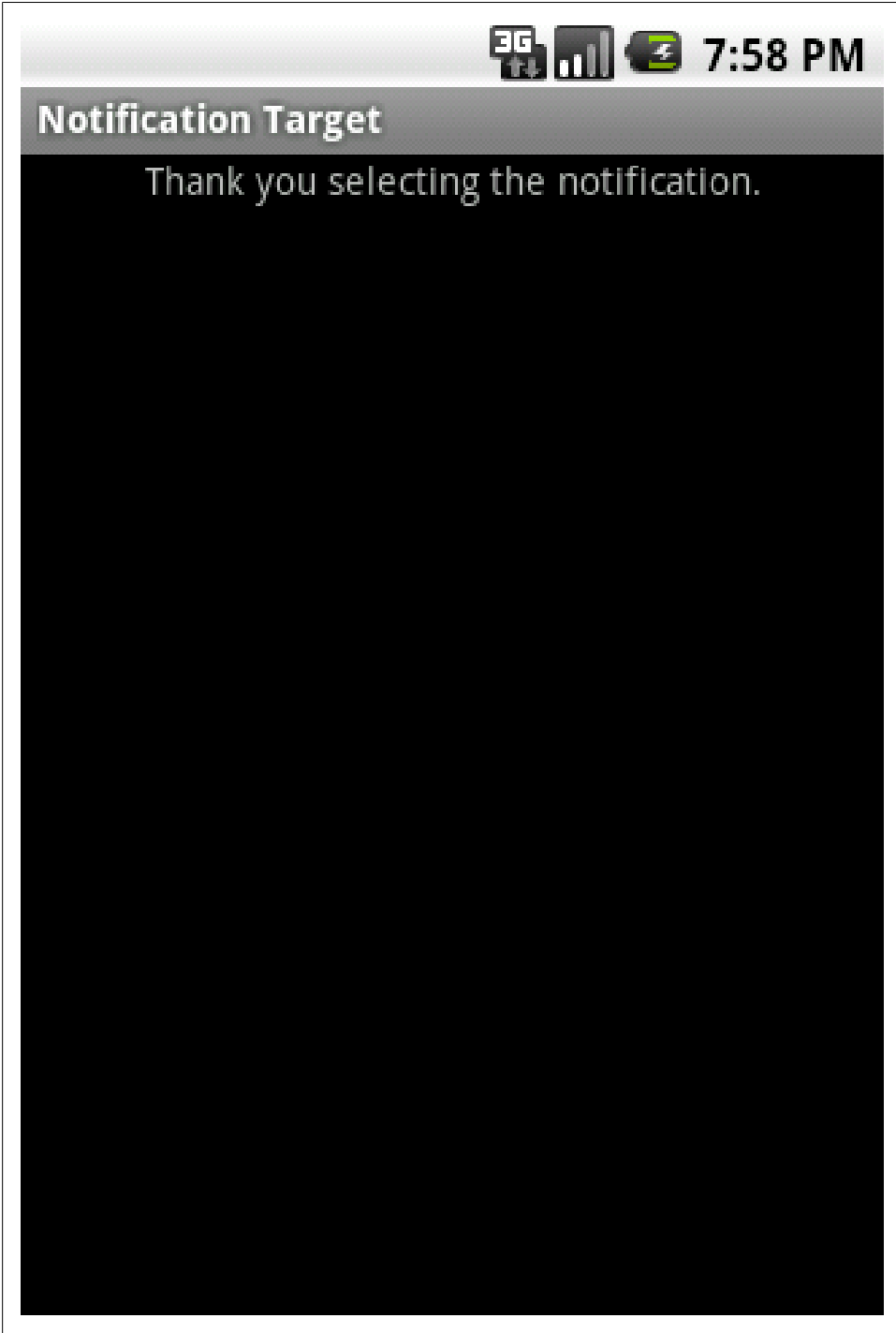


Figure 6-11.
292 | Chapter 6: Graphical User Interface

Example 6-47.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">

    <AutoCompleteTextView
        android:id="@+id/autoCompleteSearch"
        android:hint="Enter search term ..."
        android:singleLine="true"
        android:maxLines="1"
        android:layout_weight="1.0"
        android:layout_width="220dip"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip">
    </AutoCompleteTextView>

    <ImageButton
        android:id="@+id/searchButton"
        android:src="@drawable/search_icon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </ImageButton>

</LinearLayout>
```

File: SearchArrayAdapter.java

Example 6-48.

```
import java.util.ArrayList;
import java.util.List;

import android.content.Context;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ImageView;
import android.widget.TextView;

import com.pfizer.android.R;
import com.pfizer.android.model.JournalEntry;
import com.pfizer.android.utils.CategoryIconRetriever;

public class SearchItemArrayAdapter extends ArrayAdapter<JournalEntry>
{
    private static final String tag = "SearchItemArrayAdapter";
    private JournalEntry journalEntry;
    private TextView autoItem;
    private ImageView categoryIcon;
    private int journalEntryTypeId;
```

```

private List<JournalEntry> journalEntryList = new ArrayList<JournalEntry>();
private final CategoryIconRetriever iconRetriever;

/**
 *
 * @param context
 * @param textViewResourceId
 * @param objects
 */
public SearchItemArrayAdapter(Context context, int textViewResourceId, List<JournalEntry> objects)
{
    super(context, textViewResourceId, objects);
    iconRetriever = new CategoryIconRetriever(context);
    journalEntryList = objects;
    Log.d(tag, "Search List -> journalEntryList := " + journalEntryList.toString());
}

@Override
public int getCount()
{
    return this.journalEntryList.size();
}

@Override
public JournalEntry getItem(int position)
{
    JournalEntry journalEntry = this.journalEntryList.get(position);
    Log.d(tag, "*-> Retrieving JournalEntry @ position: " + String.valueOf(position) + " : " + j
    return journalEntry;
}

@Override
public View getView(int position, View convertView, ViewGroup parent)
{
    View row = convertView;
    LayoutInflater inflater = LayoutInflater.from(getContext());

    if (row == null)
    {
        row = inflater.inflate(R.layout.search_listitem_icon, parent, false);
    }

    journalEntry = this.journalEntryList.get(position);
    String searchItem = journalEntry.title;
    autoItem = (TextView) row.findViewById(R.id.search_auto_item);
    autoItem.setText(searchItem);

    // Get a reference to ImageView holder
    categoryIcon = (ImageView) row.findViewById(R.id.category_icon);
    categoryIcon.setImageBitmap(iconRetriever.getJournalEntryTypeIcon(journalEntry.typeId));

    return row;
}
}

```

Discussion

TODO: EXPLANATION CODE

6.21 Creating your own Custom Title Bar

Shraddha Shravagi

Problem

You cannot have any buttons or any custom text in the standard title bar.

Solution

Here's how to implement your own TitleBar:

1. Create an XML file for title bar
2. Create a class that uses the title bar and implements the button functionality
3. Change your layout files
4. Extend your activities from the custom class that you created in step 2

Discussion

1. Create an XML file for title bar

Here is an maintitlebar.xml file which has one textview and 3 image buttons. It has orientation = horizontal.

Example 6-49.

```
<RelativeLayout xmlns:android=
    "http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="40dp"
    android:orientation="horizontal" android:paddingLeft="5dp"
    >

    <TextView android:id="@+id/title" android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Symphony's GHealth Demo"
        />
    <View android:id="@+id/View01" android:layout_width="1dp"
        android:layout_height="500dip"
        android:background="#2B497B" android:layout_toLeftOf="@+id/facebookBtn">
    </View>
    <!-- Facebook button -->
    <ImageView android:src="@drawable/icon_facebook"
        android:layout_toLeftOf="@+id/twitterBtn" android:layout_width="28dp"
        android:layout_height="28dp" android:id="@id/facebookBtn"
        android:clickable="true" />
```

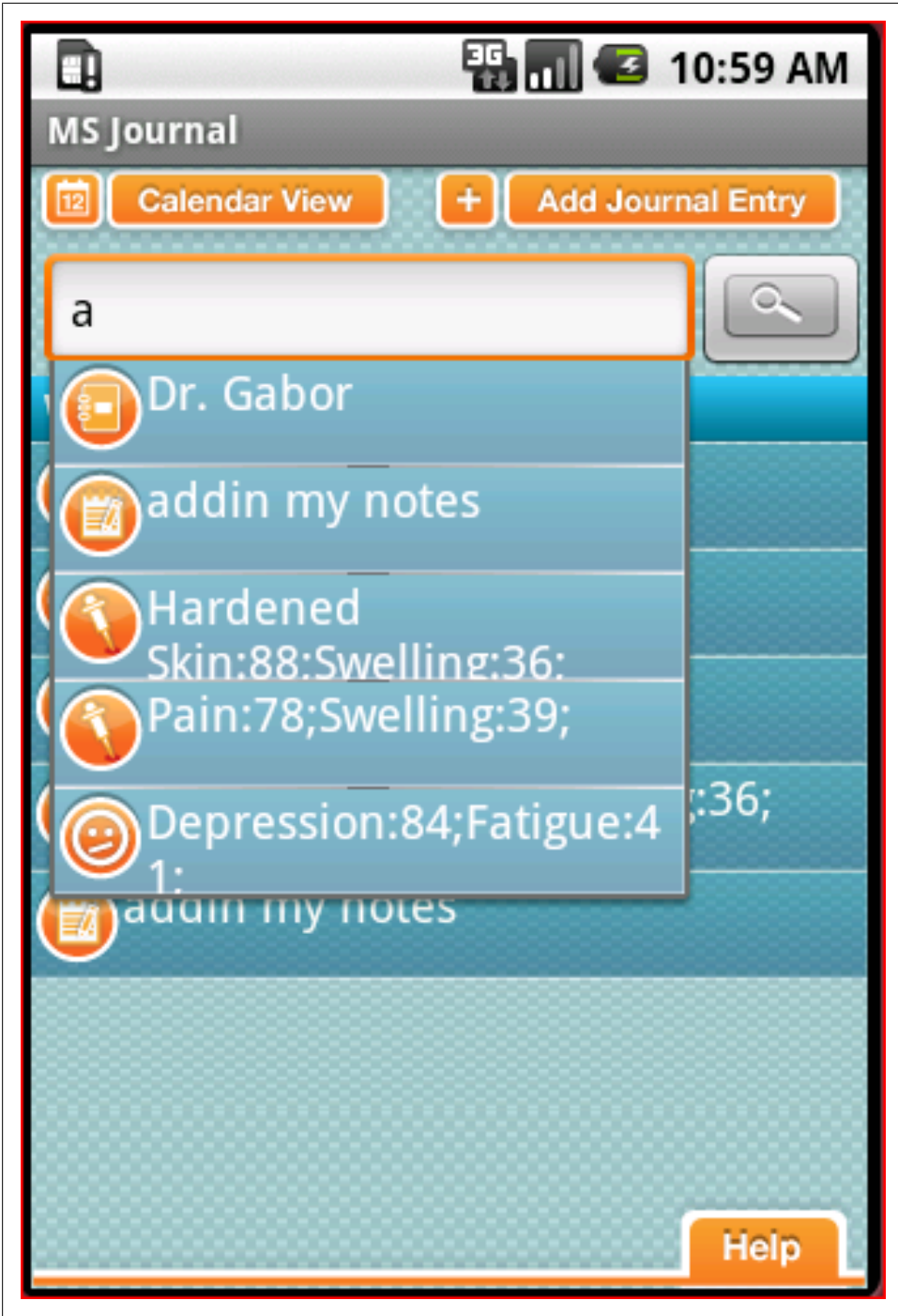


Figure 6-12.

```

<!-- Twitter button -->
<ImageView android:src="@drawable/icon_twitter"
    android:clickable="true"
    android:layout_width="28dp" android:layout_height="28dp" android:id="@id/twitterBtn"
    android:layout_marginLeft="3dp" android:layout_marginRight="3dp"
    android:layout_toLeftOf="@+id/linkedinBtn" />
<!-- LinkedIn button -->
<ImageView android:src="@drawable/icon_linkedin"
    android:layout_width="28dp"
    android:layout_height="28dp" android:clickable="true"
    android:layout_alignParentRight="true"
    android:id="@id/linkedinBtn" />
</RelativeLayout>

```

2. Create a class that uses the title bar and implements the button functionality

Here is the class which is the most important one.

1. First we will have to request the custom title bar
2. Set your layout file
3. Set the title bar

Example 6-50.

```

public class CustomWindow extends Activity {
    protected TextView title;
    protected ImageView icon;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //Request for custom title bar
        this.requestWindowFeature(Window.FEATURE_CUSTOM_TITLE);
        //set to your layout file
        setContentView(R.layout.main);
        //Set the titlebar layout
        this.getWindow().setFeatureInt(Window.FEATURE_CUSTOM_TITLE, R.layout.maintitlebar);
    }
    public void facebookBtnClicked(View v)
    {
        //Implement the button click event
    }
    public void twitterBtnClicked(View v)
    {
        //Implement the button click event
    }
    public void linkedinBtnClicked(View v)
    {
        //Implement the button click event
    }
}

```

3. Change your layout files

For every layout file where you want to implement the custom title bar use `match_parent` in `layout_height` and `layout_width`, e.g.:

Example 6-51.

```
<LinearLayout android:id="@+id/LinearLayout01"
    android:layout_width="match_parent" android:layout_height="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:background="#E5E5E5">
```

4. Extend your activities from the custom class that you created in step 2

Example 6-52.

```
//CustomWindow will take care of loading the title bar
public class Credentials extends CustomWindow
{
    //set the layout file
    setContentView(R.layout.login);
}
```

And here's how your activity will look like:

Although there is no obligation to use a separate class for the titlebar implementation but it is good coding practice.

Thus you have your own custom title bar shown in your activity in few easy steps

6.22 iPhone-like wheel picker for selection

Wagied Davids

Problem

Want a selection UI component similar to iPhone wheel picker.

Solution

File: Main.java

Example 6-53.

```
import kankan.wheel.widget.ArrayWheelAdapter;
import kankan.wheel.widget.OnWheelChangeListener;
import kankan.wheel.widget.OnWheelScrollListener;
import kankan.wheel.widget.WheelView;
import android.app.Activity;
import android.os.Bundle;
import android.widget.EditText;
import android.widget.TextView;

public class Main extends Activity
{
    // TODO: Externalize string-array
    String wheelMenu1[] = new String[]{"Right Arm", "Left Arm", "R-Abdomen", "L-Abdomen", "Right Thigh",
    String wheelMenu2[] = new String[]{"Upper", "Middle", "Lower"};
```

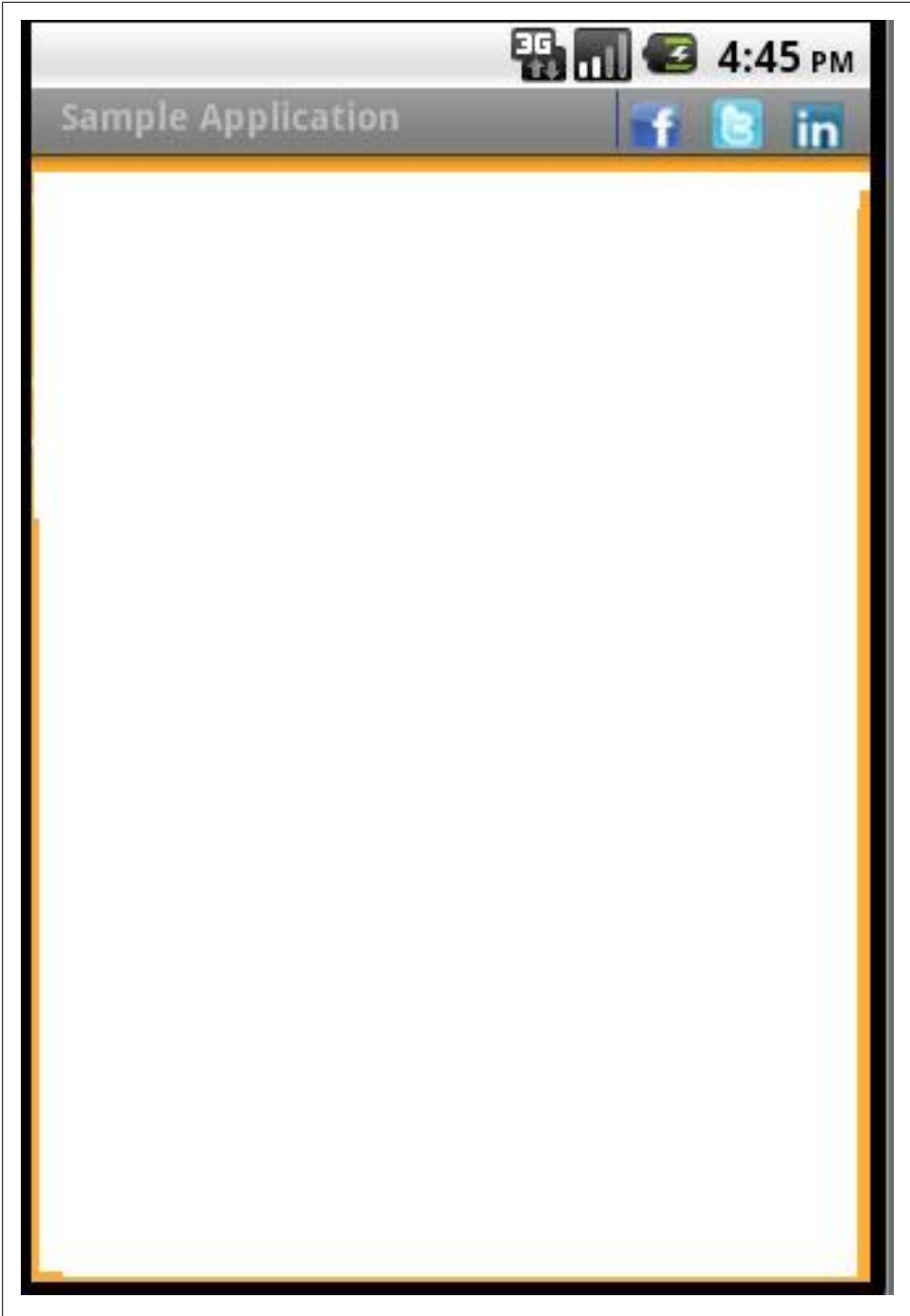


Figure 6-13.

```

String wheelMenu3[] = new String[]{"R", "L"};

// Wheel scrolled flag
private boolean wheelScrolled = false;

private TextView text;
private EditText text1;
private EditText text2;
private EditText text3;

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

    initWheel1(R.id.p1);
    initWheel2(R.id.p2);
    initWheel3(R.id.p3);

    text1 = (EditText) this.findViewById(R.id.r1);
    text2 = (EditText) this.findViewById(R.id.r2);
    text3 = (EditText) this.findViewById(R.id.r3);
    text = (TextView) this.findViewById(R.id.result);
}

// Wheel scrolled listener
OnWheelScrollListener scrolledListener = new OnWheelScrollListener()
{
    public void onScrollStarts(WheelView wheel)
    {
        wheelScrolled = true;
    }

    public void onScrollEnds(WheelView wheel)
    {
        wheelScrolled = false;
        updateStatus();
    }
};

// Wheel changed listener
private final OnWheelChangedListener changedListener = new OnWheelChangedListener()
{
    public void onChanged(WheelView wheel, int oldValue, int newValue)
    {
        if (!wheelScrolled)
        {
            updateStatus();
        }
    }
};

/**
 * Updates entered PIN status

```

```

*/
private void updateStatus()
{
    text1.setText(wheelMenu1[getWheel(R.id.p1).getCurrentItem()]);
    text2.setText(wheelMenu2[getWheel(R.id.p2).getCurrentItem()]);
    text3.setText(wheelMenu3[getWheel(R.id.p3).getCurrentItem()]);

    text.setText(wheelMenu1[getWheel(R.id.p1).getCurrentItem()] + " - " + wheelMenu2[getWheel(R.
}

/**
 * Initializes wheel
 *
 * @param id
 *         the wheel widget Id
 */

private void initWheel1(int id)
{
    WheelView wheel = (WheelView) findViewById(id);
    wheel.setAdapter(new ArrayWheelAdapter(wheelMenu1));
    wheel.setVisibleItems(2);
    wheel.setCurrentItem(0);
    wheel.addChangingListener(changedListener);
    wheel.addScrollingListener(scrolledListener);
}

private void initWheel2(int id)
{
    WheelView wheel = (WheelView) findViewById(id);
    wheel.setAdapter(new ArrayWheelAdapter(wheelMenu2));
    wheel.setVisibleItems(2);
    wheel.setCurrentItem(0);
    wheel.addChangingListener(changedListener);
    wheel.addScrollingListener(scrolledListener);
}

private void initWheel3(int id)
{
    WheelView wheel = (WheelView) findViewById(id);

    wheel.setAdapter(new ArrayWheelAdapter(wheelMenu3));
    wheel.setVisibleItems(2);
    wheel.setCurrentItem(0);
    wheel.addChangingListener(changedListener);
    wheel.addScrollingListener(scrolledListener);
}

/**
 * Returns wheel by Id
 *
 * @param id
 *         the wheel Id
 * @return the wheel with passed Id
 */

```

```

private WheelView getWheel(int id)
{
    return (WheelView) findViewById(id);
}

/**
 * Tests wheel value
 *
 * @param id
 *         the wheel Id
 * @param value
 *         the value to test
 * @return true if wheel value is equal to passed value
 */
private int getWheelValue(int id)
{
    return getWheel(id).getCurrentItem();
}
}

```

Discussion

Now you can with Android-Wheel the iPhone-like WheelPicker for Android: Android-Wheel: <http://code.google.com/p/android-wheel/> It comes with a handy ScrollListener for listen to touch events on the wheel component.

6.23 Simple Calendar

Wagied Davids

Problem

You desire a simple and native Month view Calendar, which can easily be skinned.

Solution

File: simple_calendar_view.xml

Example 6-54.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/bg">

    <Button
        android:id="@+id/selectedDayMonthYear"
        android:textColor="#FFFFFF"
        android:textAppearance="?android:attr/textAppearanceMedium"

```

```

        android:background="@drawable/button_blue_background"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
</Button>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">

    <ImageView
        android:id="@+id/prevMonth"
        android:src="@drawable/left_cal_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </ImageView>
    <Button
        android:id="@+id/currentMonth"
        android:layout_weight="0.8"
        android:textColor="#FFFFFF"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:background="@drawable/button_blue_background"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </Button>
    <ImageView
        android:id="@+id/nextMonth"
        android:src="@drawable/right_cal_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </ImageView>
</LinearLayout>

<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <ImageView
        android:id="@+id/calendarheader"
        android:src="@drawable/blue_bg_with_text"
        android:layout_gravity="center"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
    </ImageView>
</LinearLayout>

<GridView
    android:id="@+id/calendar"
    android:horizontalSpacing="-1px"
    android:verticalSpacing="-1px"
    android:numColumns="7"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
</GridView>

```

```
</LinearLayout>
```

File: day_gridcell.xml

Example 6-55.

```
&lt;?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">

    <Button
        android:id="@+id/day_gridcell"
        android:layout_gravity="center"
        android:textColor="#FFFFFF"
        android:background="@drawable/calendar_button_selector"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </Button>
</LinearLayout>
```

File: Main.java

Example 6-56.

```
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.List;
import java.util.Locale;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.text.format.DateFormat;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.View.OnClickListener;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.GridView;
import android.widget.ImageView;
import android.widget.Toast;

public class Main extends Activity implements OnClickListener
{
    private static final String tag = "Main";
    private Button selectedDayMonthYearButton;
```

```

private Button currentMonth;
private ImageView prevMonth;
private ImageView nextMonth;
private GridView calendarView;
private GridCellAdapter adapter;
private Calendar _calendar;
private int month, year;
private final DateFormat dateFormatter = new DateFormat();
private static final String dateTemplate = "MMMM yyyy";

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.simple_calendar_view);

    _calendar = Calendar.getInstance(Locale.getDefault());
    month = _calendar.get(Calendar.MONTH);
    year = _calendar.get(Calendar.YEAR);

    selectedDayMonthYearButton = (Button) this.findViewById(R.id.selectedDayMonthYear);
    selectedDayMonthYearButton.setText("Selected: ");

    prevMonth = (ImageView) this.findViewById(R.id.prevMonth);
    prevMonth.setOnClickListener(this);

    currentMonth = (Button) this.findViewById(R.id.currentMonth);
    currentMonth.setText(dateFormatter.format(dateTemplate, _calendar.getTime()));

    nextMonth = (ImageView) this.findViewById(R.id.nextMonth);
    nextMonth.setOnClickListener(this);

    calendarView = (GridView) this.findViewById(R.id.calendar);

    // Initialised
    adapter = new GridCellAdapter(getApplicationContext(), R.id.day_gridcell, month, year);
    adapter.notifyDataSetChanged();
    calendarView.setAdapter(adapter);
}

@Override
public void onClick(View v)
{
    if (v == prevMonth)
    {
        if (month <= 1)
        {
            month = 11;
            year--;
        } else
        {
            month--;
        }
    }
}

```



```

        }

        adapter = new GridCellAdapter(getApplicationContext(), R.id.day_gridcell, month, year,
            _calendar.set(year, month, _calendar.get(Calendar.DAY_OF_MONTH));
            currentMonth.setText(dateFormatter.format(dateTemplate, _calendar.getTime()));

        adapter.notifyDataSetChanged();
        calendarView.setAdapter(adapter);
    }
}

if (v == nextMonth)
{
    if (month >= 11)
    {
        month = 0;
        year++;
    } else
    {
        month++;
    }

    adapter = new GridCellAdapter(getApplicationContext(), R.id.day_gridcell, month, year,
        _calendar.set(year, month, _calendar.get(Calendar.DAY_OF_MONTH));
        currentMonth.setText(dateFormatter.format(dateTemplate, _calendar.getTime()));
        adapter.notifyDataSetChanged();
        calendarView.setAdapter(adapter);
    }
}

// Inner Class
public class GridCellAdapter extends BaseAdapter implements OnClickListener
{
    private static final String tag = "GridCellAdapter";
    private final Context _context;
    private final List<String> list;
    private final String[] weekdays = new String[] { "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun" };
    private final String[] months = { "January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December" };
    private final int[] daysOfMonth = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
    private final int month, year;
    private int daysInMonth, prevMonthDays;
    private final int currentDayOfMonth;
    private Button gridcell;

    // Days in Current Month
    public GridCellAdapter(Context context, int textViewResourceId, int month, int year)
    {
        super();
        this._context = context;
        this.list = new ArrayList<String>();
        this.month = month;
        this.year = year;

        Log.d(tag, "Month: " + month + " " + "Year: " + year);
        Calendar calendar = Calendar.getInstance();
        currentDayOfMonth = calendar.get(Calendar.DAY_OF_MONTH);
    }
}

```

```

        printMonth(month, year);
    }

    public String getItem(int position)
    {
        return list.get(position);
    }

    @Override
    public int getCount()
    {
        return list.size();
    }

    private void printMonth(int mm, int yy)
    {
        // The number of days to leave blank at
        // the start of this month.
        int trailingSpaces = 0;
        int leadSpaces = 0;
        int daysInPrevMonth = 0;
        int prevMonth = 0;
        int prevYear = 0;
        int nextMonth = 0;
        int nextYear = 0;

        GregorianCalendar cal = new GregorianCalendar(yy, mm, currentDayOfMonth);

        // Days in Current Month
        daysInMonth = daysOfMonth[mm];
        int currentMonth = mm;
        if (currentMonth == 11)
        {
            prevMonth = 10;
            daysInPrevMonth = daysOfMonth[prevMonth];
            nextMonth = 0;
            prevYear = yy;
            nextYear = yy + 1;
        } else if (currentMonth == 0)
        {
            prevMonth = 11;
            prevYear = yy - 1;
            nextYear = yy;
            daysInPrevMonth = daysOfMonth[prevMonth];
            nextMonth = 1;
        } else
        {
            prevMonth = currentMonth - 1;
            nextMonth = currentMonth + 1;
            nextYear = yy;
            prevYear = yy;
            daysInPrevMonth = daysOfMonth[prevMonth];
        }

        // Compute how much to leave before before the first day of the

```

```

// month.
// getDay() returns 0 for Sunday.
trailingSpaces = cal.get(Calendar.DAY_OF_WEEK) - 1;

if (cal.isLeapYear(cal.get(Calendar.YEAR)) && mm == 1)
{
    ++daysInMonth;
}

// Trailing Month days
for (int i = 0; i < trailingSpaces; i++)
{
    list.add(String.valueOf((daysInPrevMonth - trailingSpaces + 1) + i) + "-GREY");
}

// Current Month Days
for (int i = 1; i <= daysInMonth; i++)
{
    list.add(String.valueOf(i) + "-WHITE" + "-" + months[mm] + "-" + yy);
}

// Leading Month days
for (int i = 0; i < list.size() % 7; i++)
{
    Log.d(tag, "NEXT MONTH:= " + months[nextMonth]);
    list.add(String.valueOf(i + 1) + "-GREY" + "-" + months[nextMonth] + "-" + n);
}
}

@Override
public long getItemId(int position)
{
    return position;
}

@Override
public View getView(int position, View convertView, ViewGroup parent)
{
    Log.d(tag, "getView ...");
    View row = convertView;
    if (row == null)
    {
        // ROW INFLATION
        Log.d(tag, "Starting XML Row Inflation ... ");
        LayoutInflater inflater = (LayoutInflater) _context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        row = inflater.inflate(R.layout.day_gridcell, parent, false);

        Log.d(tag, "Successfully completed XML Row Inflation!");
    }

    // Get a reference to the Day gridcell
    gridcell = (Button) row.findViewById(R.id.day_gridcell);
    gridcell.setOnClickListener(this);

    // ACCOUNT FOR SPACING

```

```

        Log.d(tag, "Current Day: " + currentDayOfMonth);
        String[] day_color = list.get(position).split("-");
        gridcell.setText(day_color[0]);
        gridcell.setTag(day_color[0] + "-" + day_color[2] + "-" + day_color[3]);

        if (day_color[1].equals("GREY"))
            {
                gridcell.setTextColor(Color.LTGRAY);
            }
        if (day_color[1].equals("WHITE"))
            {
                gridcell.setTextColor(Color.WHITE);
            }
        if (position == currentDayOfMonth)
            {
                gridcell.setTextColor(Color.BLUE);
            }

        return row;
    }

    @Override
    public void onClick(View view)
    {
        String date_month_year = (String) view.getTag();
        Toast.makeText(getApplicationContext(), date_month_year, Toast.LENGTH_LONG).show();
        selectedDayMonthYearButton.setText("Selected: " + date_month_year);
    }
}
}
}

```

Discussion

Logic layer: Standard use of the Calendar (`java.util.Calendar`) is used for creating a calendar object, and setting the next and previous months and year.

For the GridView a custom adapter called, `GridCellAdapter` is created, `GridCellAdapter(getApplicationContext(), R.id.day_gridcell, month, year)`, which takes as parameters a Context, text view resource identifier, integer for month and year.

For previous month, the `onClick` event is intercepted and the month is decremented by 1, and for the next month it is incremented by 1. Special circumstances for the December are included, where the previous month would be November of the same year, and the next month would be January of the following year.

View layer: The calendar month view is created using a GridView component, containing a GridView cell element modelled in `day_gridcell.xml`.

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.filefactory.com/file/b460e9h/n/SimpleCalendar.zip>

6.24 Formatting Numbers

Ian Darwin

Problem

You need to format numbers, because the default formatting of `Double.toString()` and friends do not give you enough control over how the results are displayed.

Solution

Use `String.format()` or one of the `NumberFormat` subclasses.

Discussion

The `printf()` function was included in the C programming language in the 1970's, and used in many other languages since, including Java. Here's a simple `printf` example in Java SE:

Example 6-57.

```
System.out.printf("Hello %s at %s%n", userName, time);
```

This could be expected to print something like

Example 6-58.

```
Hello Robin at Wed Jun 16 08:38:46 EDT 2010
```

Since we don't use `System.out` in Android, you'll be relieved to note that you can get the same `String` that would be printed, for putting it into a `View`, by using:

Example 6-59.

```
String msg = String.format("Hello %s at %s%n", userName, time);
```

If you haven't seen `printf` before, the first argument is obviously the format code string, and all the other arguments (`userName` and `time`) are values to be formatted. The format codes begin with a percent sign (%) and have at least one "type" code; common type codes are shown in the table.

Table 6-1. Some common format codes

Character	Meaning
s	String (convert primitive values using defaults; convert objects by <code>toString</code>)
d	Decimal integer (int, long)
f	Floating point (float, double)
n	Newline
t	Time/Date formats, Java-specific; see below

The default date formatting is pretty ugly, so we often need to expand on it. The `printf` formatting capabilities are actually housed in the `java.util.Formatter` class, to which reference should be made for the full details of its formatting language.

Unlike `printf` in other languages you may have used, all these format routines optionally allow you to refer to arguments by their number, by putting a number plus a dollar sign after the `'%'` lead-in but before the formatting code proper, for example `"%2$3.1f"` means to format the second argument as a decimal number with three characters and one digit after the decimal place. This numbering can be used for two purposes: to change the order in which arguments print (often useful with internationalization), and to refer to a given argument more than once. The date/time format character `'t'` requires a second character after it, such as `Y` for the year, `'m'` for month, and so on. Here we take the `time` argument and extract several fields from it:

Example 6-60.

```
msg = String.format("Hello at %1$tB %1$td, %1$tY%n", time);
```

This might format as `July 4, 2010`.

To print numbers with a specific precision, you can use `'f'` with a width and a precision, such as

Example 6-61.

```
msg = String.format("Latitude: %10.6f", latitude);
```

This might yield:

Example 6-62.

```
Latitude: -79.281818
```

While such formatting is OK for specific uses such as latitudes and longitudes, for general use such as currencies, it may give you too much control.

General Formatters

Java has an entire package, `java.text`, full of formatting routines as general and flexible as anything you might imagine. As with `printf`, it has an involved formatting language, described in the online documentation page. Consider the presentation of numbers. In North America, the number "one thousand twenty-four and a quarter" is written `1,024.25`, in most of Europe it is `1 024,25`, and in some other part of the world it might be written `1.024,25`. Not to mention how currencies and percentages are formatted! Trying to keep track of this yourself would drive the average small software developer around the bend rather quickly.

Fortunately, the `java.text` package includes a `Locale` class, and, furthermore, the Java or Android runtime automatically sets a default `Locale` object based on the user's environment; this code works the same on desktop Java as it does in Android. To provide

formatters customized for numbers, currencies, and percentages, the `NumberFormat` class has static factory methods that normally return a `DecimalFormat` with the correct pattern already instantiated. A `DecimalFormat` object appropriate to the user's locale can be obtained from the factory method `NumberFormat.getInstance()` and manipulated using set methods. Surprisingly, the method `setMinimumIntegerDigits()` turns out to be the easy way to generate a number format with leading zeros. Here is an example:

Example 6-63.

```
import java.text.NumberFormat;

/*
 * Format a number our way and the default way.
 */
public class NumFormat2 {
    /** A number to format */
    public static final double data[] = {
        0, 1, 22d/7, 100.2345678
    };

    public static void main(String[] av) {
        // Get a format instance
        NumberFormat form = NumberFormat.getInstance();

        // Tailor it to look like 999.99[99]
        form.setMinimumIntegerDigits(3);
        form.setMinimumFractionDigits(2);
        form.setMaximumFractionDigits(4);

        // Now print using it.
        for (int i=0; i<data.length; i++)
            System.out.println(data[i] + "\tformats as " +
                form.format(data[i]));
    }
}
```

This prints the contents of the array using the `NumberFormat` instance `form`. We show running it as a main program instead of in an Android application just to isolate the effects of the `NumberFormat`.

```
$ java NumFormat2 0.0 formats as 000.00 1.0 formats as 001.00 3.142857142857143
formats as 003.1429 100.2345678 formats as 100.2346
```

You can also construct a `DecimalFormat` with a particular pattern or change the pattern dynamically using `applyPattern()`. Some of the more common pattern characters are shown in this table.

Table 6-2. DecimalFormat pattern characters

Character	Explanation
#	Numeric digit (leading zeros suppressed)

Character	Explanation
0	Numeric digit (leading zeros provided)
.	Locale-specific decimal separator (decimal point)
,	Locale-specific grouping separator (comma in English)
-	Locale-specific negative indicator (minus sign)
%	Shows the value as a percentage
;	Separates two formats: the first for positive and the second for negative values
'	Escapes one of the above characters so it appears as itself
Anything else	Appears as itself

The `NumFormatTest` program uses one `DecimalFormat` to print a number with only two decimal places and a second to format the number according to the default locale:

Example 6-64.

```
import java.text.DecimalFormat;
import java.text.NumberFormat;

public class NumFormatTest {
    /** A number to format */
    public static final double int1Number = 1024.25;
    /** Another number to format */
    public static final double ourNumber = 100.2345678;

    public static void main(String[] av) {

        NumberFormat defForm = NumberFormat.getInstance();
        NumberFormat ourForm = new DecimalFormat("#0.##");
        // toPattern() will reveal the combination of #0., etc
        // that this particular Locale uses to format with
        System.out.println("defForm's pattern is " +
            ((DecimalFormat)defForm).toPattern());
        System.out.println(int1Number + " formats as " +
            defForm.format(int1Number));
        System.out.println(ourNumber + " formats as " +
            ourForm.format(ourNumber));
        System.out.println(ourNumber + " formats as " +
            defForm.format(ourNumber) + " using the default format");
    }
}
```

This program prints the given pattern and then formats the same number using several formats:

Example 6-65.

```
$ java NumFormatTest
defForm's pattern is #,##0.###
1024.25 formats as 1,024.25
```


100.2345678 formats as 100.23

100.2345678 formats as 100.235 using the default format

See Also

Chapter Ten of the *Java Cookbook*; Part VI of *Java I/O* by Elliott Rusty Harold.

6.25 Start a Second Screen from the First

Daniel Fowler

Problem

New App developers need a simple example on how to open another screen, thus understanding how Android handles UI creation.

Solution

Building upon the Hello World Eclipse example another screen is loaded from a new button to demonstrate the principles of starting a new UI screen.

Discussion

An Android application (App) will interact with a user through one or more screens. Each screen presenting information and user interface (UI) elements, such as buttons, lists, sliders, edit boxes and many others. The number of screens depends upon the required functionality of the App and the type of Android device. A low cost Android phone may have a 2.5 inch display, an expensive phone may have a 4.5 inch display, and a tablet may have a 7 inch or 10 inch display. An App may only need one screen for functionality on a tablet, two or three screens on the high end phone, four or five on a low cost phone.

Each screen presented to the user is controlled by an **Activity**. The Activity is responsible for creating and displaying the screen and managing the UI elements. The Android **View** is the basic building block for UIs. Each screen element, such as a **Button** or **EditText**, is provided in the package `android.widget`. Screen elements are derived from **View**. They are placed on to the screen within containers derived from a **ViewGroup**, for example a **LinearLayout** (**ViewGroups** are also derived from **View**). A variety of **ViewGroup** layouts can be used, horizontal, vertical, table, grid and others.

The home screens can hold special types of **View** commonly referred to as *Widgets*; these are small UI gadgets that can be used to provide feedback from an App to the user without the need for a full App to be open. These *App Widgets* should not be confused with the package `android.widget`. The latter holds the various types of screen elements, the former is the commonly used name for home screen gadgets. *App Widgets* are defined using **RemoteViews** which are also part of the `android.widget` package.

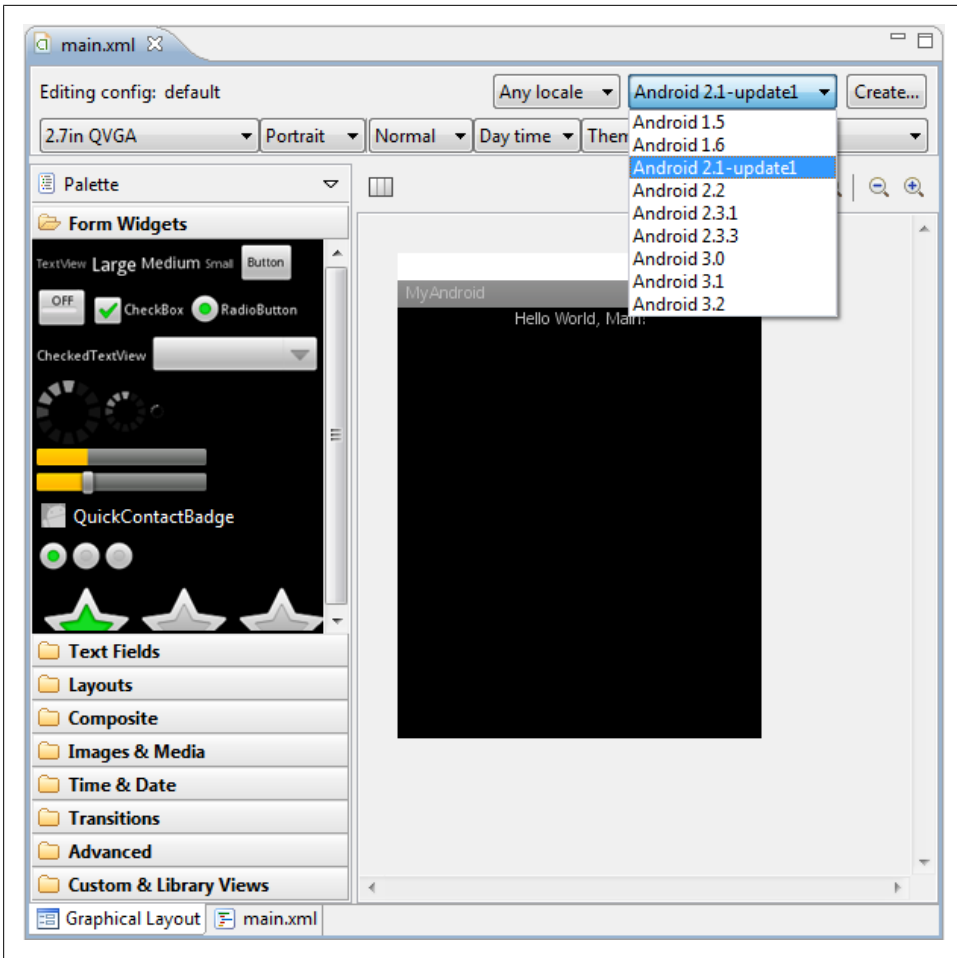


Figure 6-14.

It is possible to see the many types of Views and ViewGroups available in Android by opening or creating an Android layout resource file in Eclipse (in the project folder *res/layout*). When the resource file is open click on the *Graphical Layout* tab at the bottom of the editor. A toolbar of all available UI elements will be shown on the left of the editor. It is possible to filter by API level using the drop down towards the top right of the editor pane.

A Fragment can be defined which is a reusable piece of screen. A Fragment is also laid out using ViewGroups and Views. A Fragment can then be used on more than one screen, thus defining a section of UI once when the same section needs to be used on several screens.

As soon as an App has more than one screen defined there will be a need to load the second screen from the first. In other operating systems a second screen is often loaded directly by the first screen. Due to the design of Android an App can never directly start a new screen; it has to ask the Android operating system to start it. This is because Android was designed for mobility from the start. Android needs full control of an App to enable efficient handling of events outside of the App. Events that must interrupt the user, such as a telephone call or low battery condition; events that notify the user, such as incoming mail or a reminder firing, and the user leaving the App to deal with that notification. The user may also open another App. A variety of things can happen that will need Android to have fine control of how an App executes and responds. When Android starts a screen it knows what is running and their state. Android can dispatch messages to the activities and they can react to unexpected events accordingly. This is also why an App does not have a main method for programs as on other systems (as mention in the [Recipe 1.7](#) recipe). A main method is not required because Android itself is controlling the start up.

To get a screen up and running in an App the following is required:

1. The definition of the screen must be composed in a layout.
2. An Activity must be defined in a Java class file to handle the screen.
3. Android must be notified that the Activity exists, via the Apps manifest file.
4. The App must tell Android when it is required to start the new screen.

As an example we can add another screen to the 'MyAndroid' App in the [Recipe 1.4](#) recipe. The new screen will also contain a simple message and will be started when a button is pressed on the opening screen. Open Eclipse and open the *MyAndroid* project as created in the *Hello World* recipe. First we will add three strings, one for the new screen's title, one for the message on the new screen and one for the caption for the button that will be used to start the new screen. In the project tree in the *Package Explorer* open the *strings.xml* file in the *res/values* folder. Add three strings, one with the name *screen2Title* with value *Screen 2*, one named *hello2* with the value *Hello! Again.*, and one named *next* with the value *Next*. The *strings.xml* file will look like this:

Example 6-66.

```
&lt;?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">Hello World, Main!</string>
  <string name="app_name">MyAndroid</string>
  <string name="screen2Title">Screen 2</string>
  <string name="hello2">Hello! Again.</string>
  <string name="next">Next</string>
</resources>
```

From the *File* menu (or using the context menu on the project tree) select *New* and then *Android XML File*. Set the following fields in the dialog that opens keeping all others as default:

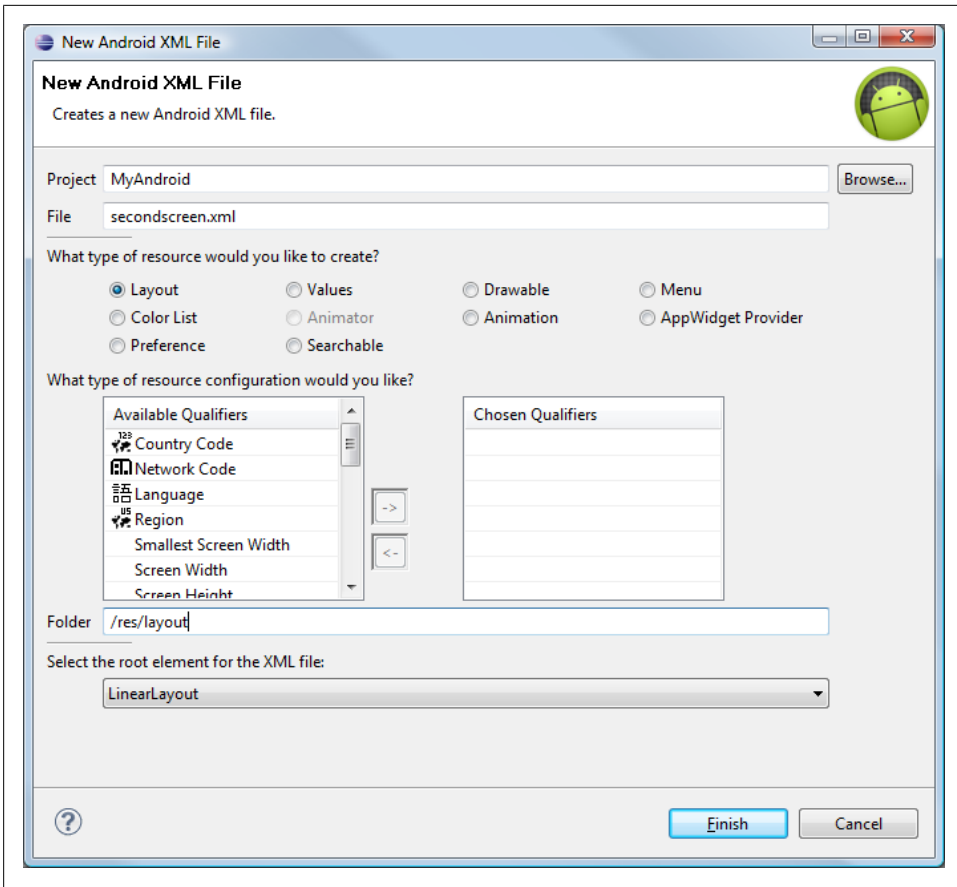


Figure 6-15.

"File"	secondscreen.xml
"Type of Resource"	Layout
"Folder"	/res/layout

Select finish.

With *secondscreen.xml* open either drag a `TextView` on to the screen in the *Graphical Layout* pane, or in the XML pane enter the `TextView` code. Set the `TextView` properties as follows:

Layout width	fill_parent
Layout height	wrap_content
Text	@string/hello2

Text size 10pt

The *secondscreen.xml* file should contain the following.

Example 6-67.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello2"
    android:gravity="center_horizontal"
    android:textSize="10pt"></TextView>
</LinearLayout>
```

Open the *main.xml* in the *res/layout* folder. Either drag a Button on to the screen in the *Graphical Layout* or add the Button in the XML view. Set the Button properties as follows:

Layout width	wrap_content
Layout height	wrap_content
Id	@+id/nextButton
Text	@string/next

Example 6-68.

```
<?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"
    android:gravity="center_horizontal" />
  <Button
    android:id="@+id/nextButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/next" />
</LinearLayout>
```

From the *File* menu (or using the context menu on the project tree) select *New* and then *Class*. Set the following fields in the dialog that opens keeping all others as default:

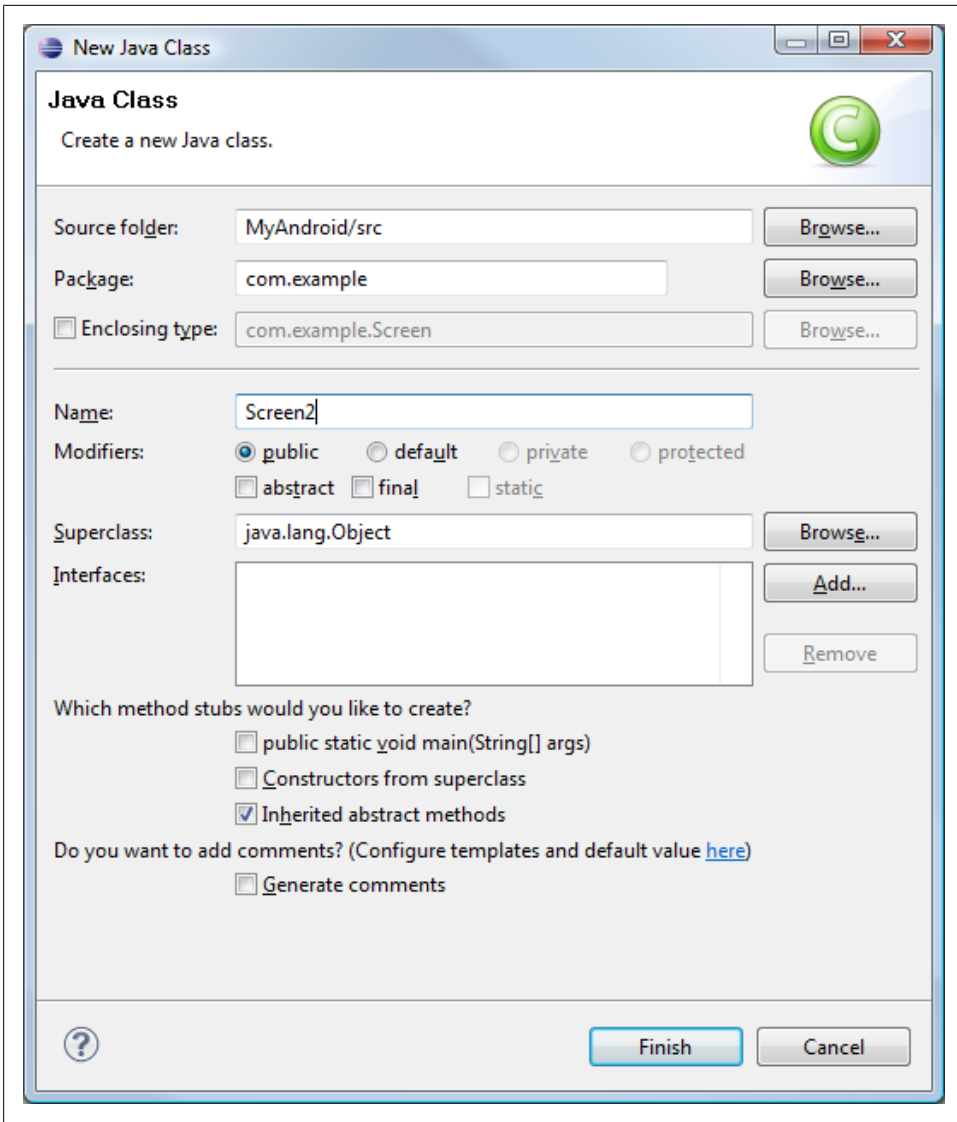


Figure 6-16.

Source folder	MyAndroid/src
Package	com.example
Name	Screen2

Select *Finish*.

Within the `Screen2.java` file we extend the class to be a subclass of `Activity` and override the `onCreate` method, the same way as in the `Main` class. We then call `setContentView` passing the new `secondscreen` layout. All resource references are accessed via a generated Java class named `R`, hence the reference to the new screen's layout is via `R.layout.secondscreen` (the `R` class is generated from the files and folders under the `res` folder). With the required imports the `Screen2.java` file will look like this:

Example 6-69.

```
package com.example;

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

public class Screen2 extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.secondscreen);
    }
}
```

The button needs code to tell Android of our intention to start the activity that contains the new screen. This can be achieved by passing the name of the required activity in an `Intent` object to the `startActivity` method when the button is pressed. The `startActivity` method is available on the `Context` object; `Context` has a host of useful methods which provide access to the environment in which the App is executing. `Activity` is a subclass of `Context` so the `startActivity` method is always available within an `Activity`. By using `startActivity` Android gets the opportunity to perform any required house-keeping and then fire up the `Activity` class that was defined in the App.

The recipe [Recipe 6.9](#) shows how to add a handler for button presses. Here instead of getting the `Main` class to implement the `onClick` method it will be done with an inner class.

Within `onClick` the code is needed to start the `Screen2` `Activity`. An `Intent` declaration requires a `Context` and `Activity` (`Screen2`). Since `Main` is an `Activity`, which is derived from `Context`, we can use `this` (in this case `Main.this` because of the inner class for the `onClick` handler). With all the imports the `Main.java` code will be:

Example 6-70.

```
package com.example;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;

public class Main extends Activity {
```

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    findViewById(R.id.nextButton).setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            Intent intent = new Intent(Main.this, Screen2.class);
            startActivity(intent);
        }
    });
}

```

Alternatively to make the code easier to understand the object to handle the button presses can be declared separately.

Example 6-71.

```

public class Main extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        findViewById(R.id.nextButton).setOnClickListener(new handleButton());
    }
    class handleButton implements OnClickListener {
        public void onClick(View v) {
            Intent intent = new Intent(Main.this, Screen2.class);
            startActivity(intent);
        }
    }
}

```

(The handler example in the [Recipe 6.9](#) recipe can also be adapted for this example).

Finally to register the new screen with Android an activity definition is add to the *AndroidManifest.xml* file in the project, after the activity declaration for *Main*. The activity section will be:

Example 6-72.

```

<application android:icon="@drawable/globe" android:label="@string/app_name">
    <activity android:name=".Main"
        android:label="@string/app_name">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <activity android:name=".Screen2"
        android:label="@string/screen2Title">
    </activity>
</application>

```


The dot in front of *Main* and *Screen2* signifies that the activity is within the application package. If the activity was defined in another package then the activity name would include the full package name.

When the App runs the first screen will show:

And pressing the Next button shows:

A button is not required to go back to the first screen. Android automatically provides a back button for all screens as part of the platform.

See Also

Hello World - Eclipse Version Invoke an action handler when a Button is pressed

6.26 Creating a Tabbed Dialog

Rachee Singh

Problem

Categorizing information display in an custom Dialog.

Solution

Using a tabbed layout within a custom Dialog.

Discussion

The custom dialog class implements the class Dialog.

Example 6-73.

```
public class CustomDialog extends Dialog
```

The constructor of the class has to be initialized.

Example 6-74.

```
public CustomDialog(final Context context)
{
    super(context);

    setTitle("My First Custom Tabbed Dialog");
    setContentView(R.layout.custom_dialog_layout);
}
```

To create the 2 tabs, insert this code within the constructor. `tab_image1` and `tab_image2` are placed in `/res/drawable`. These images are placed on the tabs of the tabbed custom dialog.

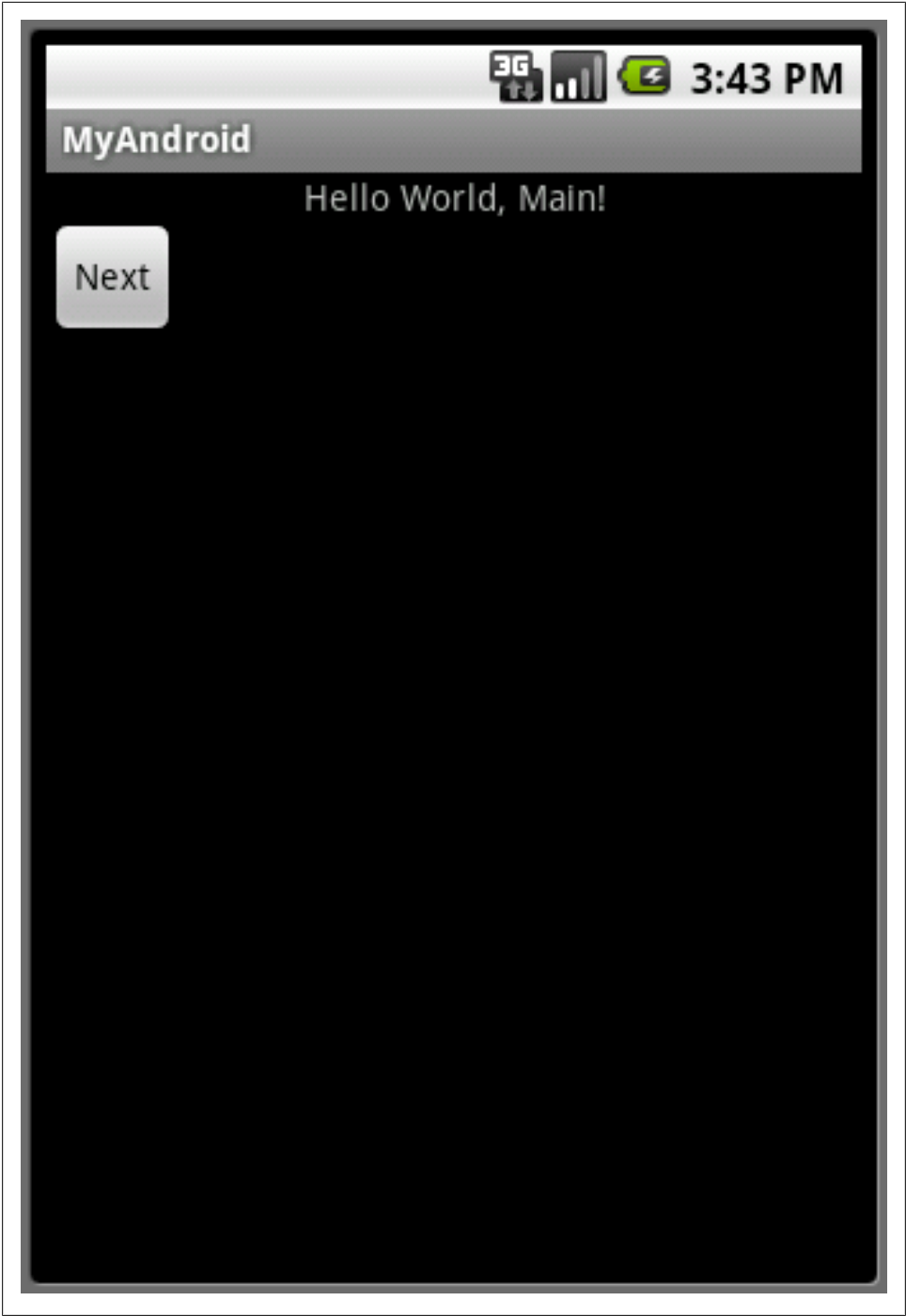


Figure 6-17.

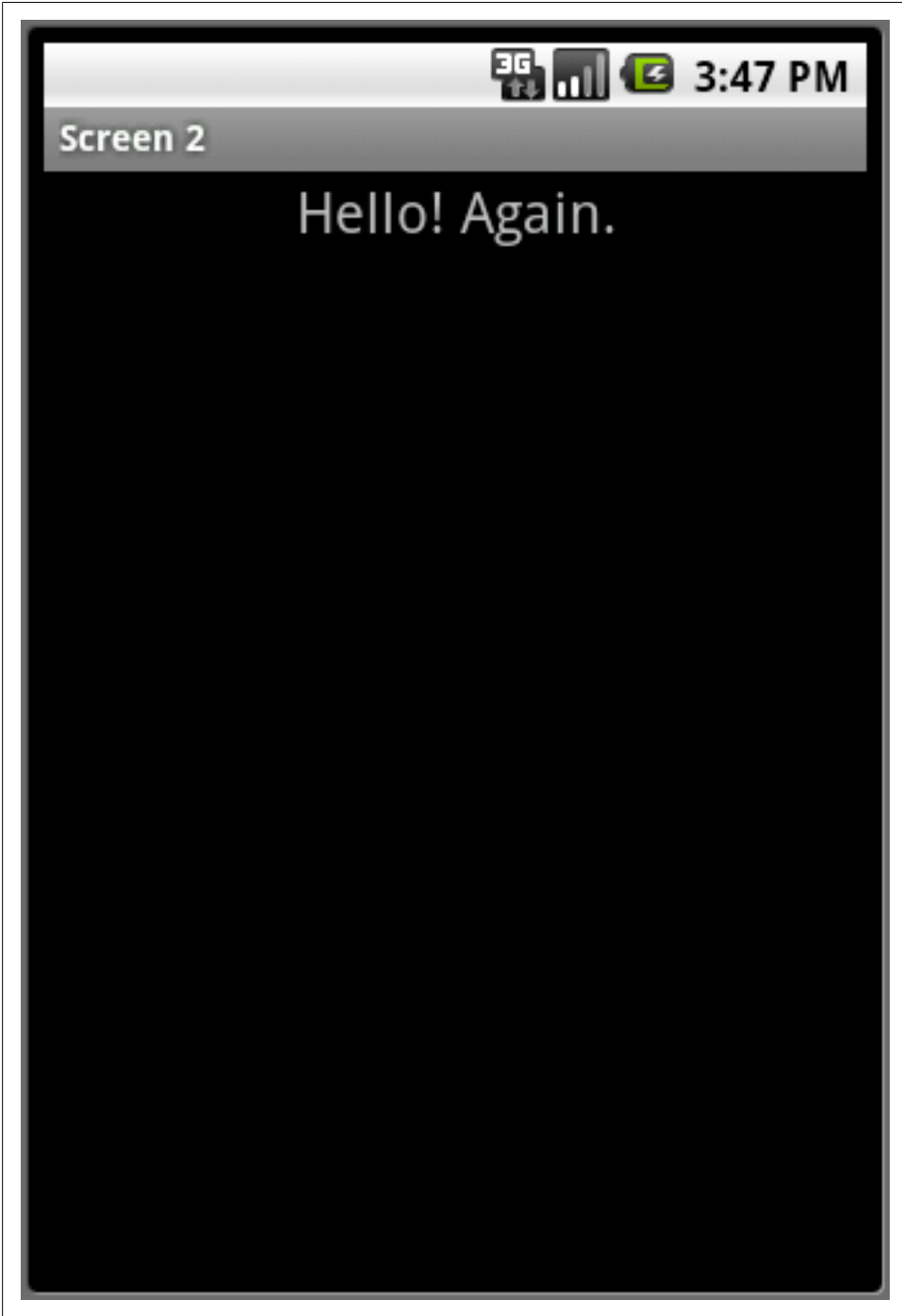


Figure 6-18.

Example 6-75.

```
// get our tabHost from the xml
TabHost tabHost = (TabHost)findViewById(R.id.TabHost01);
tabHost.setup();

// create tab 1
TabHost.TabSpec spec1 = tabHost.newTabSpec("tab1");
spec1.setIndicator("Profile", context.getResources().getDrawable(R.drawable.tab_image1));
spec1.setContent(R.id.TextView01);
tabHost.addTab(spec1);
//create tab2
TabHost.TabSpec spec2 = tabHost.newTabSpec("tab2");
spec2.setIndicator("Profile", context.getResources().getDrawable(R.drawable.tab_image2));
spec2.setContent(R.id.TextView02);
tabHost.addTab(spec2);
```

This is a simple tabbed dialog. It required the addition of these few lines into the constructor's code. To implement something like a list view, a list view adapter would be required. There are immense possibilities to the kind of tabs that can be inserted based on the requirement of the application.

The XML code for a tabbed dialog would require `<tabhost>` tags enclosing the entire layout. Within these tags, the location of various parts of the tabbed dialog are placed. A frame layout has to be used to place the content of the different tabs. In this case, we are creating 2 tabs both with a scroll view containing text (stored in Strings.xml, named 'lorem_ipsum').

Here is the code for custom_dialog_layout.xml

Example 6-76.

```
<TabHost
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/TabHost01"
    android:layout_width="fill_parent"
    android:layout_height="500dip">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:padding="5dp">

        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"/>

        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:padding="5dp">
```

```

<ScrollView android:id="@+id/ScrollView01"
    android:layout_width="wrap_content"
    android:layout_height="200px">

    <TextView
        android:id="@+id/TextView01"
        android:text="@string/lorem_ipsum"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:paddingLeft="15dip"
        android:paddingTop="15dip"
        android:paddingRight="20dip"
        android:paddingBottom="15dip"/>

</ScrollView>

<ScrollView android:id="@+id/ScrollView02"
    android:layout_width="wrap_content"
    android:layout_height="200px">

    <TextView
        android:id="@+id/TextView02"
        android:text="@string/lorem_ipsum"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:paddingLeft="15dip"
        android:paddingTop="15dip"
        android:paddingRight="20dip"
        android:paddingBottom="15dip"/>

</ScrollView>
</FrameLayout>
</LinearLayout>
</TabHost>

```

6.27 Creating a Custom Dialog with buttons, images and text

Rachee Singh

Problem

The application requires a Dialog-like structure in place of a full-fledged activity to show some information. Text, images and also a button are required on this Custom Dialog.

Solution

Creating a Custom Dialog with Tabs. Since everything can be squeezed into a Dialog in place of an entire activity, the application has seems more compact.

Discussion

Custom Dialog Class should extend 'Custom Dialog': public class CustomDialog extends Dialog

Following lines of code need to be added to the 'customDialog' class:

Example 6-77.

```
setTitle("Dialog Title");
setContentView(R.layout.custom_dialog_layout);
//On Click listeners for the buttons present in the Dialog
Button button1 = (Button) findViewById(R.id.button1);
Button button2 = (Button) findViewById(R.id.button2);
button1.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        dismiss(); //to dismiss the Dialog
    }
});

button2.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        //Fire an intent on click of this button
        Intent showQuickInfo = new Intent("com.android.oreilly.QuickInfo");
        showQuickInfo.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(showQuickInfo);
    }
});
```

The XML code in the layout file (present in /res/layout) custom_dialog_layout :

The entire code is enclosed with a linear layout. Within the LinearLayout, a Relative Layout is used to position 2 buttons (which could be denote different options like 'Login' and 'Vote up' etc). Then below the Relative layout, another Relative Layout containing a scroll view is present.

'android_button' and 'thumbsup' are the names of images in /res/drawable.

Example 6-78.

```
<LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:padding="5dp">

    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="10dip">
        <Button
```

```

        android:id="@+id/button1"
        android:background="@drawable/android_button"
        android:layout_height="80dip"
        android:layout_width="80dip"
        android:layout_alignParentLeft="true"
        android:layout_marginLeft="10dip"
        android:gravity="center"/>

    <Button
        android:id="@+id/button2"
        android:background="@drawable/thumbsup"
        android:layout_height="80dip"
        android:layout_width="80dip"
        android:layout_alignParentRight="true"
        android:layout_marginRight="10dip"
        android:gravity="center"/>
</RelativeLayout>

<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:paddingBottom="10dip">

    <ScrollView android:id="@+id/ScrollView01"
        android:layout_width="wrap_content"
        android:layout_height="200px">

        <TextView
            android:id="@+id/TextView01"
            android:text="@string/lorem"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal"
            android:paddingLeft="15dip"
            android:paddingTop="15dip"
            android:paddingRight="20dip"
            android:paddingBottom="15dip"/>

    </ScrollView>
</RelativeLayout>
</LinearLayout>

```

6.28 Create a Custom Menu

Rachee Singh

Problem

Within the Activity, showing a menu on pressing the Menu button of the Android device.

Solution

Implement a menu by setting it up in the XML and attaching it to your Activity by overriding `onCreateOptionsMenu()`.

Discussion

Its a two step process:

Step 1: Create a directory named 'menu' in the res directory of the project. In the menu directory create a menu.xml. Here is the code for menu.xml:

Example 6-79.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/icon1"
        android:title="One"
        android:icon="@drawable/first" />
  <item android:id="@+id/icon2"
        android:title="Two"
        android:icon="@drawable/second" />
  <item android:id="@+id/icon3"
        android:title="Three"
        android:icon="@drawable/three" />
  <item android:id="@+id/icon4"
        android:title="Four"
        android:icon="@drawable/four" />
</menu>
```

In this XML code we add a menu and to it we add as many items as our application requires. We can images (in this case default images have been added).

Step 2: In the java code for the activity, override the `onCreateOptionsMenu` :

Example 6-80.

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

This is how it looks:

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LNjjMzBhZjktMDcxZS00MWU4LWIzNWYtZWUxOTAzMzc2NjZk&hl=en_US&authkey=CPyM4boI

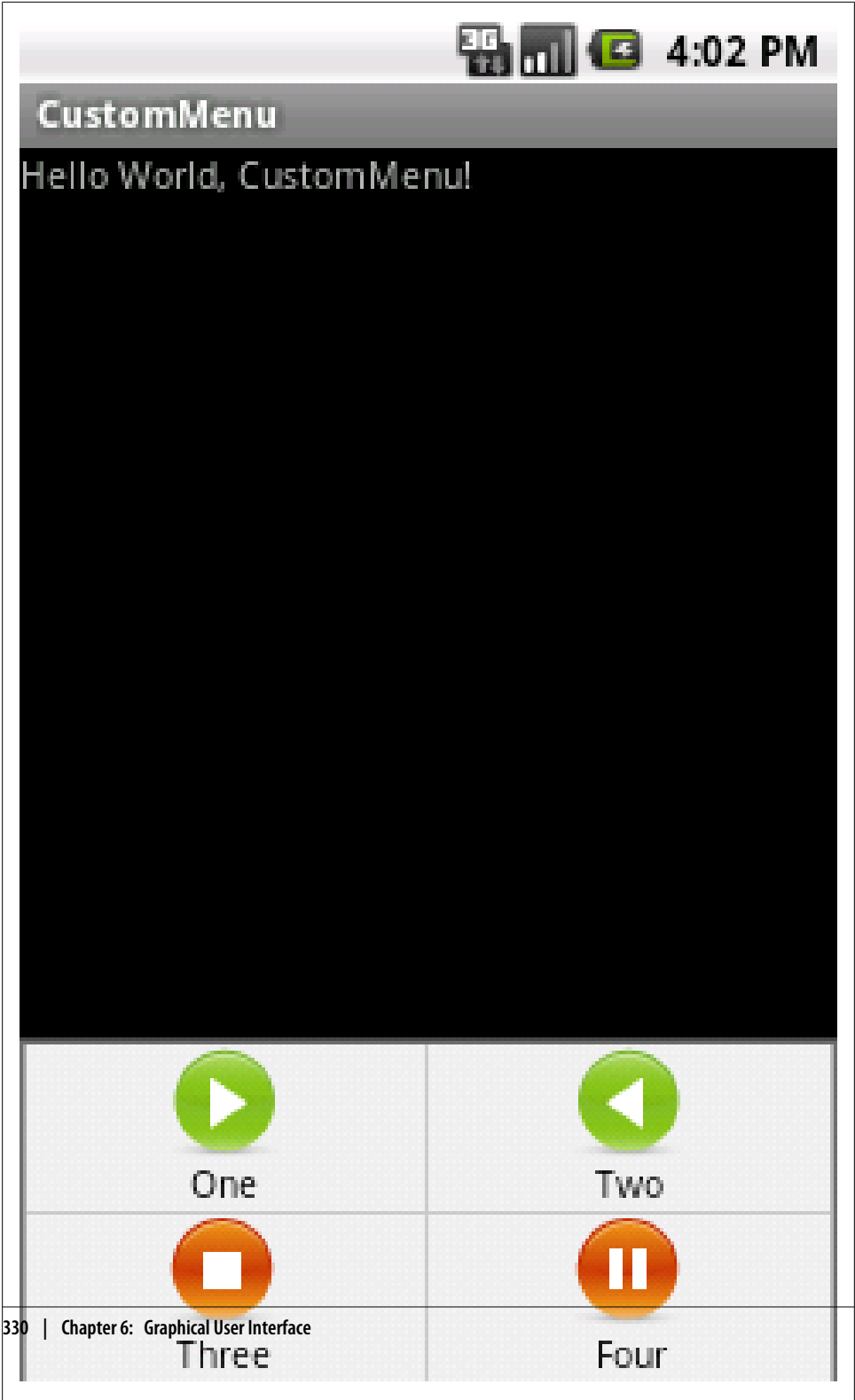


Figure 6-19.

6.29 Loading Screen in between two Activities

Shraddha Shrivagi

Problem

If you are getting an black screen before loading an activity here's how to avoid it.

Solution

Sometimes when your Activity is fetching some data from database or from internet then it takes time to load-up. To avoid this scenario we can have a simple Activity that shows a loading image instead of a black screen.

Your normal scenario may be like this: ProfileList(user selects one profile) -> Black Screen -> ProfileData Suppose you have an list activity (ProfileList) from which when user selects one then connection is established and later Profile(ProfileData) data is shown and if it takes time to load ProfileData then you introduce an LoadingScreen

ProfileList(user selects one profile) -> LoadingScreenActivity -> ProfileData

Discussion

1. Create a LoadingScreen layout file

Here you create an screen which just shows loading text and an progress bar loading_screen.xml

Example 6-81.

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"    android:gravity="center" android:orientation="vertical"
  android:layout_height="fill_parent"  android:background="#E5E5E5">

  <TextView android:text="Please wait while your data gets loaded..."
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:textColor="#000000">
  </TextView>
  <ProgressBar android:id="@+id/mainSpinner1" android:layout_gravity="center"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:indeterminate="true"
    style="?android:attr/progressBarStyleInverse">
  </ProgressBar>
</LinearLayout>
```

2. Create a LoadingScreen class file

Example 6-82.

```
public class LoadingScreenActivity extends Activity {  
  
    //Introduce an delay  
    private final int WAIT_TIME = 2500;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        // TODO Auto-generated method stub  
        super.onCreate(savedInstanceState);  
        System.out.println("LoadingScreenActivity screen started");  
        setContentView(R.layout.loading_screen);  
        findViewById(R.id.mainSpinner1).setVisibility(View.VISIBLE);  
  
        new Handler().postDelayed(new Runnable(){  
            @Override  
            public void run() {  
                //Simulating a long running task  
                this.Sleep(1000);  
                System.out.println("Going to Profile Data");  
                /* Create an Intent that will start the ProfileData-Activity. */  
                Intent mainIntent = new Intent(LoadingScreenActivity.this,ProfileData.class);  
                LoadingScreenActivity.this.startActivity(mainIntent);  
                LoadingScreenActivity.this.finish();  
            }  
        }, WAIT_TIME);  
    }  
}
```

This will load the next activity once WAIT_TIME is elapsed

3. Open LoadingScreenActivity from say your List from onItemClick event

Create an intent to launch loading screen activity

Example 6-83.

```
protected void onItemClick(ListView l, View v, int position, long id) {  
    super.onItemClick(l, v, position, id);  
  
    Intent intent = new Intent(ProfileList.this, LoadingScreenActivity.class);  
    startActivity(intent);  
}
```

This is how you show an simple loading screen for 2500ms. After the completion of these many seconds automatically the next activity that you mentioned in the handler of LoadingScreenActivity will start. Thus the black screen which is displayed when you do some web activity can be avoided.

6.30 Implementing reactions on click of items in a Custom Menu.

Rachee Singh

Problem

After Creating a custom menu, implementing a reaction on clicking the menu item.

Solution

Overriding onOptionsItemSelected method.

Discussion

In the Java Activity we need to override the onOptionsItemSelected. This method takes in a MenuItem and checks for its ID. Based on the ID of the item which is clicked, a switch-case can be used. Depending on the case selected, appropriate action can be taken. The custom menu would look something like this:

For a sample, the cases can just display Toasts.

Here's the source code for the sample:

Example 6-84.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.icon1:
            Toast.makeText(this, "Icon 1 Beep Bop!", Toast.LENGTH_LONG).show();
            break;
        case R.id.icon2:
            Toast.makeText(this, "Icon 2 Beep Bop!", Toast.LENGTH_LONG).show();
            break;
        case R.id.icon3:
            Toast.makeText(this, "Icon 3 Beep Bop!", Toast.LENGTH_LONG).show();
            break;
        case R.id.icon4 :
            Toast.makeText(this, "Icon 4 Beep Bop!", Toast.LENGTH_LONG).show();
            break;
    }
    return true;
}
```

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LZWM0ODRiNjAtNzJhOS00MGRjLTkwMjMtMjNlOTQwZDU0OGE2&hl=en_US&authkey=CJKD4IoH

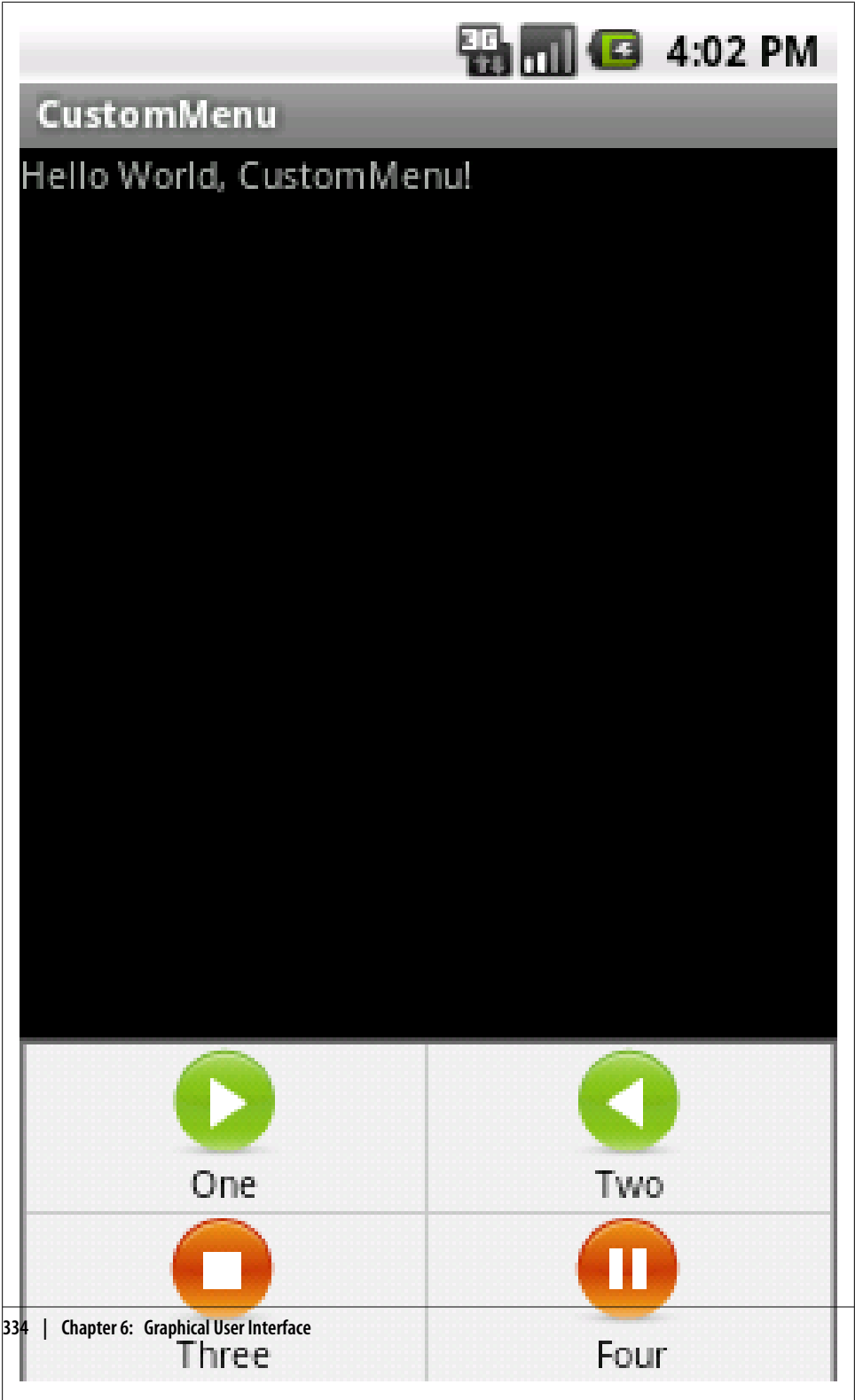


Figure 6-20.



11:05 AM

CustomMenu

Hello World, CustomMenu!

Icon 1 Beep Bop!

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LNjjjMzBhZjktMDcxZS00MWU4LWlZnWYtZWUxOTAzMzc2NjZk&hl=en_US&authkey=CPyM4boI

6.31 Navigate different activities within a TabView

Pratik Rupwal

Problem

You want to move from an activity within a tab view to another activity within the same tab.

Solution

It can be achieved by replacing the content view of the tab by the new activity you want to move to.

Discussion

When an activity (lets call it as 'calling activity' here onwards) within a 'TabView' calls another activity (lets call it as 'called activity' here onwards) through an intent, the 'TabView' gets replaced by the view of called activity. To show the called activity within the 'TabView' we can replace the view of calling activity by the view of called activity so the TabView remains stable. To achieve this the calling activity needs to be extended from 'ActivityGroup' rather than 'Activity'.

Below activity 'Calling' extended from 'ActivityGroup' has been set within a 'TabView'.

Example 6-85.

// 'Calling' activity.

```
public class Calling extends ActivityGroup implements OnClickListener
{
    Button b1;
    Intent i1;

    /** Called when the activity is first created.*/
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.calling);

        b1=(Button)findViewById(R.id.changeactivity);
    }
}
```

```

        b1.setOnClickListener();
    }

    public void onClick(View view)
    {
        // This creates an intent to call the 'Called' activity

        i1=new Intent(this.getContext(),Called.class);

        // calls the method to replace View.

        replaceContentView("Called", i1);
    }

    // This method is used for replacing the view of 'Calling' activity by 'Called' activity.

    public void replaceContentView(String id, Intent newIntent)
    {
        //Obtain the view of 'Called' activity using its Intent 'newIntent'
        View view = getLocalActivityManager().startActivity(id,newIntent.addFlags(Intent.FLAG_ACTIVITY_CLEAR

        //set the above view to the content of 'Calling' activity.
        this.setContentview(view);
    }
}

```

The 'Called activity' can also call another activity (say 'CalledSecond') as below.

```

//'Called activity'

public class Called extends Activity implements OnClickListener
{
    Button b1;
    Intent i1;
    Calling caller;

    /** Called when the activity is first created.*/
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.called);

        b1=(Button)findViewById(R.id.changeactivity);

        b1.setOnClickListener();
    }

    public void onClick(View view)
    {
        // This creates an intent to call the 'CalledSecond' activity

        i1=new Intent(this.getContext(),CalledSecond.class); /** 'CalledSecond' can be any activity,
                                                                    even the 'Calling'(In case backward navigation is

```



```

        // Initialize the object of the 'Calling' class.
        caller=(Calling)getParent();

        // calls the method to replace View.

        caller.replaceContentView("CalledSecond", i1);
    }
}

```

6.32 Drop-down Chooser via the Spinner Class

Ian Darwin

Problem

You want to offer a drop-down choice item.

Solution

Use a Spinner object; you can pass the list of selections as an Adapter.

Discussion

Generally known as a 'combo box', the Spinner is the analog of the HTML *SELECT* or the Swing JComboBox. It provides a drop-down chooser whose values appear to float over the screen when the Spinner is clicked. One item can be selected and the floating version will pop down, displaying the selection in the Spinner.

Like all standard components, the Spinner can be created and customized in XML. In this example - from healthcare practice - the term "Context" is used to indicate the phase of the person's day when a reading was taken - after breakfast, after lunch, etc., so the health care practitioner can understand the value *in context* of the patient's day. Here is an excerpt from res/layout/main.xml:

Example 6-86.

```

<Spinner android:id="@+id/contextChooser"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:prompt="@string/context_choice"/>

```

Ideally the list of values won't be hard-coded but will come from a Resource file, so as to be internationalizable. Here is a file res/values/contexts.xml containing the XML values for the list of times to choose.

Example 6-87.

```

<?xml version="1.0" encoding="utf-8"?>
<resources>

```

```

        <string name="context_choice">When Reading Taken</string>
    <string-array name="context_names">
        <item>Breakfast</item>
        <item>Lunch</item>
        <item>Dinner</item>
        <item>Snack</item>
    </string-array>
</resources>

```

To tie the list of Strings to the Spinner at run time, just locate the Spinner and set the values, as shown:

Example 6-88.

```

Spinner contextChooser = (Spinner) findViewById(R.id.contextChooser);
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(
    this, R.array.context_names, android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(
    android.R.layout.simple_spinner_dropdown_item);
contextChooser.setAdapter(adapter);

```

That is all you need in order for the Spinner to appear, and to allow the user to select items. If you want to know the chosen value right away, you can send an instance of `OnItemSelectedListener` to the Spinner's `setOnItemSelectedListener`. This interface has two callback methods, `setItemSelected` and `setNothingSelected`. Both are called with the Spinner (but the argument is declared as a `ViewAdapter`); the former method is also called with two integer arguments, the list position and the identity of the selected item.

Example 6-89.

```

contextChooser.setOnItemSelectedListener(new OnItemSelectedListener() {

    @Override
    public void onItemSelected(AdapterView<?> spinner, View arg1,
        int pos, long id) {
        Toast.makeText(SpinnerDemoActivity.this, "You selected " + contextChooser.getSelectedItem(),
    }

    @Override
    public void onNothingSelected(AdapterView<?> spinner) {
        Toast.makeText(SpinnerDemoActivity.this, "Nothing selected.", Toast.LENGTH_LONG).show();
    }
});

```

On the other hand, you may not need the value from the Spinner until the user fills in multiple items and clicks a Button. In this case, you can simply call the Spinner's `getSelectedItem()` method, which returns the item place in that position by the Adapter. Assuming you placed Strings in the list, you can just call `toString()` to get back the given String value.

6.33 Effective UI design using Image Buttons

Rachee Singh

Problem

While building an application, many a times the UI design requires the effective use of image buttons. It saves effort on text views etc since an image explains the scenario much better.

Solution

This application provides the user with 2 options: 1. To start the application (which could be, start bluetooth scan or play music, GPS scan etc) 2. Configure the application (Complete settings for the application). So here is an intuitive UI with 2 image buttons providing these choices.

Discussion

Making one's own Image Buttons requires defining the characteristics of the button as an XML file that should be placed in /res/drawable. This XML specifies the 3 states of an Image Button:

1. Pressed State
2. Focused State
3. Otherwise.

For Instance:

Example 6-90.

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:drawable="@drawable/play_pressed"
        android:state_checked="true" />
    <item android:drawable="@drawable/play" />
</selector>
```

When Play Button is not pressed:

When Play Button is pressed:

So, for each of these states, the ID of an image is specified (the image present in /res/drawable as a .png). When the button is pressed, the play_pressed image is displayed. There are 2 such buttons in the application, Play Button and the Settings Button. In the .java file of the application, onClick aspect of the buttons can be taken care of. In this recipe, merely a toast is displayed with some appropriate text. Programmers can kick off an new activity from here or broadcast an intent and many other things based on their requirement.



Figure 6-22.

BlueLink: Theft Prevention



Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LYTVjZGMzZmItNDYzNC00YmRmLTlkMTktOTIzNTM0NzVmMDQ2&hl=en_US

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LOWYyODVlMTAtNzNiNy00YzlmLWlxMjUtNmViNjAxMjFkYmFh&hl=en_US

6.34 Pinch to zoom

Pratik Rupwal

Problem

Use the finger touch to change the position of an image viewed on the screen and use pinch-in and pinch-out movements for zooming in and zooming out operations.

Solution

Scale the image as a matrix to apply transformations to it for showing different visual effects.

Discussion

A simple ImageView is to be added inside a FrameLayout in main.xml as below:

Example 6-91.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
<ImageView android:id="@+id/imageView"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:src="@drawable/nature"
    android:scaleType="matrix" >
</ImageView>
</FrameLayout>
```

The code below scales the ImageView as a matrix to apply transformations on it.

Example 6-92.

```
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.Matrix;
import android.graphics.PointF;
import android.os.Bundle;
import android.util.FloatMath;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.GridView;
import android.widget.ImageView;

public class Touch extends Activity implements OnTouchListener {
    private static final String TAG = "Touch";

    // These matrices will be used to move and zoom image
    Matrix matrix = new Matrix();
    Matrix savedMatrix = new Matrix();

    // We can be in one of these 3 states
    static final int NONE = 0;
    static final int DRAG = 1;
    static final int ZOOM = 2;
    int mode = NONE;

    // Remember some things for zooming
    PointF start = new PointF();
    PointF mid = new PointF();
    float oldDist = 1f;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        ImageView view = (ImageView) findViewById(R.id.imageView);
        view.setScaleType(ImageView.ScaleType.FIT_CENTER); // make the image fit to the center.
        view.setOnTouchListener(this);
    }

    public boolean onTouch(View v, MotionEvent event) {
        ImageView view = (ImageView) v;
        // make the image scalable as a matrix
        view.setScaleType(ImageView.ScaleType.MATRIX);
        float scale;

        // Handle touch events here...
        switch (event.getAction() & MotionEvent.ACTION_MASK) {

            case MotionEvent.ACTION_DOWN: //first finger down only
                savedMatrix.set(matrix);
                start.set(event.getX(), event.getY());
                Log.d(TAG, "mode=DRAG" );
                mode = DRAG;
            case MotionEvent.ACTION_MOVE:
                float dx = event.getX() - start.getX();
                float dy = event.getY() - start.getY();
                float oldMatrix[] = savedMatrix.values();
                float newMatrix[] = new float[9];
                newMatrix[0] = oldMatrix[0] + dx;
                newMatrix[1] = oldMatrix[1] + dy;
                newMatrix[2] = oldMatrix[2];
                newMatrix[3] = oldMatrix[3];
                newMatrix[4] = oldMatrix[4];
                newMatrix[5] = oldMatrix[5];
                newMatrix[6] = oldMatrix[6];
                newMatrix[7] = oldMatrix[7];
                newMatrix[8] = oldMatrix[8];
                savedMatrix.set(newMatrix);
                scale = savedMatrix.getScale();
                view.setScaleType(ImageView.ScaleType.MATRIX);
                break;
            case MotionEvent.ACTION_UP:
                savedMatrix.set(matrix);
                mode = NONE;
                break;
        }
    }
}
```

```

        break;
    case MotionEvent.ACTION_UP: //first finger lifted
    case MotionEvent.ACTION_POINTER_UP: //second finger lifted
        mode = NONE;
        Log.d(TAG, "mode=NONE" );
        break;
    case MotionEvent.ACTION_POINTER_DOWN: //second finger down
        oldDist = spacing(event); // calculates the distance between two points where user touched.
        Log.d(TAG, "oldDist=" + oldDist);
        // minimal distance between both the fingers
        if (oldDist > 5f) {
            savedMatrix.set(matrix);
            midPoint(mid, event); // sets the mid-point of the straight line between two points where user touched
            mode = ZOOM;
            Log.d(TAG, "mode=ZOOM" );
        }
        break;

    case MotionEvent.ACTION_MOVE:
        if (mode == DRAG)
        { //movement of first finger
            matrix.set(savedMatrix);
            if (view.getLeft() >= -392)
            {
                matrix.postTranslate(event.getX() - start.x, event.getY() - start.y);
            }
        }
        else if (mode == ZOOM) { //pinch zooming
            float newDist = spacing(event);
            Log.d(TAG, "newDist=" + newDist);
            if (newDist > 5f) {
                matrix.set(savedMatrix);
                scale = newDist/oldDist; //thinking I need to play around with this value to limit it**
                matrix.postScale(scale, scale, mid.x, mid.y);
            }
        }
        break;
    }

    // Perform the transformation
    view.setImageMatrix(matrix);

    return true; // indicate event was handled
}

private float spacing(MotionEvent event) {
    float x = event.getX(0) - event.getX(1);
    float y = event.getY(0) - event.getY(1);
    return FloatMath.sqrt(x * x + y * y);
}

private void midPoint(PointF point, MotionEvent event) {
    float x = event.getX(0) + event.getX(1);
    float y = event.getY(0) + event.getY(1);
    point.set(x / 2, y / 2);
}

```



```
}  
}
```

6.35 Add a Border with Rounded Corners to a Layout

Daniel Fowler

Problem

There is a need to put a border around an area of the screen or add interest to a user interface.

Solution

Define an Android shape in an XML file and assign it to a layout's background attribute.

Discussion

The `drawable` folder, under `res`, in an Android project is not restricted to bitmaps (PNG or JPG files) but can also hold shapes defined in XML files. These shapes can then be reused in the project. A shape can be used to put a border around a layout. This example shows a rectangular border with curved corners.

A new file called `customborder.xml` is created in the `drawable` folder (in Eclipse use the `File` menu and select `New` then `File`, with the `drawable` folder selected type in the file name and click `Finish`).

The XML defining the border shape is entered:

Example 6-93.

```
<?xml version="1.0" encoding="UTF-8"?>  
<shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle">  
    <corners android:radius="20dp"/>  
    <padding android:left="10dp" android:right="10dp" android:top="10dp" android:bottom="10dp"/>  
    <solid android:color="#CCCCCC"/>  
</shape>
```

The attribute `android:shape` is set to **rectangle** (shape files also support **oval**, **line** and **ring**). Rectangle is the default value so this attribute could be left out if it is a rectangle being defined. See the Android documentation on shapes for detailed information on a shape file (link below).

The element `corners` sets the rectangle corners to be rounded, it is possible to set a different radius on each corner (see the Android reference).

The `padding` attributes are used to move the contents of the View to which the shape is applied, to prevent the contents overlapping the border.

The border color here is set to a light gray (CCCCCC hexadecimal RGB value).

Shapes also support gradients but that is not being used here, again see the Android resources to see how a gradient is defined.

The shape is applied to the layout using `android:background="@drawable/customborder"`.

Within the layout other views can be added as normal, in this example a single `TextView` has been added, the text is white (FFFFFF hexadecimal RGB). The background is set to blue, plus some transparency to reduce the brightness (A00000FF hexadecimal alpha RGB value).

Finally the layout is offset from the screen edge by placing it into another layout with a small amount of padding. The full layout file is thus:

Example 6-94.

```
<?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"
    android:padding="5dp">
    <LinearLayout android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="@drawable/customborder">
        <TextView android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:text="Text View"
            android:textSize="20dp"
            android:textColor="#FFFFFF"
            android:gravity="center_horizontal"
            android:background="#A00000FF" />
    </LinearLayout>
</LinearLayout>
```

Which produces:

See Also

<http://developer.android.com/guide/topics/resources/drawable-resource.html#Shape>

6.36 Creating a ProgressDialog in Android.

Rachee Singh

Problem

Making the user aware of background processing happening in the application.

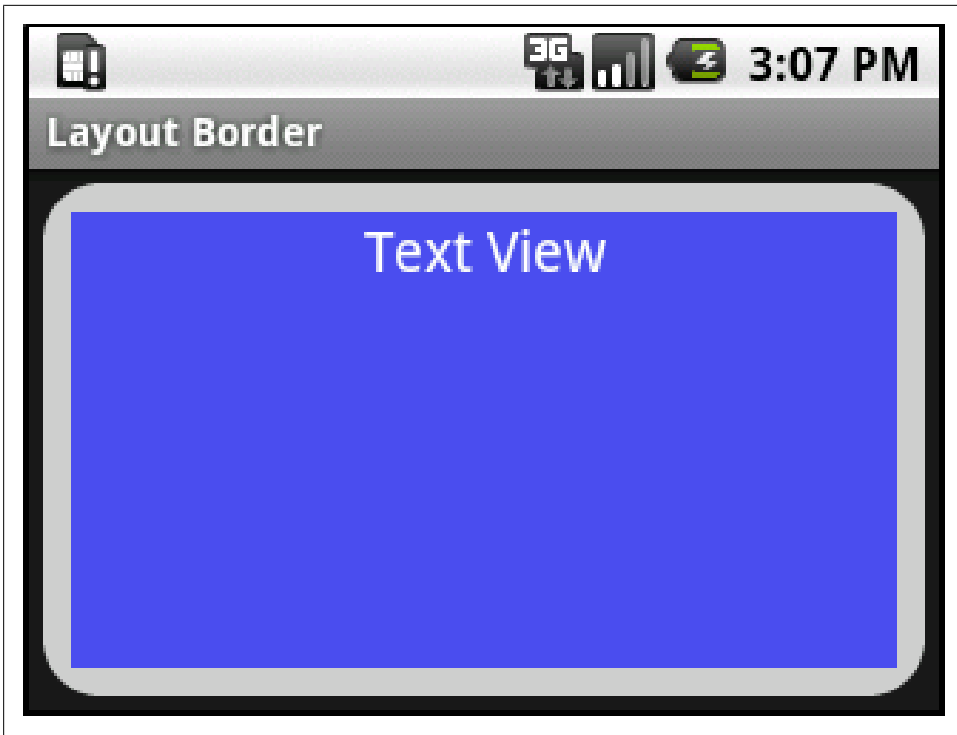


Figure 6-24.

Solution

Show a ProgressDialog while the processing is being carried out.

Discussion

We provide a button that on being clicked shows a ProgressDialog. In the ProgressDialog we set the title as 'Please Wait' and content as 'Processing Information..'. After this we create a new thread and start the thread's execution. In the run() method (which gets executed once the thread gets started) we call the sleep method for 4 seconds. After these 4 seconds expire the ProgressDialog is dismissed and the text in TextView gets changed.

Example 6-95.

```
complete = (TextView) this.findViewById(R.id.complete);
complete.setText("Press the Button to start Processing");
processing = (Button)findViewById(R.id.processing);
processing.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View arg0) {
        progressDialog = ProgressDialog.show(ProgressDialogExp.this, "Please Wait", "Processing Information.
```

```

        Thread thread = new Thread(ProgressDialogExp.this);
        thread.start();
    }
});

```

A Handler is used so as to update the UI once thread execution finishes. So we send an empty message to the handler after thread execution completes and then in the Handler we dismiss the ProgressDialog and update the text of the TextView.

Example 6-96.

```

public void run() {
    try {
        Thread.sleep(4000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    handler.sendMessage(0);
}

private Handler handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        progressDialog.dismiss();
        complete.setText("Processing Finished");
    }
};

```

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LMTE2NDcyMDEtNGMzMS00MzI4LTgyNGUtNzliZmY4ZjhhOWE2&hl=en_US

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LNjg2Y2UzZTEtOGY2YS00NGMyLTkzMzMtN2EzMTY0M2NmYTE0&hl=en_US

6.37 Creating a Submenu.

Rachee Singh

Problem

Display options to the user in an floating window.

Solution

Use a Submenu implementation and provide options to the user.

Discussion

A submenu is a part of a menu that displays options in a hierarichal manner as opposed the style shown in a custom menu otherwise. The menus can be created in 2 ways: # By inflating an XML layout # Creating the menu items in the Java code. Following the second approach in this recipe, the creating menu/submenu items is done in the on-CreateOptionsMenu() method.

We add the submenu to the menu using the addSubMenu() method. In order to prevent conflicts with other items in the menu, we explicitly provide group Id and item ID to the submenu we are creating (Constants for the item IDs and Group ID are specified). Then we set an icon for the submenu with setIcon method and an icon for the header of the Submenu.

To add items to the submenu we use the add() method. As arguments to the method, the group ID, item ID, the posiiton of the item in the submenu and the text associated with each item is specified.

Example 6-97.

```
private static final int OPTION_1 = 0;
private static final int OPTION_2 = 1;
private int GROUP_ID = 4;
private int ITEM_ID =3;
```

Example 6-98.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {

    SubMenu sub1 = menu.addSubMenu(GROUP_ID, ITEM_ID , Menu.NONE, R.string.submenu_1);
    sub1.setHeaderIcon(R.drawable.icon);
    sub1.setIcon(R.drawable.icon);

    sub1.add(GROUP_ID , OPTION_1, 0, "Submenu Option 1");
    sub1.add(GROUP_ID, OPTION_2, 1, "Submenu Option 2");

    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case OPTION_1:
            Toast.makeText(this, "Submenu 1, Option 1", Toast.LENGTH_LONG).show();
            break;
        case OPTION_2:
            Toast.makeText(this, "Submenu 1, Option 2", Toast.LENGTH_LONG).show();
            break;
    }
    return true;
}
```

onOptionsItemSelected() method is called when an item of the menu/submenu is selected. In this method, using switch-case we check for the item that is clicked and an appropriate message is displayed.

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LN2I5ZmIxNjEtYzc3Zi00MjczLTk5NzEtYmZjNzRlNjM1ZTc2&hl=en_US&authkey=CN-BsekI

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LNTQyYzU0ZjEtMmY4Ny00YmQwLTg4ZTYtY2I0OWFIM2E1ZjRm&hl=en_US&authkey=CLTs67UG

6.38 Processing key press events in an Activity.

Rachee Singh

Problem

Intercepting the keys pressed by the user and performing actions corresponding to them.

Solution

This can be achieved by overriding the onKeyDown method in an Activity.

Discussion

If the application requires to react differently at different key presses, then in the Activity's Java code, the onKeyDown method needs to be overridden. This method takes the KeyCode as an argument, so within a switch-case block different actions can be carried out.

Example 6-99.

```
public boolean onKeyDown(int keyCode, KeyEvent service) {
    switch(keyCode) {
        case KeyEvent.KEYCODE_HOME:
            keyType.setText("Home Key Pressed!");
            break;
        case KeyEvent.KEYCODE_DPAD_CENTER :
            keyType.setText("Center Key Pressed!");
            break;
        case KeyEvent.KEYCODE_DPAD_DOWN :
            keyType.setText("Down Key Pressed!");
            break;
    }
}
```

```
    //and so on..  
  }  
}
```

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LMDdhMDllymYtOWE5Mi00MDU0LWE4YWEtODkwNGYwMWVvOTNI&hl=en_US

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LODAyODlIMmYtYTliYS00ZTc5LWE1OGItODE1MmMwMDliZjcx&hl=en_US

6.39 Constrain EditText Values with Attributes and the TextWatcher Interface

Daniel Fowler

Problem

There is a need to limit the range and type of values being input.

Solution

Use appropriate attributes on the EditText Views in the layout XML and enhance it by implementing the TextWatcher interface.

Discussion

When an Application needs input from a user sometimes only a specific type of value is required; maybe a whole number, a decimal number, a number between two values or words that are capitalized. When defining an EditText in a layout attributes such as `android:inputType` can be used to constrain what the user is able to type. This automatically reduces the amount of code required later on because there are fewer checks to perform on the data that was entered. The `TextWatcher` interface is also useful for restricting values. In the following example an EditText only allows a value between 0 and 100, for example to represent a percentage. There is no code need to check the value because it is all done as the user types.

Here a simple layout has one EditText.

Example 6-100.

```
<?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">
<EditText android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/percent"
        android:text="0"
        android:maxLength="3"
        android:inputType="number"/>
</LinearLayout>

```

It is given a starting value of zero with `android:text="0"`, the number of characters that can be typed has been limited to three with `android:maxLength="3"` because the largest number we need, 100, only has three digits. Finally the user is restricted to only positive numbers with `android:inputType="number"`.

Within the Activity class an inner class is used to implement the `TextWatcher` interface (the Activity itself could be used to implement the interface). The `afterTextChanged()` method is overridden and will be called when the text changes as the user types. In this method the value being typed is checked to see if it is greater than 100. If so it is set to 100. There is no need to check for values below zero because they cannot be entered, because of the XML attributes. The `try` catch is needed for when all the numbers are deleted in which case the test for values above 100 would throw an exception (trying to parse an empty string).

`TextWatcher` also has a `beforeTextChanged()` and `onTextChanged()` method to be overridden but they are not used in this example.

Example 6-101.

```

class CheckPercentage implements TextWatcher{
    @Override
    public void afterTextChanged(Editable s) {
        try {
            Log.d("Percentage", "input: " + s);
            if(Integer.parseInt(s.toString())>100)
                s.replace(0, s.length(), "100");
        }
        catch(NumberFormatException nfe){}
    }
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
        // Not used, details on text just before it changed
        // used to track in detail changes made to text, e.g. implement an undo
    }
    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        // Not used, details on text at the point change made
    }
}

```

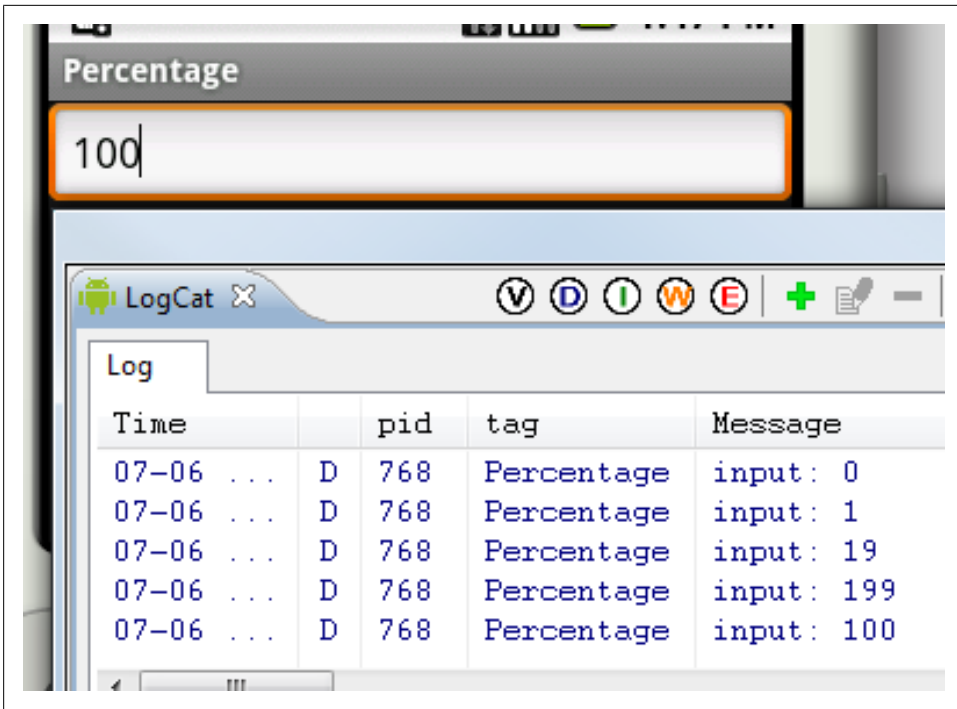



Figure 6-25.

Finally in the `onCreate()` for the Activity the class implementing `TextWatcher` is connected to the `EditText` using its `addTextChangedListener()` method.

Example 6-102.

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    EditText percentage=(EditText) findViewById(R.id.percent);
    percentage.addTextChangedListener(new CheckPercentage());
}
```

Note that it is fine to change the `EditText` value in `afterTextChanged()` as it's internal `Editable` class is passed in. However, it cannot be changed by altering the `CharSequence` passed into `beforeTextChanged()` and `onTextChanged()`.

For further details on the attributes supported by `EditText` see the Android documentation on the `TextView`, from which `EditText` is subclassed.

Also remember that changing the value in the `EditText` causes the `afterTextChanged()` method to be called again. Care must be taken to ensure that the code using a `TextWatcher` does not result in endless looping.

It is a good idea to review the attributes that Android views support, as defining them in the XML layout can reduce the amount of code to write.

See Also

<http://developer.android.com/reference/android/widget/TextView.html>

<http://developer.android.com/reference/android/widget/EditText.html>

<http://developer.android.com/reference/android/text/TextWatcher.html>

6.40 Gesture Detection in Android

Pratik Rupwal

Problem

You want to traverse through different screens using simple gestures like flip/scroll the page.

Solution

In Android we can detect simple gestures using the 'GestureDetector' class. This class can be used to detect simple gestures like tap, scroll, swipe or flip, etc.

Discussion

The application has 4 views and each view has different color. It has and 2 modes, SCROLL mode and FLIP mode. The application starts in FLIP mode. In

this mode when you perform the swipe/fling gesture in left right, up and down direction, the view changes back and forth. When a long-press is detected, the application changes to SCROLL mode, in this mode you scroll the displayed view. While in this mode, you can double-tap on the screen to bring back the screen to its original position. Again when a long-press is detected the application changes to FLIP mode.

This recipe focuses on gesture detection hence the animation applied is not discussed briefly. Refer to 'android.view.animation.*' for animation.

This will give you an introduction to simple gesture detection in Android.

Example 6-103.

```
import java.util.ArrayList;
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Vibrator;
import android.view.GestureDetector;
import android.view.Gravity;
import android.view.MotionEvent;
```

```

import android.view.ViewGroup.LayoutParams;
import android.view.animation.Animation;
import android.view.animation.OvershootInterpolator;
import android.view.animation.TranslateAnimation;
import android.widget.TextView;
import android.widget.ViewFlipper;

/**
 * GestureDetector class detects gestures using the supplied MotionEvent class.
 * We use this class along with the onTouchEvent, inside this method we call the
 * GestureDetector.onTouchEvent. GestureDetector identify the gestures or events
 * that occurred and report back to us using GestureDetector.OnGestureListener
 * callback interface. We create an instance of the GestureDetector class by
 * passing Context and GestureDetector.OnGestureListener listener. Double-tap
 * event is not present in theGestureDetector.onGestureListener callback
 * interface, this event is reported using another callback interface
 * GestureDetector.onDoubleTapListener. To use this callback interface we have
 * to register for these events using GestureDetector.setOnDoubleTapListener.
 * The MotionEvent class contains all the values correspond to a movement and
 * touch event. This class holds values such as X and Y position at which the
 * event occurred, timestamp at which the event occurred, mouse pointer index,
 * etc.
 */
public class FlipperActivity extends Activity implements GestureDetector.OnGestureListener,GestDetector.OnGestureListener {

    final private int SWIPE_MIN_DISTANCE = 100;
    final private int SWIPE_MIN_VELOCITY = 100;

    private ViewFlipper flipper = null;
    private ArrayList<TextView> views = null;
    private GestureDetector gesturedetector = null;
    private Vibrator vibrator = null;
    int colors[] = { Color.rgb(255,128,128),
        Color.rgb(128,255,128),
        Color.rgb(128,128,255),
        Color.rgb(128,128,128) };

    private Animation animleftin = null;
    private Animation animleftout = null;

    private Animation animrightin = null;
    private Animation animrightout = null;

    private Animation animupin = null;
    private Animation animupout = null;

    private Animation animdownin = null;
    private Animation animdownout = null;

    private boolean isDragMode = false;
    private int currentview = 0;

    /** Initializes the first screen and animation to be applied to the screen after detecting the gesture */

    @Override

```

```

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    flipper = new ViewFlipper(this);
    gesturedetector = new GestureDetector(this, this);
    vibrator = (Vibrator) getSystemService(VIBRATOR_SERVICE);
    gesturedetector.setOnDoubleTapListener(this);

    flipper.setInAnimation(animleftin);
    flipper.setOutAnimation(animleftout);
    flipper.setFlipInterval(3000);
    flipper.setAnimateFirstView(true);

    prepareAnimations();
    prepareViews();
    addViews();
    setViewText();

    setContentView(flipper);
}

private void prepareAnimations() {
    animleftin = new TranslateAnimation(
        Animation.RELATIVE_TO_PARENT, +1.0f, Animation.RELATIVE_TO_PARENT, 0.0f,
        Animation.RELATIVE_TO_PARENT, 0.0f, Animation.RELATIVE_TO_PARENT, 0.0f);

    animleftout = new TranslateAnimation(
        Animation.RELATIVE_TO_PARENT, 0.0f, Animation.RELATIVE_TO_PARENT, -1.0f,
        Animation.RELATIVE_TO_PARENT, 0.0f, Animation.RELATIVE_TO_PARENT, 0.0f);

    animrightin = new TranslateAnimation(
        Animation.RELATIVE_TO_PARENT, -1.0f, Animation.RELATIVE_TO_PARENT, 0.0f,
        Animation.RELATIVE_TO_PARENT, 0.0f, Animation.RELATIVE_TO_PARENT, 0.0f);

    animrightout = new TranslateAnimation(
        Animation.RELATIVE_TO_PARENT, 0.0f, Animation.RELATIVE_TO_PARENT, +1.0f,
        Animation.RELATIVE_TO_PARENT, 0.0f, Animation.RELATIVE_TO_PARENT, 0.0f);

    animupin = new TranslateAnimation(
        Animation.RELATIVE_TO_PARENT, 0.0f, Animation.RELATIVE_TO_PARENT, 0.0f,
        Animation.RELATIVE_TO_PARENT, +1.0f, Animation.RELATIVE_TO_PARENT, 0.0f);

    animupout = new TranslateAnimation(
        Animation.RELATIVE_TO_PARENT, 0.0f, Animation.RELATIVE_TO_PARENT, 0.0f,
        Animation.RELATIVE_TO_PARENT, 0.0f, Animation.RELATIVE_TO_PARENT, -1.0f);

    animdownin = new TranslateAnimation(
        Animation.RELATIVE_TO_PARENT, 0.0f, Animation.RELATIVE_TO_PARENT, 0.0f,
        Animation.RELATIVE_TO_PARENT, -1.0f, Animation.RELATIVE_TO_PARENT, 0.0f);

    animdownout = new TranslateAnimation(
        Animation.RELATIVE_TO_PARENT, 0.0f, Animation.RELATIVE_TO_PARENT, 0.0f,
        Animation.RELATIVE_TO_PARENT, 0.0f, Animation.RELATIVE_TO_PARENT, +1.0f);

    animleftin.setDuration(1000);
}

```

```

animleftin.setInterpolator(new OvershootInterpolator());
animleftout.setDuration(1000);
animleftout.setInterpolator(new OvershootInterpolator());

animrightin.setDuration(1000);
animrightin.setInterpolator(new OvershootInterpolator());
animrightout.setDuration(1000);
animrightout.setInterpolator(new OvershootInterpolator());

animupin.setDuration(1000);
animupin.setInterpolator(new OvershootInterpolator());
animupout.setDuration(1000);
animupout.setInterpolator(new OvershootInterpolator());

animdownin.setDuration(1000);
animdownin.setInterpolator(new OvershootInterpolator());
animdownout.setDuration(1000);
animdownout.setInterpolator(new OvershootInterpolator());
}

private void prepareViews(){
    TextView view = null;

    views = new ArrayList<TextView>();

    for(int color: colors)
    {
        view = new TextView(this);

        view.setBackgroundColor(color);
        view.setTextColor(Color.BLACK);
        view.setGravity(Gravity.CENTER_HORIZONTAL | Gravity.CENTER_VERTICAL);

        views.add(view);
    }
}

private void addViews(){
    for(int index=0; index<views.size(); ++index)
    {
        flipper.addView(views.get(index),index,
            new LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT))
    }
}

private void setViewText(){
    String text = getString(isDragMode ? R.string.app_info_drag : R.string.app_info_flip);
    for(int index=0; index<views.size(); ++index)
    {
        views.get(index).setText(text);
    }
}

/**Gets invoked when a screen touch is detected*/

```

```

@Override
public boolean onTouchEvent(MotionEvent event) {
    return gesturedetector.onTouchEvent(event);
}

```

/**The onDown method is called when the user first touch the screen, the MotionEvent parameter represent to the touch event. */

```

@Override
public boolean onDown(MotionEvent e) {
    return false;
}

```

/**The onFling method is called whenever the user swipes the screen in any direction, i.e. the user touch moves the finger in any direction.*/

```

@Override
public boolean onFling(MotionEvent event1, MotionEvent event2, float velocityX, float velocityY) {
    if(isDragMode)
        return false;

    final float ev1x = event1.getX();
    final float ev1y = event1.getY();
    final float ev2x = event2.getX();
    final float ev2y = event2.getY();
    final float xdiff = Math.abs(ev1x - ev2x);
    final float ydiff = Math.abs(ev1y - ev2y);
    final float xvelocity = Math.abs(velocityX);
    final float yvelocity = Math.abs(velocityY);

    if(xvelocity > this.SWIPE_MIN_VELOCITY && xdiff > this.SWIPE_MIN_DISTANCE)
    {
        if(ev1x > ev2x) //Swipe Left
        {
            --currentview;

            if(currentview < 0)
            {
                currentview = views.size() - 1;
            }

            flipper.setInAnimation(animleftin);
            flipper.setOutAnimation(animleftout);
        }
        else //Swipe Right
        {
            ++currentview;

            if(currentview >= views.size())
            {
                currentview = 0;
            }

            flipper.setInAnimation(animrightin);
            flipper.setOutAnimation(animrightout);
        }
    }
}

```

```

        }
        flipper.scrollTo(0,0);
        flipper.setDisplayedChild(currentview);
    }
    else if(yvelocity > this.SWIPE_MIN_VELOCITY && ydiff > this.SWIPE_MIN_DISTANCE)
    {
        if(ev1y > ev2y) //Swipe Up
        {
            --currentview;

            if(currentview < 0)
            {
                currentview = views.size() - 1;
            }

            flipper.setInAnimation(animupin);
            flipper.setOutAnimation(animupout);
        }
        else //Swipe Down
        {
            ++currentview;

            if(currentview >= views.size())
            {
                currentview = 0;
            }
            flipper.setInAnimation(animdownin);
            flipper.setOutAnimation(animdownout);
        }

        flipper.scrollTo(0,0);
        flipper.setDisplayedChild(currentview);
    }

    return false;
}

```

/** The onLongPress method is called when user touches the screen and holds it for a period of time. The event that corresponds to the touch event. */

```

@Override
public void onLongPress(MotionEvent e) {
    vibrator.vibrate(200);
    flipper.scrollTo(0,0);

    isDragMode = !isDragMode;

    setViewText();
}

```

/**The onScroll method is called when the user touches the screen and moves to another location on the s

```

@Override
public boolean onScroll(MotionEvent e1, MotionEvent e2, float distanceX,float distanceY) {

```

```

        if(isDragMode)
            flipper.scrollBy((int)distanceX, (int)distanceY);

        return false;
    }

    /**The onShowPress method is called when the user touches the phone and not moved yet. This event is mos
to the user to show their action.*/

    @Override
    public void onShowPress(MotionEvent e) {
    }

    /**The onSingleTapUp method is called when a tap occurred, i.e. user taps the screen.*/

    @Override
    public boolean onSingleTapUp(MotionEvent e) {
        return false;
    }

    /** The onDoubleTap method is called when there is a double-tap event occurred. The only parameter Motio
event that occurred. */

    @Override
    public boolean onDoubleTap(MotionEvent e) {
        flipper.scrollTo(0,0);

        return false;
    }

    /** The onDoubleTapEvent is called for all events that occurred within the double-tap, i.e. down, move a

    @Override
    public boolean onDoubleTapEvent(MotionEvent e) {
        return false;
    }

    /** The onSingleTapConfirmed method is called when there is a single tap occurred and confirmed, but thi
event in the GestureDetector.onGestureListener. This is called when the GestureDetector detects and confirm
lead to a double-tap. */

    @Override
    public boolean onSingleTapConfirmed(MotionEvent e) {
        return false;
    }
}

```

When the mode of the application changes it is notified to the user with a vibration. To use the vibrator set the following permission in 'androidmanifest.xml' file of your application.

Example 6-104.

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


The Application uses some strings which are declared under 'res->values->string.xml'

Example 6-105.

```
&lt;?xml version="1.0" encoding="utf-8"?>
```

```
<resources>
  <string name="app_info_drag">GestureDetector sample.\n\nCurrent Mode: SCROLL\n\nDrag the view using fing
FLIP.\n\nDouble tap to reposition the view to normal.</string>
  <string name="app_name">Gesture Detector Sample</string>
  <string name="app_info_flip">GestureDetector sample.\n\nCurrent Mode: FLIP\n\nSwipe left, right, up, dow
change to mode to SCROLL</string>
</resources>
```

See Also

Check 'GestureOverlayView' class for handling complex gestures in android.

6.41 Customizing the Look of a Toast

Rachee Singh

Problem

You want to customize the look of Toast notifications.

Solution

By defining an XML layout for the toast and then inflating the view in Java, a Toast can be customized.

Discussion

We will first define the layout of the custom Toast in an XML file, toast_layout.xml. It contains an ImageView and a TextView:

Example 6-106.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/toast_layout_root"
  android:orientation="horizontal"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:padding="10dp"
  android:background="#f0fffef"
  >
  <ImageView android:id="@+id/image"
    android:layout_width="wrap_content"
    android:layout_height="fill_parent"
    android:layout_marginRight="10dp"
    />
  <TextView android:id="@+id/text"
```

```

        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:textColor="#000000"
    />
</LinearLayout>

```

Then, in the Java code, we inflate this view using `LayoutInflater`. We set the gravity and duration of the toast. The `setGravity` method modifies the position at which the toast will be displayed. On the click of the `customToast` button, we show the Toast.

Example 6-107.

```

        customToast = (Button)findViewById(R.id.customToast);

        LayoutInflater inflater = getLayoutInflater();
        View layout = inflater.inflate(R.layout.toast_layout, ViewGroup
        findViewById(R.id.toast_layout_root));

        ImageView image = (ImageView) layout.findViewById(R.id.image);
        image.setImageResource(R.drawable.icon);
        TextView text = (TextView) layout.findViewById(R.id.text);
        text.setText("Hello! This is a custom toast!");

        final Toast toast = new Toast(getApplicationContext());
        toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
        toast.setDuration(Toast.LENGTH_LONG);
        toast.setView(layout);
        customToast.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                toast.show();
            }
        });
    });

```

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LYTFjYjY4NWEtM2YzZC00NzEzLTg5ZGZEtMzFhM2UxOWM2MmFk&hl=en_US

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LM2VjMjFkNDUtNDA4ZS00MDRiLWFmNmEtOTYzODk3OGZlMjA0&hl=en_US

6.42 Using SlidingDrawer to Overlap Other Components

Mike Rowehl

Problem

The proper layout of SlidingDrawer isn't covered to well in the SDK documentation. Here's how to use the control to overlay other components in a layout. As well as how to position elements in the underlying layout to avoid the drawer handle.

Solution

The SlidingDrawer really should be in a FrameLayout or a RelativeLayout. Using it in a LinearLayout makes it difficult to get the drawer to overlay the rest of the controls on the screen. When using a RelativeLayout a new problem arises however. The SlidingDrawer wants to overlay everything. Thus positioning it over a ListView could result in the bottom entries of the list being covered. A clean solution is to use a spacer in the underlying layout to get everything to line up.

Discussion

First off is the layout including the SlidingDrawer itself. Note that there's a spacer TextView aligned with the bottom of the RelativeLayout using the DrawerButton style. The drawer handle itself is also a TextView using the same style. Positioning the main ListView for the layout above the spacer ensures that none of the list items are hidden by the handle when the drawer is closed.

Example 6-108.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextView style="@style/DrawerButton" android:layout_alignParentBottom="true"
        android:id="@+id/spacer" android:text="Spacer" />

    <ListView
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/contact_list"
        android:layout_alignParentTop="true"
        android:layout_above="@id/spacer"
    >
    </ListView>

    <SlidingDrawer android:layout_width="fill_parent"
        android:id="@+id/drawer" android:handle="@+id/drawer_button"
        android:content="@+id/drawer_content"
        android:layout_height="wrap_content" android:layout_alignParentBottom="true">
        <TextView android:id="@+id/drawer_button" style="@style/DrawerButton"
            android:gravity="right|center_vertical" android:text="Handle"
        >></TextView>
    <ListView
        android:layout_width="fill_parent"
```

```

        android:layout_height="fill_parent"
        android:id="@+id/drawer_content"
        android:background="#000000"
    >
</ListView>
</SlidingDrawer>
</RelativeLayout>

```

I pull the DrawerButton settings out into a style so that I don't have to change them on both the spacer and the handle item to keep them in sync:

Example 6-109.

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="DrawerButton" parent="@android:style/TextAppearance.Medium">
        <item name="android:layout_width">fill_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:background">#EEEEEE</item>
        <item name="android:textColor">#111111</item>
        <item name="android:gravity">right|center_vertical</item>
        <item name="android:paddingRight">3pt</item>
        <item name="android:paddingTop">2pt</item>
        <item name="android:paddingBottom">2pt</item>
    </style>
</resources>

```

Now the drawer should slide up over the ListView on the main screen without hiding any of the content when closed.

GUI: ListView

7.1 Introduction: ListView

Ian Darwin

Discussion

It may seem odd to have a separate chapter for the ListView component. But it is in fact one of the most important components, being used in probably 90% of all Android applications. And, it is very flexible; there is a lot you can do with it, but figuring out how is sometimes not as intuitive as it could be.

In this chapter we have everything from very basic ListView uses through to very advanced.

... XXX list of them ...

See the official doc at <http://developer.android.com/reference/android/widget/ListView.html>.

Another good overview of ListView can be found in a Google IO 2010 presentation on Google's YouTube channel, at <http://www.youtube.com/watch?v=wDBM6wVEO70>; this was presented by Romain Guy and Adam Powell who work on ListView itself.

7.2 Building list-based applications with ListView

Jim Blackler

Problem

Many mobile applications follow a similar pattern, allowing users to browse and interact with multiple items in a list. How can developers use standard Android UI classes to quickly build an app that works the way users will expect, providing them a list-based view onto their data.

Solution

ListView is an extremely versatile control. It is well suited to the screen size and control constraints of a mobile application, displaying information in a vertical stack of rows. This recipe shows how to set up a ListView, including rows that contain any combination of standard UI Views and perform actions on single and long-clicks.

Discussion

ListView

Many Android applications are based around the list ListView control. It is extremely versatile; well suited to the screen size and controls available on a mobile application. It solves the problem of how to present a lot of information in a way that's quick for the user to browse. It displays information in a vertical stack of rows that the user can scroll through. As the user reaches the results towards the end of the list, more results can be generated and added. This allows results paging in a natural and intuitive manner.

From using desktop applications, people are accustomed to UI controls which allow data to be manipulated in different ways. However, on the small screen of a phone these controls may be difficult to pick out in preference to others in the same area; particularly when the display is controlled with fingers rather than a stylus.

Android's ListView helps deal with this problem by separating browsing and editing operations into separate activities. A ListView simply requires the user to press somewhere in the row, which works well on a small, finger operated screen. When the row is clicked, a new Activity can be launched that can contain further options to manipulate the data shown in the row.

Menus

ListView rows can have context or 'long click' menus associated with them. This allows a list of actions that can be performed on the data represented by the row without navigating into the new activity. For instance, a list of search results could have actions such as 'Share' or 'Report spam' associated with them. One peril of this approach is that users often are not aware of the presence of long click controls.

Paging

Another advantage of the ListView format is that it allows paging in an uncomplicated way. Paging is where all the information requested by a user cannot feasibly be shown at once. For instance, the user may be browsing their email inbox, which contains 2,000 emails; it would not be feasible to download all 2,000 from the email server. Nor would it be required as the user will probably only scan the first ten or so entries.

Most web applications handle this problem by segmenting the results into pages, and having controls at the footer to allow the user to navigate through these pages. With a ListView, the application can retrieve an initial batch of the first results, which are

shown to the user in a list. When the user reaches the end of the list, a final row is seen, containing an indeterminate progress bar. As this comes into view, the application can fetch the next batch of results in the background. When they are ready to be shown, the last progress bar row is replaced with rows containing the new data. The user's view of the list is not interrupted, and new data is fetched purely on demand.

Implementation

To implement a `ListView` in your Android application, the first thing that is required is an activity layout to host it. This should contain a `ListView` control configured to take up most of the screen layout. You may wonder why a layout is required if it is to be dominated by a single full-screen `ListView`. The reason is that other elements such as progress bars or extra overlaid indicators may be supplied in the layout.

Base class

I would not recommend the use of `ListActivity` to host the view. It supplies little extra logic over a plain `Activity`, but using it restricts the form of the inheritance tree your application's activities can take. For instance, it is very common that all activities will inherit from a single common activity, e.g. `ApplicationActivity`, supplying common functionality such as 'About' or 'Help' menus. This pattern won't be possible if some activities are inherited from `ListActivity` and some are directly inherited from `Activity`.

An application controls the data added to a `ListView` by supplying a `ListAdapter` using the `setListAdapter()` method. There are 13 functions that a `ListAdapter` is expected to supply. However if a `BaseAdapter` is used this reduces to four, representing the minimum functionality that must be supplied. The adapter specifies the number of item rows in the list, and is expected to supply a `View` object to represent any item given its row number. It is also expected to return both an object and an object ID to represent any given row number. This is to aid advanced list features such as row selection (not covered by this tutorial).

The documentation steers developers to a simple version of the adapter (the `SimpleAdapter`) which simply binds array entries to text fields. I wouldn't recommend this approach as it is a dead end as far as development goes. If you need to add something more complex than just text, and you probably will, you'll have to remove the `SimpleAdapter` and convert it into a more flexible adapter type such as a `BaseAdapter`. I suggest starting with the most versatile type of `ListAdapter`, the `BaseAdapter` (`android.widget.BaseAdapter`). This allows any layout to be specified for a row (multiple layouts can be matched to multiple row types). These layouts can contain any `View` elements that a layout would normally contain.

Rows are created on demand by the adapter as they come on to the screen. The adapter is expected to either inflate a `View` of the appropriate type, or recycle the existing `View`, then customize it to display a row of data.

This 'recycling' is a technique employed by the Android OS to improve performance. When new rows come onto the screen, the OS will pass the `View` of a row that has moved off the screen into the adapter method `getView()`. It is up to the method to decide whether

it is appropriate to reuse that View to create the new row. For this to be the case the View has to represent the layout of the new row. One way to check this is to write the layout ID into the Tag of each View inflated with `setTag()`. When checking to see if it is appropriate to reuse a given View, use `getTag()` to see if the View was inflated with the correct type. If an application is able to recycle a view the scrolling appears to be smoother for the user because CPU time is saved inflating the view.

Another way to make scrolling smoother is to do as little as possible on the UI thread. This is the default thread that your `$` method will be invoked on. If time-intensive operations need to be invoked, these can be done by creating a new background thread especially for the operation. (`$example`). Then when the UI thread is required again so that controls can be updated, operations can be invoked on it with `$.` Care must be taken to ensure the View to be modified has not been recycled for another row. This can happen if the row has moved off the screen in the time it took the operation to complete. This is quite feasible if the operation was a lengthy download operation.

Setting up a basic ListView.

Use Eclipse's New Project wizard to create a new Android project with a starting activity called MainActivity. In the main.xml layout replace the existing `TextView` section with following:

Example 7-1.

```
<ListView android:id="@+id/ListView01"
    android:layout_width="wrap_content"
    android:layout_height="fill_parent"/>
```

In `MainActivity.onCreate()` insert the following snippet at the bottom of the method. This will declare a dummy anonymous class extending `BaseAdapter`, and apply an instance of it to the `ListView`. The code illustrates the methods that need to be supplied in order to populate the `ListView` with data.

Example 7-2.

```
ListView listView = (ListView) findViewById(R.id.ListView01);
listView.setAdapter(new BaseAdapter(){

    public int getCount() {
        return 0;
    }

    public Object getItem(int position) {
        return null;
    }

    public long getItemId(int position) {
        return 0;
    }

    public View getView(int position, View convertView, ViewGroup parent) {
```

```

        return null;
    });
}

```

By customizing the anonymous class members the developer can modify the data shown by the control. However before any data can be shown, a layout must be supplied to present the data in rows. Add a file `list_row.xml` to your project's `res/layout` directory with the following content:

Example 7-3.

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="wrap_content"
    <TextView android:text="@+id/TextView01" android:id="@+id/TextView01" android:layout_width="fill_parent" a
</LinearLayout>

```

Now in your `MainActivity` add the following static array field containing just three strings.

Example 7-4.

```

static String[] words = {"one", "two", "three"};

```

Now customize your existing anonymous `BaseAdapter` as follows, in order to display the contents of the `words` array in the `ListView`.

Example 7-5.

```

listView.setAdapter(new BaseAdapter(){

    public int getCount() {
        return words.length;
    }

    public Object getItem(int position) {
        return words[position];
    }

    public long getItemId(int position) {
        return position;
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = (LayoutInflater) getSystemService(LAYOUT_INFLATER_SERVICE);
        View view = inflater.inflate(R.layout.list_row, null);
        TextView textView = (TextView) view.findViewById(R.id.TextView01);
        textView.setText(words[position]);
        return view;
    }
});

```

`getCount()` is customized to return the number of items in the list. `getItem()` and `getItemId()` supply the `ListView` with unique objects and IDs to identify the data the rows. Finally `getView()` creates and customizes an Android View to represent the row. This is the most complex step, so let's break down what's done.

Example 7-6.

```
LayoutInflater inflater = (LayoutInflater) getSystemService(LAYOUT_INFLATER_SERVICE);
```

The system `LayoutInflater` is obtained. This is the service that creates views.

Example 7-7.

```
View view = inflater.inflate(R.layout.list_row, null)
```

The new layout we created earlier is inflated.

Example 7-8.

```
TextView textView = (TextView) view.findViewById(R.id.TextView01)
```

The `TextView` is located.

Example 7-9.

```
textView.setText(words[position])
```

The `TextView` is customized with the appropriate item in the words array.

Example 7-10.

```
return view;
```

The view is returned to the system for display.

View Recycling

XXX

7.3 'No data' View for Lists

Rachee Singh

Problem

When the `ListView` has no items to show, the screen would just appear blank. In order to show an appropriate message on the screen, indicating the absence of data in the list.

Solution

Use of 'No Data' View from the XML layout.

Discussion

Often we require to use a `ListView` in an Android App. Prior to the input of data by the user, the List is empty. This in general would show a blank screen. In order to make the user feel better, we might want to display an appropriate message (or even an image)

stating that the list is empty. For this purpose, the concept of No Data View can be used. This is a simple process involving addition of a few lines of code in the XML layout of the activity that contains the list view.

Example 7-11.

```
&lt;?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

<ListView
    android:id="@id/android:list"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/textView1"/>
    <TextView
        android:id="@id/android:empty"
        android:text="@string/list_is_empty"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_below="@id/textView1"
        android:textSize="25sp"
        android:gravity="center_vertical|center_horizontal"/>
</RelativeLayout>
```

The important point is line 'android:id="@id/android:empty'. This line ensures that when the list is empty, the TextView with this ID will be displayed on the screen. In this text view the string 'List is Empty' is displayed.

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LMDE4OGM5MGEtMWM3Ny00NDdhLWE4YTItNzljMGJhNWlxNzE0&hl=en_US&authkey=CJvf1KYO

7.4 Advanced ListView: populating a list with images and text

Marco Dinacci

Problem

You want to write a ListView that shows an image next to a string.

Solution

We're going to create an Activity that extends from ListActivity, prepare the XML resource files and at last create a custom view adapter to load the resources onto the view.

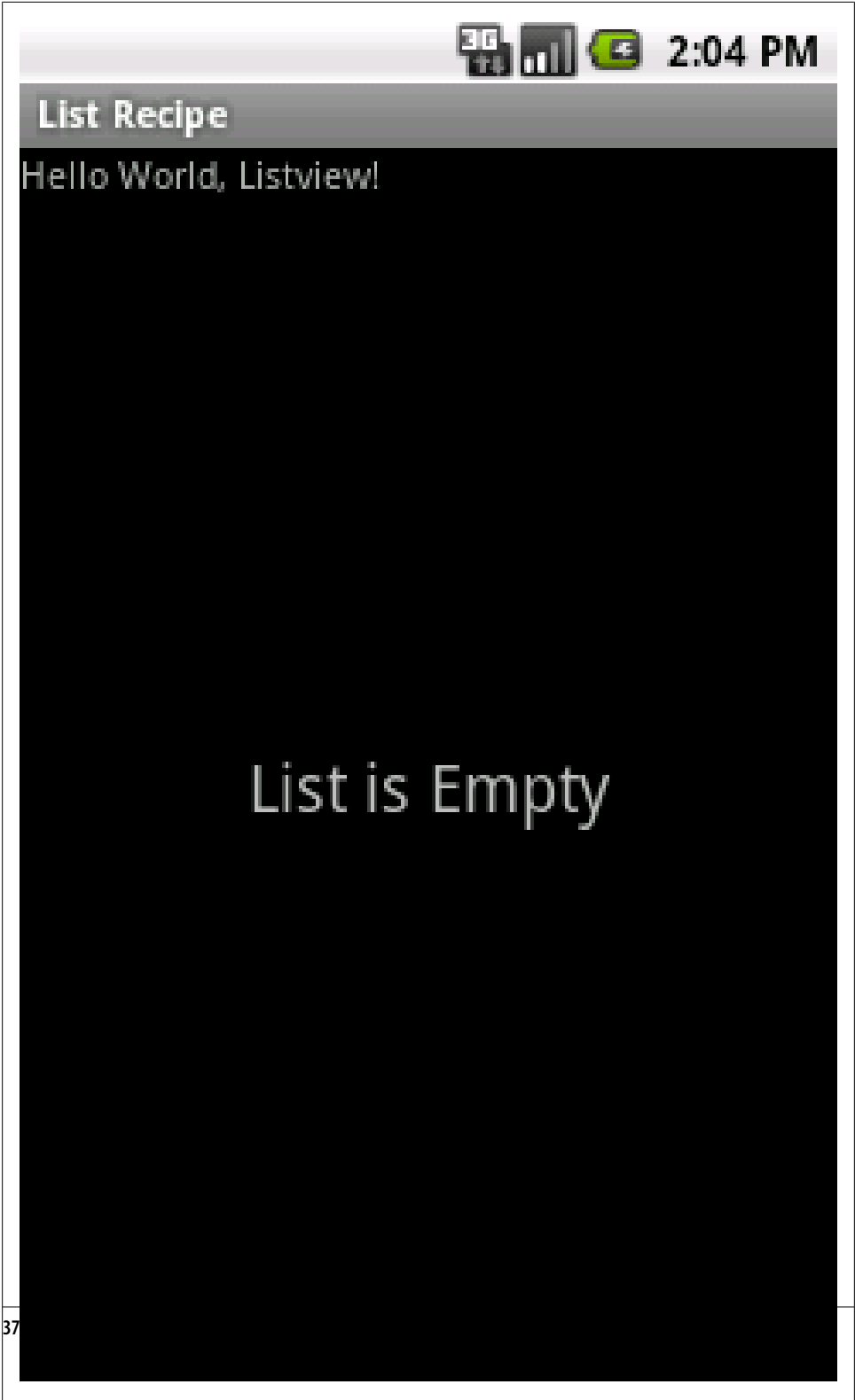


Figure 7-1.

Discussion

The Android documentation says that the `ListView` widget is easy to use. It is true if you just want to display a simple list of strings but as soon as you want to customize your list things become more complicated.

This recipe shows you how to write a `ListView` that displays a static list of images and strings, similar to the settings list on your phone.

Here's a picture of the final result:

Let's start with the `Activity` code. First of all, we extend from `ListActivity` instead of `Activity` so we can easily supply our custom adapter:

Example 7-12.

```
public class AdvancedListViewActivity extends ListActivity {

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

        Context ctx = getApplicationContext();
        Resources res = ctx.getResources();

        String[] options = res.getStringArray(R.array.country_names);
        TypedArray icons = res.obtainTypedArray(R.array.country_icons);

        setListAdapter(new ImageAndTextAdapter(ctx, R.layout.main_list_item, options, icons));
    }
}
```


In the `onCreate` we also create an array of strings, which contains the country names, and a `TypedArray`, which will contain our `Drawable` flags.

The arrays are created from an XML file, here's the content of the `countries.xml` file.


Example 7-13.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="country_names">
        <item>Bhutan</item>
        <item>Colombia</item>
        <item>Italy</item>
        <item>Jamaica</item>
        <item>Kazakhstan</item>
        <item>Kenya</item>
    </string-array>
    <array name="country_icons">
        <item>@drawable/bhutan</item>
        <item>@drawable/colombia</item>
        <item>@drawable/italy</item>
    </array>
</resources>
```

AdvancedListViewDemo


 Bhutan

 Colombia

 Italy

 Jamaica

 Kazakhstan

 Kenya

```

        <item>@drawable/jamaica</item>
        <item>@drawable/kazakhstan</item>
        <item>@drawable/kenya</item>
    </array>
</resources>

```

Now we're ready to create the adapter. The official documentation for `Adapter` says:

An `Adapter` object acts as a bridge between an `AdapterView` and the underlying data for that view. The `Adapter` provides access to the data items. The `Adapter` is also responsible for making a `View` for each item in the data set.

There are several subclasses of `Adapter`; we're going to extend on `ArrayAdapter` which is a concrete `BaseAdapter` that is backed by an array of arbitrary objects.

Example 7-14.

```

public class ImageAndTextAdapter extends ArrayAdapter<String> {

    private LayoutInflater mInflater;

    private String[] mStrings;
    private TypedArray mIcons;

    private int mViewResourceId;

    public ImageAndTextAdapter(Context ctx, int viewResourceId,
        String[] strings, TypedArray icons) {
        super(ctx, viewResourceId, strings);

        mInflater = (LayoutInflater)ctx.getSystemService(
            Context.LAYOUT_INFLATER_SERVICE);

        mStrings = strings;
        mIcons = icons;

        mViewResourceId = viewResourceId;
    }

    @Override
    public int getCount() {
        return mStrings.length;
    }

    @Override
    public String getItem(int position) {
        return mStrings[position];
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    @Override

```



```

public View getView(int position, View convertView, ViewGroup parent) {
    convertView = inflater.inflate(mViewResourceId, null);

    ImageView iv = (ImageView)convertView.findViewById(R.id.option_icon);
    iv.setImageDrawable(mIcons.getDrawable(position));

    TextView tv = (TextView)convertView.findViewById(R.id.option_text);
    tv.setText(mStrings[position]);

    return convertView;
}
}

```

The constructor accepts a `Context`, the `id` of the layout that will be used for every row (more on this soon), an array of strings (the country names) and a `TypedArray` (our flags).

The `getView` method is where we build a row for the list. We first use a `LayoutInflater` to create a `View` from XML, then we retrieve the country flag as a `Drawable` and the country name as a `String` and we use them to populate the `ImageView` and `TextView` that we've declared in the layout.

The layout for the list rows is the following:

Example 7-15.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android">
    <ImageView
        android:id="@+id/option_icon"
        android:layout_width="48dp"
        android:layout_height="fill_parent"/>
    <TextView
        android:id="@+id/option_text"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:padding="10dp"
        android:textSize="16dp" >
    </TextView>
</LinearLayout>

```

And this is the content of the main layout:

Example 7-16.

```

<?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"
    >
<ListView android:id="@android:id/list"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent"
    />
</LinearLayout>

```

Note that the `ListView` ID must be exactly `@android:id/list` or you'll get a `RuntimeException`.

Source Download URL

The source code for this example may be downloaded from this URL: http://www.intransitio.com/intransitio.com/code/android/adv_listview_demo.zip

7.5 ListView with Icons/Images

Wagied Davids

Problem

Loading data from XML based file, Viewing results in a `ListView` with an icon image for each `ListView` row item.

Solution

Create a custom `ArrayAdapter` for modifying visual display of list items.

File: `listview.xml` - The XML-layout file

Example 7-17.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content">
<ListView
    android:id="@+id/countryLV"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
</ListView>
</LinearLayout>
```

File: `country_listitem.xml`

Example 7-18.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">

    <ImageView
        android:id="@+id/country_icon"
        android:layout_gravity="left"
        android:layout_width="wrap_content"
```

```

        android:layout_height="wrap_content" />

<TextView
    android:id="@+id/country_name"
    android:text="Country Name"
    android:paddingLeft="10dip"
    android:layout_weight="0.5"
    android:layout_gravity="center"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
<TextView
    android:id="@+id/country_abbrev"
    android:text="Country Abbrev"
    android:layout_gravity="right"
    android:paddingRight="10dip"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

</LinearLayout>

```

File: countries.xml

Example 7-19.

```

<?xml version="1.0" encoding="utf-8"?>
<countries>
    <country name="Australia" abbreviation="au" region="Asia" />
    <country name="Austria" abbreviation="at" region="Europe" />
    <country name="Belgium" abbreviation="be" region="Europe" />
    <country name="Brazil" abbreviation="br" region="S. America" />
    <country name="Canada" abbreviation="ca" region="N. America" />
    <country name="China" abbreviation="cn" region="Asia" />
    <country name="Denmark" abbreviation="dk" region="Europe" />
    <country name="France" abbreviation="fr" region="Europe" />
    <country name="Germany" abbreviation="de" region="Europe" />
    <country name="Hong Kong" abbreviation="hk" region="Asia" />
    <country name="India" abbreviation="in" region="Asia" />
    <country name="Indonesia" abbreviation="id" region="Asia" />
    <country name="Italy" abbreviation="it" region="Europe" />
    <country name="Korea" abbreviation="kr" region="Asia" />
    <country name="Netherlands" abbreviation="nl" region="Europe" />
    <country name="Norway" abbreviation="no" region="Europe" />
    <country name="Portugal" abbreviation="pt" region="Europe" />
    <country name="Singapore" abbreviation="sg" region="Asia" />
    <country name="Spain" abbreviation="es" region="Europe" />
    <country name="Sweden" abbreviation="se" region="Europe" />
    <country name="Switzerland" abbreviation="ch" region="Europe" />
    <country name="Taiwan" abbreviation="tw" region="Asia" />
    <country name="United Kingdom" abbreviation="uk" region="Europe" />
    <country name="United States" abbreviation="us" region="N. America" />
</countries>

```

File: Main.java

Example 7-20.

```
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.widget.ListView;

public class Main extends Activity {

    private List<Country> countryList= new ArrayList<Country>();

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Set the View layer
        setContentView(R.layout.listview);
        setTitle("TestIconizedListView");

        // Create Parser for raw/countries.xml
        CountryParser countryParser = new CountryParser();
        InputStream inputStream = getResources().openRawResource(
            R.raw.countries);

        // Parse the inputstream
        countryParser.parse(inputStream);

        // Get Countries
        List<Country> countryList = countryParser.getList();

        // Create a customized ArrayAdapter
        CountryArrayAdapter adapter = new CountryArrayAdapter(
            getApplicationContext(), R.layout.country_listitem, countryList);

        // Get reference to ListView holder
        ListView lv = (ListView) this.findViewById(R.id.countryLV);

        // Set the ListView adapter
        lv.setAdapter(adapter);
    }
}
```

File: Country.java

Example 7-21.

```
public class Country
{
    public String name;
    public String abbreviation;
    public String region;
    public String resourceId;
}
```

```

public Country()
{
    // TODO Auto-generated constructor stub
}

public Country(String name, String abbreviation, String region, String resourceFilePath)
{
    this.name = name;
    this.abbreviation = abbreviation;
    this.region= region;
    this.resourceId = resourceFilePath;
}

@Override
public String toString()
{
    return this.name;
}
}

```

File: CountryParser.java

Example 7-22.

```

import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
import android.util.Log;

public class CountryParser {

    private static final String tag = "CountryParser";
    private static final String FILE_EXTENSION= ".png";

    private DocumentBuilderFactory factory;
    private DocumentBuilder builder;
    private final List<Country> list;

    public CountryParser() {
        this.list = new ArrayList<Country>();
    }

    private String getNodeValue(NamedNodeMap map, String key) {
        String nodeValue = null;

```

```

        Node node = map.getNamedItem(key);
        if (node != null) {
            nodeValue = node.getNodeValue();
        }
        return nodeValue;
    }

    public List<Country> getList() {
        return this.list;
    }

    /**
     * Parse XML file containing body part X/Y/Description
     *
     * @param inStream
     */
    public void parse(InputStream inStream) {
        try {
            // TODO: after we must do a cache of this XML!!!!
            this.factory = DocumentBuilderFactory.newInstance();
            this.builder = this.factory.newDocumentBuilder();
            this.builder.isValidating();
            Document doc = this.builder.parse(inStream, null);

            doc.getDocumentElement().normalize();

            NodeList countryList = doc.getElementsByTagName("country");
            final int length = countryList.getLength();

            for (int i = 0; i < length; i++) {
                final NamedNodeMap attr = countryList.item(i).getAttributes();
                final String countryName = getNodeValue(attr, "name");
                final String countryAbbr = getNodeValue(attr, "abbreviation");
                final String countryRegion = getNodeValue(attr, "region");

                // Construct Country object
                Country country = new Country(countryName, countryAbbr,
                    countryRegion, countryAbbr + FILE_EXTENSION);

                // Add to list
                this.list.add(country);

                Log.d(tag, country.toString());
            }
        } catch (SAXException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ParserConfigurationException e) {
            e.printStackTrace();
        }
    }
}

```

File: CountryArrayAdapter.java

Example 7-23.

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ImageView;
import android.widget.TextView;

public class CountryArrayAdapter extends ArrayAdapter<Country> {

    private static final String tag = "CountryArrayAdapter";
    private static final String ASSETS_DIR = "images/";
    private Context context;

    private ImageView countryIcon;
    private TextView countryName;
    private TextView countryAbbrev;
    private List<Country> countries = new ArrayList<Country>();

    public CountryArrayAdapter(Context context, int textViewResourceId,
        List<Country> objects) {
        super(context, textViewResourceId, objects);
        this.context = context;
        this.countries = objects;
    }

    public int getCount() {
        return this.countries.size();
    }

    public Country getItem(int index) {
        return this.countries.get(index);
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        View row = convertView;
        if (row == null) {
            // ROW INFLATION
            Log.d(tag, "Starting XML Row Inflation ... ");
            LayoutInflater inflater = (LayoutInflater) this.getContext()
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            row = inflater.inflate(R.layout.country_listitem, parent, false);
            Log.d(tag, "Successfully completed XML Row Inflation!");
        }

        // Get item
        Country country = getItem(position);
```

```

// Get reference to ImageView
countryIcon = (ImageView) row.findViewById(R.id.country_icon);

// Get reference to TextView - country_name
countryName = (TextView) row.findViewById(R.id.country_name);

// Get reference to TextView - country_abbrev
countryAbbrev = (TextView) row.findViewById(R.id.country_abbrev);

//Set country name
countryName.setText(country.name);

// Set country icon usign File path
String imgFilePath = ASSETS_DIR + country.resourceId;
try {
    Bitmap bitmap = BitmapFactory.decodeStream(this.context.getResources().getAssets()
        .open(imgFilePath));
    countryIcon.setImageBitmap(bitmap);
} catch (IOException e) {
    e.printStackTrace();
}

// Set country abbreviation
countryAbbrev.setText(country.abbreviation);
return row;
}
}

```

Discussion

This recipe demonstrate how to load data from a XML file, and view the results in a ListView with each row item having an image/icon.

Firstly, the XML resource file must be place in a raw/ directory beneath the res/ folder. In the countries.xml file, all the data is enclosed in the `countries` and `countries` tag with each element being enclosed in a `country` tag with attributes indicating the name, abbreviation and region of the respective country.

A Java class `Country.java` is used to model each ListView row item. XML parsing in Android is accomplished by using the built-in SAX parser. In parsing the `countries.xml` file, the parser `CountryParser.java` is used to obtain the country names, abbreviations, and regions using the node values extracted from `countries.xml`. A new `Country` java object for each XML node which includes the filename of the country image to use.

The above explanation covers the data input from an XML-based file. Now moving to the meat and bones so to speak.

`CountryArrayAdapter` extends the `ArrayAdapter` class in which a `Context` and a list of `Country` objects are supplied as parameters. The most important function is the `getView()` function. The `getView()` function allows the ListView row elements to be altered, in this case the icon image is inserted to the left, and the country name and abbreviation to the right of it.

A simple XML-based View layer `listview.xml` is created, in which a reference to a `countryLV` is declared in a `ListView` element.

For data-binding, a `Main.java` Android activity is created in which the various elements describe above is brought together. The function `setContentView(R.layout.listview)` loads the XML view file `listview.xml`, and extracts a reference to the `countryLV` `ListView` holder. The XML parser, called `CountryParser` loads the data into a Java `List` containing `Country` objects.

The custom `CountryArrayAdapter` is used to override the `ListView` display so that each row contains the country icon, country name and abbreviation.

Re-wrote this example: 1. `xml` for `listview` of countries 2. `xml` for each row item in `listview` with place holders for image, `country_name`, `country_abbrev` 3. Customized `ArrayAdapter` for using `Country` object. 4. Accessing image files in the `assets/` directory using: `// Set country icon usign File path String imgFilePath = ASSETS_DIR + country.resourceId; try { Bitmap bitmap = BitmapFactory.decodeStream(this.context.getResources().getAssets().open(imgFilePath)); countryIcon.setImageBitmap(bitmap); } catch (IOException e) { e.printStackTrace(); }`

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.filefactory.com/file/b4a5548/n/IconizedListView.zip>

7.6 Sectioned Headers in ListViews

Wagied Davids

Problem

Sectioned headers in a list - when you want to display categorized items eg. by time/day, by product category or sales price.

Solution

Use Jeff Sharkey's "Sectioned Headers" to display journal entries by day.

Discussion

File: `main.xml`

Example 7-24.

```
<?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">

        <ListView
            android:id="@+id/add_journalentry_menuitem"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
        <ListView
            android:id="@+id/list_journal"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
    </LinearLayout>

```

File: list_header.xml

Example 7-25.

```

<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/list_header_title"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:paddingTop="2dip"
    android:paddingBottom="2dip"
    android:paddingLeft="5dip"
    style="?android:attr/listSeparatorTextViewStyle" />

```

File: list_item.xml

Example 7-26.

```

<?xml version="1.0" encoding="utf-8"?>
<!-- list_item.xml -->
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/list_item_title"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingTop="10dip"
    android:paddingBottom="10dip"
    android:paddingLeft="15dip"
    android:textAppearance="?android:attr/textAppearanceLarge"
    />

```

File: list_complex.xml

```

<example><title></title><programlisting><![CDATA[
<?xml version="1.0" encoding="utf-8"?>
<!-- list_complex.xml -->
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:paddingTop="10dip"
    android:paddingBottom="10dip"
    android:paddingLeft="15dip"
    >

```

```

<TextView
    android:id="@+id/list_complex_title"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    />
<TextView
    android:id="@+id/list_complex_caption"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceSmall"
    />
</LinearLayout>

```

File: add_journalentry_menuitem.xml

Example 7-27.

```

<?xml version="1.0" encoding="utf-8"?>
<!-- list_item.xml -->
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/list_item_title"
    android:gravity="right"
    android:drawableRight="@drawable/ic_menu_add"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingTop="0dip"
    android:paddingBottom="0dip"
    android:paddingLeft="10dip"
    android:textAppearance="?android:attr/textAppearanceLarge" />

```

File: ListSample.java

Example 7-28.

```

import java.util.HashMap;
import java.util.Map;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class ListSample extends Activity
{
    public final static String ITEM_TITLE = "title";
    public final static String ITEM_CAPTION = "caption";

    // SectionHeaders
    private final static String[] days = new String[]{"Mon", "Tue", "Wed", "Thur", "Fri"};

```

```

// Section Contents
private final static String[] notes = new String[]{"Ate Breakfast", "Ran a Marathan ...yah really",

// MENU - ListView
private ListView addJournalEntryItem;

// Adapter for ListView Contents
private SeparatedListAdapter adapter;

// ListView Contents
private ListView journalListView;

public Map<String, ?> createItem(String title, String caption)
{
    Map<String, String> item = new HashMap<String, String>();
    item.put(ITEM_TITLE, title);
    item.put(ITEM_CAPTION, caption);
    return item;
}

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

    // Sets the View Layer
    setContentView(R.layout.main);

    // Interactive Tools
    final ArrayAdapter<String> journalEntryAdapter = new ArrayAdapter<String>(this, R.layout.add

    // AddJournalEntryItem
    addJournalEntryItem = (ListView) this.findViewById(R.id.add_journalentry_menuitem);
    addJournalEntryItem.setAdapter(journalEntryAdapter);
    addJournalEntryItem.setOnItemClickListener(new OnItemClickListener()
    {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long duration)
        {
            String item = journalEntryAdapter.getItem(position);
            Toast.makeText(getApplicationContext(), item, Toast.LENGTH_SHORT).show();
        }
    });

    // Create the ListView Adapter
    adapter = new SeparatedListAdapter(this);
    ArrayAdapter<String> listadapter = new ArrayAdapter<String>(this, R.layout.list_item, notes)

    // Add Sections
    for (int i = 0; i < days.length; i++)
    {
        adapter.addSection(days[i], listadapter);
    }

    // Get a reference to the ListView holder

```

```

journalListView = (ListView) this.findViewById(R.id.list_journal);

// Set the adapter on the ListView holder
journalListView.setAdapter(adapter);

// Listen for Click events
journalListView.setOnItemClickListener(new OnItemClickListener()
{
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long duration)
    {
        String item = (String) adapter.getItem(position);
        Toast.makeText(getApplicationContext(), item, Toast.LENGTH_SHORT).show();
    }
});
}
}

```

File: SeparatedListAdapter.java

Example 7-29.

```

import java.util.LinkedHashMap;
import java.util.Map;
import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.Adapter;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;

public class SeparatedListAdapter extends BaseAdapter
{
    public final Map<String, Adapter> sections = new LinkedHashMap<String, Adapter>();
    public final ArrayAdapter<String> headers;
    public final static int TYPE_SECTION_HEADER = 0;

    public SeparatedListAdapter(Context context)
    {
        headers = new ArrayAdapter<String>(context, R.layout.list_header);
    }

    public void addSection(String section, Adapter adapter)
    {
        this.headers.add(section);
        this.sections.put(section, adapter);
    }

    public Object getItem(int position)
    {
        for (Object section : this.sections.keySet())
        {
            Adapter adapter = sections.get(section);
            int size = adapter.getCount() + 1;

```

```

        // check if position inside this section
        if (position == 0) return section;
        if (position < size) return adapter.getItem(position - 1);

        // otherwise jump into next section
        position -= size;
    }
    return null;
}

public int getCount()
{
    // total together all sections, plus one for each section header
    int total = 0;
    for (Adapter adapter : this.sections.values())
        total += adapter.getCount() + 1;
    return total;
}

@Override
public int getViewTypeCount()
{
    // assume that headers count as one, then total all sections
    int total = 1;
    for (Adapter adapter : this.sections.values())
        total += adapter.getViewTypeCount();
    return total;
}

@Override
public int getItemViewType(int position)
{
    int type = 1;
    for (Object section : this.sections.keySet())
    {
        Adapter adapter = sections.get(section);
        int size = adapter.getCount() + 1;

        // check if position inside this section
        if (position == 0) return TYPE_SECTION_HEADER;
        if (position < size) return type + adapter.getItemViewType(position - 1);

        // otherwise jump into next section
        position -= size;
        type += adapter.getViewTypeCount();
    }
    return -1;
}

public boolean areAllItemsSelectable()
{
    return false;
}

```

```

@Override
public boolean isEnabled(int position)
{
    return (getItemViewType(position) != TYPE_SECTION_HEADER);
}

@Override
public View getView(int position, View convertView, ViewGroup parent)
{
    int sectionnum = 0;
    for (Object section : this.sections.keySet())
    {
        Adapter adapter = sections.get(section);
        int size = adapter.getCount() + 1;

        // check if position inside this section
        if (position == 0) return headers.getView(sectionnum, convertView, parent);
        if (position < size) return adapter.getView(position - 1, convertView, parent);

        // otherwise jump into next section
        position -= size;
        sectionnum++;
    }
    return null;
}

@Override
public long getItemId(int position)
{
    return position;
}
}

```

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.filefactory.com/file/b3g5g1b/n/TestSectionedHeaderList.zip>

7.7 Making Lists Behave Nicely

Ian Darwin

Problem

Lists are easy to get started with, but there are a lot of variations that will provide optimal user experience.

Solution

Studying the ListView API and considering the user experience will lead to some improvements.

Discussion

This will list such items as:

- Set the current position when adding;
- Use the `overScroll` header and footer;
- List Navigator for large lists (like Contacts and OISafe)
- Adjust list content dynamically
- Filter non-text List elements

The net result will make your application easier to use and more predictable for users.

See Also

The official [ListView documentation](#).

7.8 Writing A Custom List Adapter

Alex Leffelman

Problem

Lists are ubiquitous throughout mobile applications. Android provides a simple and powerful interface to make exactly the list you need. This recipe explains the steps for customizing the content of a `ListView`.

Solution

In the Activity that will host your `ListView`, we will define a private class that extends Android's `BaseAdapter` class. We will override the base class's methods to display custom views that you define in an XML layout file.

Discussion

It's no secret that the best way to explain something is through an example, so let's dive in. This is code lifted out of a media application I wrote that allowed the user to build playlists from the songs on their SD card. As promised, we'll be extending the `BaseAdapter` class inside of my `MediaListActivity`:

Example 7-30.

```
private class MediaAdapter extends BaseAdapter {  
    ...  
}
```

Querying the phone for the media info is outside the scope of this recipe, but the data to populate the list was stored in a `MediaItem` class that kept standard Artist, Title,

Album, and Track Number information, as well as a boolean field indicating if the item was selected for the current playlist. In certain cases you may want to continually add items to your list - for example, if you're downloading information and displaying it as it comes in - but for this purpose we're going to supply all the required data to the Adapter at once in the constructor.

Example 7-31.

```
public MediaAdapter(ArrayList<MediaItem> items) {
    mMediaList = items;
    ...
}
```

Now, if you're developing in Eclipse you'll notice that it wants us to override BaseAdapter's abstract methods. Let's take a look at those.

Example 7-32.

```
public int getCount() {
    return mMediaList.size();
}
```

The framework needs to know how many Views it needs to create in your list. It finds out by asking your Adapter how many items you're managing. In our case we'll have a View for every item in the media list.

Example 7-33.

```
public Object getItem(int position) {
    return mMediaList.get(position);
}
public long getItemId(int position) {
    return position;
}
```

We won't really be using these methods, but for completeness: `getItem(int)` is what gets returned when the ListView hosting this adapter calls `getItemAtPosition(int)`, which won't happen in our case. `getItemId(int)` is what gets passed to the `ListView.onItemClickListener(ListView, View, int, int)` callback when you select an item. It gives you the position of the view in the list and the ID supplied by your adapter. In our case they're the same.

The real work of your custom adapter will be done in the `getView()` method. This method is called every time the ListView brings a new item into view. When an item goes out of view, it is recycled by the system to be used later. This is a powerful mechanism for providing potentially thousands of View objects to our ListView while using only as many Views as can be displayed on the screen. The `getView()` method provides the position of the item it is creating, a View that may be not-null which the system is recycling for you to use, and the ViewGroup parent. You'll return either a new View

for the list to display, or a modified copy of the supplied convertView parameter to conserve system resources. Let's look at the code:

Example 7-34.

```
public View getView(int position, View convertView, ViewGroup parent) {
    View V = convertView;

    if(V == null) {
        LayoutInflater vi = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        V = vi.inflate(R.layout.media_row, null);
    }

    MediaItem mi = mMedialist.get(position);
    ImageView icon = (ImageView)V.findViewById(R.id.media_image);
    TextView title = (TextView)V.findViewById(R.id.media_title);
    TextView artist = (TextView)V.findViewById(R.id.media_artist);

    if(mi.isSelected()) {
        icon.setImageResource(R.drawable.item_selected);
    }
    else {
        icon.setImageResource(R.drawable.item_unselected);
    }

    title.setText(mi.getTitle());
    artist.setText("by " + mi.getArtist());

    return V;
}
```

We start by checking if we'll be recycling a View (which is good practice), or if we need to generate a new View from scratch. If we weren't given a convertView, we'll call the LayoutInflater service to build a View that we've defined in an XML layout file.

Using the View which we've ensured was built with our desired layout resource (or is a recycled copy of one we previously built), it's simply a matter of updating its UI elements. In our case we want to display the song title, the artist, and an indication of whether or not the song is in our current playlist. (I've removed the error-checking, but it's good practice to make sure any UI elements you're updating are not null - you don't want to crash the whole ListView if there was a small mistake in one item) This method gets called for every (visible) item in the ListView, so in this example we have a list of identical View objects with different data being displayed in each one. If you wanted to get really creative, you could populate the list with different view layouts based on its position or content.

That takes care of the required BaseAdapter overrides. However, you can add any functionality to your Adapter to work on the data set it represents. In my example, I want the user to be able to click a list item and toggle it on/off for the current playlist. This is easily accomplished with a simple callback on the ListView and a short function in the Adapter:

(This function belongs to ListActivity)

Example 7-35.

```
protected void onItemClick(ListView l, View v, int position, long id) {
    super.onItemClick(l, v, position, id);

    mAdapter.toggleItem(position);
}
```

(This is a member function in our MediaAdapter)

Example 7-36.

```
public void toggleItem(int position) {
    MediaItem mi = mMedialist.get(position);

    mi.setSelected(!mi.getSelected());
    mMedialist.set(position, mi);

    this.notifyDataSetChanged();
}
```

First we simply register a callback for when the user clicks an item in our list. We're given the ListView, the View, the position, and the ID of the item that was clicked, but we'll only need the position, which we simply pass to the MediaAdapter.toggleItem(int) method. In that method we update the state of the corresponding MediaItem and make an important call to notifyDataSetChanged(). This method lets the framework know that it needs to redraw the ListView. If we don't call it, we can do whatever we want to the data, but we won't see anything change until the next redraw (for example, when we scroll the list).

When it's all said and done, we need to tell the parent ListView to use our Adapter to populate the list. That's done with a simple call in the ListActivity's onCreate(Bundle) method:

Example 7-37.

```
MediaAdapter mAdapter = new MediaAdapter(getSongsFromSD());
this.setListAdapter(mAdapter);
```

First we instantiate a new Adapter with data generated from a private function that queries the phone for the song data, then we tell the ListActivity to use that adapter to draw the list. And there it is - your own list adapter with a custom view and extensible functionality.

7.9 Orientation Changes : From ListView data values to Landscape Charting

Wagied Davids

Problem

Accomplish view changes on device orientation changes. For example, data values to be plotted are contained in a Portrait ListView, and upon device orientation a graphical display of the data values in a chart/plot is displayed.

Solution

In this example, data values to be plotted are contained in a Portrait ListView. When the device/emulator is changed to counter-clockwise, a new Intent is launched to change to a plot/charting View to graphically display the data values. Charting is accomplished using the excellent DroidCharts package (<http://code.google.com/p/droidcharts/>).

File: AndroidManifest.xml

Example 7-38.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.examples"
  android:versionCode="1"
  android:versionName="1.0">
  <application
    android:icon="@drawable/icon"
    android:label="@string/app_name"
    android:debuggable="true">
    <activity
      android:name=".DemoList"
      android:label="@string/app_name"
      android:configChanges="orientation|keyboardHidden"
      android:screenOrientation="portrait">
      <intent-filter>
        <action
          android:name="android.intent.action.MAIN" />
        <category
          android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <activity
      android:name=".DemoCharts"
      android:configChanges="orientation|keyboardHidden"></activity>
  </application>
</manifest>
```

File: DemoCharts.java

Example 7-39.

```
import java.util.ArrayList;

import net.droidsolutions.droidcharts.core.data.XYDataset;
```

```

import net.droidsolutions.droidcharts.core.data.xy.XYSeries;
import net.droidsolutions.droidcharts.core.data.xy.XYSeriesCollection;
import android.app.Activity;
import android.content.Intent;
import android.content.res.Configuration;
import android.os.Bundle;
import android.util.Log;
import android.widget.Toast;

public class DemoCharts extends Activity
{
    private static final String tag = "DemoCharts";
    private final String chartTitle = "My Daily Starbucks Allowance";
    private final String xLabel = "Week Day";
    private final String yLabel = "Allowance";

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        // Access the Extras from the Bundle
        Bundle params = getIntent().getExtras();

        // If we get no parameters, we do nothing
        if (params == null) { return; }

        // Get the passed parameter values
        String paramVals = params.getString("param");

        Log.d(tag, "Data Param:= " + paramVals);
        Toast.makeText(getApplicationContext(), "Data Param:= " + paramVals, Toast.LENGTH_LONG).show();

        ArrayList<ArrayList<Double>> dataVals = stringArrayToDouble(paramVals);

        XYDataset dataset = createDataset("My Daily Starbucks Allowance", dataVals);
        XYLineChartView graphView = new XYLineChartView(this, chartTitle, xLabel, yLabel, dataset);
        setContentView(graphView);
    }

    private String arrayToString(String[] data)
    {
        StringBuilder strBuilder = new StringBuilder();
        for (int i = 0; i < data.length; i++)
        {
            strBuilder.append(data[i]);
        }
        return strBuilder.toString();
    }

    private String cleanNumericString(String val)
    {
        return val.replaceAll("\\[", "").replaceAll("\\]", "").trim();
    }
}

```

```

private ArrayList<ArrayList<Double>> stringArrayToDouble(String paramVals)
{
    ArrayList<ArrayList<Double>> plotVals = new ArrayList<ArrayList<Double>>();
    if (paramVals.startsWith("[") && paramVals.endsWith("]"))
    {
        String[] vals = paramVals.substring(1, paramVals.length() - 1).split(" , ");
        for (String v : vals)
        {
            if (v.startsWith("[") && v.endsWith("]"))
            {
                String[] dataVals = v.split(",");

                String xvalStr = cleanNumericString(dataVals[0]);
                String yvalStr = cleanNumericString(dataVals[1]);
                Log.d(paramVals, xvalStr + " - " + yvalStr);

                // Convert to Numeric Values
                Double x = Double.parseDouble(xvalStr);
                Double y = Double.parseDouble(yvalStr);

                // Create (x,y) tuple for data point
                ArrayList<Double> list1 = new ArrayList<Double>();
                list1.add(x);
                list1.add(y);

                // Add to our final list
                plotVals.add(list1);
            }
        }
        Log.d(tag, "Values to plot: " + plotVals.toString());
    }
    return plotVals;
}

/**
 * Creates a sample dataset.
 *
 * @return a sample dataset.
 */
private XYDataset createDataset(String title, ArrayList<ArrayList<Double>> dataVals)
{
    final XYSeries series1 = new XYSeries(title);
    for (ArrayList<Double> tuple : dataVals)
    {
        double x = tuple.get(0).doubleValue();
        double y = tuple.get(1).doubleValue();

        series1.add(x, y);
    }

    // Create a collection to hold various data sets
    final XYSeriesCollection dataset = new XYSeriesCollection();
    dataset.addSeries(series1);
    return dataset;
}

```

```

    }

    @Override
    public void onConfigurationChanged(Configuration newConfig)
    {
        super.onConfigurationChanged(newConfig);
        Toast.makeText(this, "Orientation Change", Toast.LENGTH_SHORT);

        // Lets get back to our DemoList view
        Intent intent = new Intent(this, DemoList.class);
        startActivity(intent);

        // Finish current Activity
        this.finish();
    }
}

```

File: DemoList.java

Example 7-40.

```

import java.util.ArrayList;

import android.app.ListActivity;
import android.content.Intent;
import android.content.res.Configuration;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class DemoList extends ListActivity implements OnItemClickListener
{
    private static final String tag = "DemoList";
    private ListView listView;
    private ArrayAdapter<String> listAdapter;

    // Want to pass data values as parameters to next Activity/View/Page
    private String params;

    // Our data for plotting
    private final double[][] data = { { 1, 1.0 }, { 2.0, 4.0 }, { 3.0, 10.0 }, { 4, 2.0 }, { 5.0, 20 },

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

        // Set the View Layer
        setContentView(R.layout.data_listview);
    }
}

```

```

        // Get the Default declared ListView @android:list
        listView = getListView();

        // List for click events to the ListView items
        listView.setOnItemClickListener(this);

        // Get the data to
        ArrayList<String> dataList = getDataStringList(data);

        // Create an Adapter to for viewing the ListView
        listAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, dataList);

        // Bind the adapter to the ListView
        listView.setAdapter(listAdapter);

        // Set the parameters to pass to the next view/ page
        setParameters(data);
    }

private String doubleArrayToString(double[][] dataVals)
{
    StringBuilder strBuilder = new StringBuilder();
    for (int i = 0; i < dataVals.length; i++)
    {
        String datum = "[" + String.valueOf(dataVals[i][0]) + "," + String.valueOf(dataVals[

        if (i < dataVals.length - 1)
        {
            strBuilder.append(datum + " , ");
        }
        else
        {
            strBuilder.append(datum);
        }
    }
    return strBuilder.toString();
}

/**
 * Sets parameters for the Bundle
 *
 * @param dataList
 */
private void setParameters(double[][] dataVals)
{
    params = toJSON(dataVals);
}

public String getParameters()
{
    return this.params;
}

/**
 * Need todo JSONArray

```



```

*
* @param dataVals
* @return
*/
private String toJSON(double[][] dataVals)
{
    StringBuilder strBuilder = new StringBuilder();

    strBuilder.append("[");
    strBuilder.append(doubleArrayToString(dataVals));
    strBuilder.append("]");
    return strBuilder.toString();
}

/**
*
* @param dataVals
* @return
*/
private ArrayList<String> getDataStringList(double[][] dataVals)
{
    ArrayList<String> list = new ArrayList<String>();

    // TODO: CONVERT INTO JSON FORMAT
    for (int i = 0; i < dataVals.length; i++)
    {
        String datum = "[" + String.valueOf(dataVals[i][0]) + "," + String.valueOf(dataVals[
        list.add(datum);
    }
    return list;
}

@Override
public void onConfigurationChanged(Configuration newConfig)
{
    super.onConfigurationChanged(newConfig);

    // Create an Intent to switch view to the next page view
    Intent intent = new Intent(this, DemoCharts.class);

    // Pass parameters along to the next page
    intent.putExtra("param", getParameters());

    // Start the activity
    startActivity(intent);

    Log.d(tag, "Orientation Change...");
    Log.d(tag, "Params: " + getParameters());
}

@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long duration)
{
    // Upon clicking item in list pop a toast
    String msg = "#Item: " + String.valueOf(position) + " - " + listAdapter.getItem(position);

```

```

        Toast.makeText(getApplicationContext(), msg, Toast.LENGTH_LONG).show();
    }
}

```

File: XYLineChartView.java

Example 7-41.

```

import net.droidsolutions.droidcharts.awt.Rectangle2D;
import net.droidsolutions.droidcharts.core.ChartFactory;
import net.droidsolutions.droidcharts.core.JFreeChart;
import net.droidsolutions.droidcharts.core.axis.NumberAxis;
import net.droidsolutions.droidcharts.core.data.XYDataset;
import net.droidsolutions.droidcharts.core.plot.PlotOrientation;
import net.droidsolutions.droidcharts.core.plot.XYPlot;
import net.droidsolutions.droidcharts.core.renderer.xy.XYLineAndShapeRenderer;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.os.Handler;
import android.view.View;

public class XYLineChartView extends View
{
    private final String _chartTitle;
    private final String _xLabel;
    private final String _yLabel;
    private final XYDataset _dataSet;

    /** The view bounds. */
    private final Rect mRect = new Rect();

    /** The user interface thread handler. */
    private final Handler mHandler;

    /**
     * Creates a new graphical view.
     *
     * @param context
     *         the context
     * @param chart
     *         the chart to be drawn
     */
    public XYLineChartView(Context context, String chartTitle, String xLabel, String yLabel, XYDataset dataSet)
    {
        super(context);
        mHandler = new Handler();
        _chartTitle = chartTitle;
        _xLabel = xLabel;
        _yLabel = yLabel;
        _dataSet = dataSet;
    }
}

```

```

@Override
protected void onDraw(Canvas canvas)
{
    super.onDraw(canvas);
    canvas.getClipBounds(mRect);

    // Get the passed in data set
    final XYDataset dataset = _dataSet;

    // Create the Chart
    final JFreeChart chart = createChart(dataset);

    // Draw it
    chart.draw(canvas, new Rectangle2D.Double(0, 0, mRect.width(), mRect.height()));
    Paint p = new Paint();
    p.setColor(Color.RED);
}

/**
 * Schedule a user interface repaint.
 */
public void repaint()
{
    mHandler.post(new Runnable()
    {
        public void run()
        {
            invalidate();
        }
    });
}

/**
 * Creates a chart.
 *
 * @param dataset
 *         the data for the chart.
 *
 * @return a chart.
 */
private JFreeChart createChart(final XYDataset dataset)
{
    // create the chart...
    // (chart title, x-axis label, y-axis label,
    // dataset,orientation,orientation ,url)

    final JFreeChart chart = ChartFactory.createXYLineChart(_chartTitle, _xLabel, _yLabel, dataset,
        ChartFactory.DEFAULT_ORIENTATION, ChartFactory.DEFAULT_ORIENTATION, null);

    Paint white = new Paint(Paint.ANTI_ALIAS_FLAG);
    white.setColor(Color.WHITE);

    Paint dkGray = new Paint(Paint.ANTI_ALIAS_FLAG);
    dkGray.setColor(Color.DKGRAY);
}

```

```

Paint lightGray = new Paint(Paint.ANTI_ALIAS_FLAG);
lightGray.setColor(Color.LTGRAY);
lightGray.setStrokeWidth(10);

// Set Chart Background color
chart.setBackgroundPaint(white);

final XYPlot plot = chart.getXYPlot();

plot.setBackgroundPaint(dkGray);
plot.setDomainGridlinePaint(lightGray);
plot.setRangeGridlinePaint(lightGray);

final XYLineAndShapeRenderer renderer = new XYLineAndShapeRenderer();
renderer.setSeriesLinesVisible(0, true);
plot.setRenderer(renderer);

// change the auto tick unit selection to integer units only...
final NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();
rangeAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits());
// final NumberAxis domainAxis = (NumberAxis) plot.getDomainAxis();
// domainAxis.set(CategoryLabelPositions.STANDARD);

return chart;
}
}

```

Discussion

Handle physical device orientation changes. Android emulator Control-F11 key combination will result in a Portrait to Landscape orientation change. A new View object is created on orientation changes. The Android method `onConfigurationChanged(Configuration newConfig)` can be overridden to accommodate for orientation changes.

Most important trick is to modify the `AndroidManifest.xml` to allow for the following: `android:configChanges="orientation|keyboardHidden" android:screenOrientation="portrait"`

A neat trick to prevent sudden changes in Android Views is to declare/specify the orientations in the `AndroidManifest.xml` file.

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.filefactory.com/file/b43d470/n/AndroidOrientationChanges.zip>

8.1 Introduction: Multimedia

Ian Darwin

Discussion

Android is a rich multimedia environment. The standard Android load includes Music and Video players, and most commercial devices ship with these or fancier versions as well as YouTube players and more. The recipes in this section show you how to control some aspects of the multimedia world that Android provides.

8.2 Play a Youtube Video

Marco Dinacci

Problem

You want to play a video from Youtube on your device

Solution

Given a URI to play the video, create an ACTION_VIEW Intent with it and start a new Activity.

Discussion

For this recipe to work the user needs the standard Youtube application installed on the device.

Example 8-1.

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
}
```

```

String video_path = "http://www.youtube.com/watch?v=opZ69P-0Jbc";
Uri uri = Uri.parse(video_path);

// With this line the Youtube application, if installed, will launch immediately.
// Without it you will be prompted with a list of the application to choose.
uri = Uri.parse("vnd.youtube:" + uri.getQueryParameter("v"));

Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);
}

```

The example uses a standard YouTube.com URL. The `uri.getQueryParameter("v")` is used to extract the video ID from the URI itself, in our example the ID is `opZ69P-0Jbc`.

8.3 Using Gallery with ImageSwitcher

Nidhin Jose Davis

Problem

Creating UI for browse through multiple images.

Solution

You could use Gallery with Image Switcher view to achieve this

Discussion

The `Gallery(android.widget.Gallery)` and `ImageSwitcher(android.widget.ImageSwitcher)` can be used together to create a nice image browser for your application.

Example 8-2.

```

<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

    <ImageSwitcher
        android:id="@+id/switcher"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        />

    <Gallery
        android:id="@+id/gallery"

```

```

        android:background="#55000000"
        android:layout_width="fill_parent"
        android:layout_height="60dip"
        android:spacing="16px"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:gravity="center_vertical"
    />

```

</RelativeLayout>

Now let's see how to use this layout.

Example 8-3.

```

public class ImageBrowser extends Activity implements AdapterView.OnItemClickListener, ViewSwitcher.ViewF
    private ImageSwitcher mISwitcher;
    private ArrayList<Drawable> allimages = new ArrayList<Drawable>();

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // lets remove the title bar
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.main);

        getImages();

        mISwitcher = (ImageSwitcher)findViewById(R.id.switcher);
        mISwitcher.setFactory(this);
        // some animation when image changes
        mISwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.fade_in));
        mISwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.fade_out));

        Gallery gallery = (Gallery) findViewById(R.id.gallery);
        gallery.setAdapter(new ImageAdapter(this));
        gallery.setOnItemClickListener(this);
    }

    private void getImages() {
        allimages.add(this.getResources().getDrawable(R.drawable.image1));
        allimages.add(this.getResources().getDrawable(R.drawable.image2));
        allimages.add(this.getResources().getDrawable(R.drawable.image3));
        allimages.add(this.getResources().getDrawable(R.drawable.image4));
        allimages.add(this.getResources().getDrawable(R.drawable.image5));
        allimages.add(this.getResources().getDrawable(R.drawable.image6));
        allimages.add(this.getResources().getDrawable(R.drawable.image7));
        allimages.add(this.getResources().getDrawable(R.drawable.image8));
        allimages.add(this.getResources().getDrawable(R.drawable.image9));
    }

```



```

@Override
public void onItemClick(AdapterView<?> arg0, View v, int position, long id) {
    try{
        mISwitcher.setImageDrawable(allimages.get(position));
    }catch(Exception e){}
}

@Override
public void onNothingSelected(AdapterView<?> arg0) {
    // TODO Auto-generated method stub

}

@Override
public View makeView() {
    ImageView i = new ImageView(this);
    i.setBackgroundColor(0xFF000000);
    i.setScaleType(ImageView.ScaleType.FIT_CENTER);
    i.setLayoutParams(new ImageSwitcher.LayoutParams(ImageSwitcher.LayoutParams.FILL_PARENT,ImageSwitcher.LayoutParams.WRAP_CONTENT));
    return i;
}

public class ImageAdapter extends BaseAdapter {
    private Context mContext;

    public ImageAdapter(Context c) {
        mContext = c;
    }

    public int getCount() {
        return allimages.size();
    }

    public Object getItem(int position) {
        return position;
    }

    public long getItemId(int position) {
        return position;
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView galleryview = new ImageView(mContext);
        galleryview.setImageDrawable(allimages.get(position));
        galleryview.setAdjustViewBounds(true);
        galleryview.setLayoutParams(new LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
        galleryview.setPadding(5, 0, 5, 0);
        galleryview.setBackgroundResource(android.R.drawable.picture_frame);
        return galleryview;
    }
}

```

```
}  
}
```

8.4 Grabbing a video using MediaRecorder

Marco Dinacci

Problem

You want to grab a video using the built-in device camera and save it to disk.

Solution

This recipe teaches the reader how to grab a video and record it on the phone by using the MediaRecorder class provided by the Android framework.

Discussion

The MediaRecorder is normally used to perform audio and/or video recording. The class has a straightforward API but as it's based on a simple state machine the methods must be called in the proper order in order to avoid IllegalStateException from popping up.

Create a new Activity and override the **onCreate** method with the following:

Example 8-4.

```
@Override  
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.media_recorder_recipe);  
  
    // we shall take the video in landscape orientation  
    setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);  
  
    mSurfaceView = (SurfaceView) findViewById(R.id.surfaceView);  
    mHolder = mSurfaceView.getHolder();  
    mHolder.addCallback(this);  
    mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);  
  
    mToggleButton = (ToggleButton) findViewById(R.id.toggleRecordingButton);  
    mToggleButton.setOnClickListener(new OnClickListener() {  
        @Override  
        // toggle video recording  
        public void onClick(View v) {  
            if (((ToggleButton)v).isChecked())  
                mMediaRecorder.start();  
            else {  
                mMediaRecorder.stop();  
                mMediaRecorder.reset();  
                try {  
                    initRecorder(mHolder.getSurface());  
                }  
            }  
        }  
    });  
}
```

```

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
});
}

```

The preview frames from the camera will be displayed on a **SurfaceView**. Recording is controlled by a toggle button. After the recording is over, we stop the `MediaRecorder`. Since the **stop** method resets all the state machine variables in order to be able to grab another video we reset the state machine and call our **initRecorder** once more.

`initRecorder` is where we configure the `MediaRecorder` and the camera:

Example 8-5.

```

/* Init the MediaRecorder, the order the methods are called is vital to
 * its correct functioning.
 */
private void initRecorder(Surface surface) throws IOException {
    // It is very important to unlock the camera before doing setCamera
    // or it will results in a black preview
    if(mCamera == null) {
        mCamera = Camera.open();
        mCamera.unlock();
    }

    if(mMediaRecorder == null)
        mMediaRecorder = new MediaRecorder();

    mMediaRecorder.setPreviewDisplay(surface);
    mMediaRecorder.setCamera(mCamera);

    mMediaRecorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
    mMediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.DEFAULT);
    File file = createFile();

    mMediaRecorder.setOutputFile(file.getAbsolutePath());

    // No limit. Don't forget to check the space on disk.
    mMediaRecorder.setMaxDuration(-1);
    mMediaRecorder.setVideoFrameRate(15);

    mMediaRecorder.setVideoEncoder(MediaRecorder.VideoEncoder.DEFAULT);

    try {
        mMediaRecorder.prepare();
    } catch (IllegalStateException e) {
        // This is thrown if the previous calls are not called with the
        // proper order
        e.printStackTrace();
    }
}

```

```

        mInitSuccessful = true;
    }

```

It is important to create and unlock a Camera object before of the creation of a MediaRecorder. setPreviewDisplay and setCamera must be called immediately after the creation of the MediaRecorder. The choice of the format and the output file is obligatory. Other options are optional but they must be called in the order outlined in the code above.

The MediaRecorder is best initialized when the surface has been created. We register our Activity as a SurfaceHolder.Callback listener in order to be notified of this and override the surfaceCreated method to call our initialization code:

Example 8-6.

```

@Override
public void surfaceCreated(SurfaceHolder holder) {
    try {
        if(!mInitSuccessful)
            initRecorder(mHolder.getSurface());
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

```

When you're done with the surface, don't forget to release the resources, the camera is a shared object and may be used by other applications as well:

Example 8-7.

```

private void shutdown() {
    // Release MediaRecorder and especially the Camera as it's a shared
    // object that can be used by other applications
    mMediaRecorder.reset();
    mMediaRecorder.release();
    mCamera.release();

    // once the objects have been released they can't be reused
    mMediaRecorder = null;
    mCamera = null;
}

```

Override the surfaceDestroyed method so the previous code can be called automatically when the user is done with the Activity:

Example 8-8.

```

@Override
public void surfaceDestroyed(SurfaceHolder holder) {
    shutdown();
}

```

Source Download URL

The source code for this example may be downloaded from this URL: http://www.intransitione.com/intransitione.com/code/android/media_recorder_recipe_code.zip

8.5 Android Face Detection

Wagied Davids

Problem

Face detection is a cool and fun hidden api feature of Android, and has been around since Android 1.5. In essence face detection is part of a machine learning technique of recognizing objects using a set of features.

Solution

Use Android's built-in face detection capability.

Discussion

File: Main.java

Example 8-9.

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

public class Main extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(new FaceDetectionView(this, "face5.JPG"));
    }
}
```

File: FaceDetectionView.java

Example 8-10.

```
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.PointF;
import android.graphics.Rect;
import android.media.FaceDetector;
```

```

import android.util.Log;
import android.view.View;

/**
 * @author wdauid01
 *
 */
public class FaceDetectionView extends View
{
    private static final String tag = FaceDetectionView.class.getName();
    private static final int NUM_FACES = 10;
    private FaceDetector arrayFaces;
    private final FaceDetector.Face getAllFaces[] = new FaceDetector.Face[NUM_FACES];
    private FaceDetector.Face getFace = null;

    private final PointF eyesMidPts[] = new PointF[NUM_FACES];
    private final float eyesDistance[] = new float[NUM_FACES];

    private Bitmap sourceImage;

    private final Paint tmpPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
    private final Paint pOuterBullsEye = new Paint(Paint.ANTI_ALIAS_FLAG);
    private final Paint pInnerBullsEye = new Paint(Paint.ANTI_ALIAS_FLAG);

    private int picWidth, picHeight;
    private float xRatio, yRatio;
    private ImageLoader mImageLoader = null;

    public FaceDetectionView(Context context, String imagePath)
    {
        super(context);
        init();
        mImageLoader = ImageLoader.getInstance(context);
        sourceImage = mImageLoader.loadFromFile(imagePath);
        detectFaces();
    }

    private void init()
    {
        Log.d(tag, "Init()...");
        pInnerBullsEye.setStyle(Paint.Style.FILL);
        pInnerBullsEye.setColor(Color.RED);
        pOuterBullsEye.setStyle(Paint.Style.STROKE);
        pOuterBullsEye.setColor(Color.RED);
        tmpPaint.setStyle(Paint.Style.STROKE);
        tmpPaint.setTextAlign(Paint.Align.CENTER);
        BitmapFactory.Options bfo = new BitmapFactory.Options();
        bfo.inPreferredConfig = Bitmap.Config.RGB_565;
    }

    private void loadImage(String imagePath)
    {
        sourceImage = mImageLoader.loadFromFile(imagePath);
    }

```

```

}

@Override
protected void onDraw(Canvas canvas)
{
    Log.d(tag, "onDraw(...)");

    xRatio = getWidth() * 1.0f / picWidth;
    yRatio = getHeight() * 1.0f / picHeight;
    canvas.drawBitmap(sourceImage, null, new Rect(0, 0, getWidth(), getHeight()), tmpPaint);
    for (int i = 0; i < eyesMidPts.length; i++)
    {
        if (eyesMidPts[i] != null)
        {
            pOuterBullsEye.setStrokeWidth(eyesDistance[i] / 6);
            canvas.drawCircle(eyesMidPts[i].x * xRatio, eyesMidPts[i].y * yRatio, eyesDistance[i] / 2, pOuterBullsEye);
            canvas.drawCircle(eyesMidPts[i].x * xRatio, eyesMidPts[i].y * yRatio, eyesDistance[i] / 6, pInnerBullsEye);
        }
    }

    private void detectFaces()
    {
        Log.d(tag, "detectFaces(...)");

        picWidth = sourceImage.getWidth();
        picHeight = sourceImage.getHeight();

        arrayFaces = new FaceDetector(picWidth, picHeight, NUM_FACES);
        arrayFaces.findFaces(sourceImage, getAllFaces);

        for (int i = 0; i < getAllFaces.length; i++)
        {
            getFace = getAllFaces[i];
            try
            {
                PointF eyesMP = new PointF();
                getFace.getMidPoint(eyesMP);
                eyesDistance[i] = getFace.eyesDistance();
                eyesMidPts[i] = eyesMP;

                Log.i("Face", i + " " + getFace.confidence() + " " + getFace.eyesDistance() + " " + "Pose: (" + getFace.pose

            }
            catch (Exception e)
            {
                Log.e("Face", i + " is null");
            }
        }
    }
}

```

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.filefactory.com/file/cbf2e8b/n/AndroFaces-Example.zip>

8.6 Playing audio from a file

Marco Dinacci

Problem

You want to play an audio file stored on the device.

Solution

Create and configure properly a `MediaPlayer` and a `MediaController`, provide the path of the audio file to play and enjoy the music.

Discussion

Playing an audio file is as easy as setting up a `MediaPlayer` and a `MediaController`.

First create a new activity that implements the `MediaPlayerControl` interface.

Example 8-11.

```
public class PlayAudioActivity extends Activity implements MediaPlayerControl {
    private MediaController mMediaController;
    private MediaPlayer mMediaPlayer;
    private Handler mHandler = new Handler();
```

In the `onCreate` method we create and configure a `MediaPlayer` and a `MediaController`. The first is the object that perform the typical operations on an audio file like playing, pausing and seeking. The second is a view containing the buttons that launch the just mentioned operations through our `MediaPlayerControl` class.

Let's see the `onCreate` code:

Example 8-12.

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

    mMediaPlayer = new MediaPlayer();
    mMediaController = new MediaController(this);
    mMediaController.setMediaPlayer(PlayAudioActivity.this);
    mMediaController.setAnchorView(findViewById(R.id.audioView));

    String audioFile = "";
    try {
```



```

        mMediaPlayer.setDataSource(audioFile);
        mMediaPlayer.prepare();
    } catch (IOException e) {
        Log.e("PlayAudioDemo", "Could not open file " + audioFile + " for playback.", e);
    }

    mMediaPlayer.setOnPreparedListener(new OnPreparedListener() {
        @Override
        public void onPrepared(MediaPlayer mp) {
            mHandler.post(new Runnable() {
                public void run() {
                    mMediaController.show(10000);
                    mMediaPlayer.start();
                }
            });
        }
    });
}

```

In addition to configuring our `MediaController` and `MediaPlayer` we create an anonymous `OnPreparedListener` in order to start the player only when the media source is ready for playback.

Remember to cleanup the media player when the `Activity` is destroyed.

Example 8-13.

```

@Override
protected void onDestroy() {
    super.onDestroy();
    mMediaPlayer.stop();
    mMediaPlayer.release();
}

```

At last we implement the `MediaPlayerControl` interface. The code is very straightforward:

Example 8-14.

```

@Override
public boolean canPause() {
    return true;
}

@Override
public boolean canSeekBackward() {
    return false;
}

@Override
public boolean canSeekForward() {
    return false;
}

@Override

```

```

public int getBufferPercentage() {
    int percentage = (mMediaPlayer.getCurrentPosition() * 100) / mMediaPlayer.getDuration();

    return percentage;
}

@Override
public int getCurrentPosition() {
    return mMediaPlayer.getCurrentPosition();
}

@Override
public int getDuration() {
    return mMediaPlayer.getDuration();
}

@Override
public boolean isPlaying() {
    return mMediaPlayer.isPlaying();
}

@Override
public void pause() {
    if(mMediaPlayer.isPlaying())
        mMediaPlayer.pause();
}

@Override
public void seekTo(int pos) {
    mMediaPlayer.seekTo(pos);
}

@Override
public void start() {
    mMediaPlayer.start();
}

@Override
public boolean onTouchEvent(MotionEvent event) {
    mMediaController.show();

    return false;
}
}

```

As a final touch we override the `onTouchEvent` in order to show the `MediaController` buttons when the user click on the screen.

Since we create our `MediaController` programmatically, the layout is very simple:

Example 8-15.

```

<?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"
        android:id="@+id/audioView"
    >
</LinearLayout>
```

Source Download URL

The source code for this example may be downloaded from this URL: http://www.intransitio.com/intransitio.com/code/android/play_audio_demo.zip

8.7 Playing Audio without Interaction

Ian Darwin

Problem

You want to play an audio file with no interaction.

Solution

All you need to play a file with no interaction (e.g., not user-settable volume, pause, etc. controls) is to create a `MediaPlayer` for the file, and call its `start()` method.

Discussion

This is the simplest way to play a sound file. In contrast with the recipe [Recipe 8.6](#), this version offers the user no controls to interact with the sound. You should therefore usually offer at least a "stop" or "cancel" button, especially if the audio file is or might be long. If you're just playing a short sound effect within your application, no such control is needed.

You must have a `MediaPlayer` created for your file. The audio file may be on the SD Card or it may be in your application's `res/raw` directory. If the sound file is part of your application, store it under `res/raw`. Suppose it is in `res/raw/alarm_sound.3gp`. Then the reference to it is `R.raw.alarm_sound`, and you can play it as follows:

Example 8-16.

```
MediaPlayer player = MediaPlayer.create(this, R.raw.alarm_sound);
player.start();
```

In the SD Card case, use the following invocation:

Example 8-17.

```
MediaPlayer player = new MediaPlayer();
player.setDataSource(fileName);
player.prepare();
player.start();
```

There is also a convenience routine, `MediaPlayer.create(Context, URI)` that can be used; in all cases, `create()` calls `prepare()` for you.

To control the play from within your application, you can call the relevant methods such as `player.stop()`, `player.pause()`, etc. If you want to reuse a player after stopping it, you must call `prepare()` again.

To be notified when the audio is finished, use an `OnCompletionListener`:

Example 8-18.

```
player.setOnCompletionListener(new OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mp) {
        Toast.makeText(Main.this,
            "Media Play Complete", Toast.LENGTH_SHORT).show();
    }
});
```

When you are truly done with any `MediaPlayer` instance, you should call its `release()` method to free up memory, or you will run out of resources if you are creating a lot of `MediaPlayer` objects.

See Also

To really use the `MediaPlayer` effectively you should understand its various states and transitions, as this will help you to understand what methods are valid. There is a complete state diagram for the `MediaPlayer` at <http://developer.android.com/reference/android/media/MediaPlayer.html>.

Source Download URL

The source code for this example may be downloaded from this URL: <http://projects.darwinsys.com/MediaPlayerDemo..tgz>

8.8 Using Speech to Text

Corey Sunwold

Problem

How to get speech input and display it as text

Solution

One of Android's unique features is native speech to text processing. This provides an alternative form of text input for the user, who in some situations might not have their hands readily available to type in information.

Discussion

Android provides an easy API for using its built in voice recognition through the `RecognizerIntent`.

The example layout will be very simple. I've only included a `TextView` and a `Button`. The button will be used to launch the voice recognizer, and when results are returned they will be displayed in the `TextView`.

Example 8-19.

```
<?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"
    >
<Button
    android:id="@+id/getSpeechButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Press to begin voice recognition"
    ></Button>
<TextView
    android:id="@+id/speechText"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text=""
    />
</LinearLayout>
```

Example 8-20.

```
public class Main extends Activity {

    private static final int RECOGNIZER_RESULT = 1234;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button startSpeech = (Button)findViewById(R.id.getSpeechButton);
        startSpeech.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
                intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
                    RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
                intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Speech to text");
                startActivityForResult(intent, RECOGNIZER_RESULT);
            }
        });
    }
}
```

```

    });
}

/**
 * Handle the results from the recognition activity.
 */
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == RECOGNIZER_RESULT && resultCode == RESULT_OK) {
        ArrayList<String> matches = data.getStringArrayListExtra(
            RecognizerIntent.EXTRA_RESULTS);

        TextView speechText = (TextView)findViewById(R.id.speechText);
        speechText.setText(matches.get(0).toString());
    }

    super.onActivityResult(requestCode, resultCode, data);
}
}

```

See Also

<http://developer.android.com/reference/android/speech/RecognizerIntent.html>

8.9 Making the Device Speak with TTS

Ian Darwin

Problem

You want your application to pronounce words of text so the user can perceive them without watching the screens (e.g., when driving).

Solution

Use the TextToSpeech api.

Discussion

The TextToSpeech API is built in to Android (though you may have to install the voice files, depending on version).

To get started you just need a TextToSpeech object. In theory you could just do this:

Example 8-21.

```

private TextToSpeech myTTS = new TextToSpeech(this, this);
myTTS.setLanguage(Locale.US);
myTTS.speak(textToBeSpoken, TextToSpeech.QUEUE_FLUSH, null);
myTTS.shutdown();

```

However, to ensure success, you actually have to use a couple of intents, one to check that the TTS data are available and/or install them if not, and another to start the TTS mechanism. So in practice the code needs to look something like the following. This quaint little application chooses one of half a dozen banal phrases to utter each time the Speak button is pressed.

Example 8-22.

```
public class Main extends Activity implements OnInitListener {

    private TextToSpeech myTTS;
    private List<String> phrases = new ArrayList<String>();

    public void onCreate(Bundle savedInstanceState) {

        phrases.add("Hello Android, Goodbye iPhone");
        phrases.add("The quick brown fox jumped over the lazy dog");
        phrases.add("What is your mother's maiden name?");
        phrases.add("Etaoin Shrdlu for Prime Minister");
        phrases.add("The letter 'Q' does not appear in 'antidisestablishmentarianism')");
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button startButton = (Button) findViewById(R.id.start_button);
        startButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View arg0) {
                Intent checkIntent = new Intent();
                checkIntent.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
                startActivityForResult(checkIntent, 1);
            }
        });
    }

    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (requestCode == 1) {

            if (resultCode == TextToSpeech.Engine.CHECK_VOICE_DATA_PASS) {
                myTTS = new TextToSpeech(this, this); // 1
                myTTS.setLanguage(Locale.US);
            } else {
                // TTS data not yet loaded, try to install it
                Intent ttsLoadIntent = new Intent();
                ttsLoadIntent.setAction(TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);
                startActivity(ttsLoadIntent);
            }
        }
    }

    public void OnInit(int status) {
        if (status == TextToSpeech.SUCCESS) {

            int n = (int)(Math.random() * phrases.size());
            myTTS.speak(phrases.get(n), TextToSpeech.QUEUE_FLUSH, null);
        }
    }
}
```

```
    } else if (status == TextToSpeech.ERROR) {  
        myTTS.shutdown();  
    }  
}
```

At the line marked "1", midway through the above code, the first argument is a Context (the Activity) and the second is an OnInitListener, also implemented by the Main Activity in this case. When the initialization of the TextToSpeech object is done, it calls the listener, whose onInit() method is meant to notify that the TTS is ready. In our trivial Speaker program here, we simply do the speaking. In a longer example you would probably want to start a thread or service to do the speaking operation.

Source Download URL

The source code for this example may be downloaded from this URL: <http://projects.darwinsys.com/Speaker..zip>

Data Persistence

9.1 Listing a Directory

Ian Darwin

Problem

You need to list the filesystem entries named in a directory.

Solution

Use a `java.io.File` object's `list()` or `listFiles()` method.

Discussion

The `java.io.File` class contains several methods for working with directories. For example, to list the filesystem entities named in the current directory, just write:

Example 9-1.

```
String<ulink url=<emphasis><citetitle></citetitle></ulink> list = new File(".").list()
```

To get an array of already constructed `File` objects rather than `Strings`, use:

Example 9-2.

```
File<ulink url=<emphasis><citetitle></citetitle></ulink> list = new File(".").listFiles();
```

You can display the result in a (BROKEN XREF TO RECIPE -1 'Building list-based applications with `ListView|ListView`').

Of course, there's lots of room for elaboration. You could print the names in multiple columns across or down the screen in a `TextView` in a monospace font, since you know the number of items in the list before you print. You could omit filenames with leading periods, as does the Unix `ls` program. Or print the directory names first; as some "file manager" type programs do. By using `listFiles()`, which constructs a new `File` object for

each name, you could print the size of each, as per the DOS `dir` command or the Unix `ls -l` command (see (BROKEN XREF TO RECIPE -1 'Getting File Information')). Or you could figure out whether each is a file, a directory, or neither. Having done that, you could pass each directory to your top-level function, and you'd have directory recursion (the Unix `find` command, or `ls -R`, or the DOS `DIR /S` command). Quite the makings of a file manager application of your own.

A more flexible way to list filesystem entries is with `list(FilenameFilter ff)`. `FilenameFilter` is a tiny interface with only one method: `boolean accept(File inDir, String fileName)`. Suppose you want a listing of only Java-related files (`*.java`, `*.class`, `*.jar`, etc.). Just write the `accept()` method so that it returns `true` for these files and `false` for any others. Here is the `Ls` class warmed over to use a `FilenameFilter` instance

Example 9-3.

```
import java.io.*;

/**
 * FNFilter - directory lister modified to use FilenameFilter
 */
public class FNFilter {
    public static String[] getListing(String startingDir) {
        // Generate the selective list, with a one-use File object.
        String[] dir = new java.io.File(startingDir).list(new OnlyJava());
        java.util.Arrays.sort(dir); // Sort it (Data Structuring chapter)
        return dir;
    }

    /** FilenameFilter implementation:
     * The Accept method only returns true for .java , .jar and class files.
     */
    class OnlyJava implements FilenameFilter {
        public boolean accept(File dir, String s) {
            if (s.endsWith(".java") || s.endsWith(".dex") || s.endsWith(".jar"))
                return true;
            // others: projects, ... ?
            return false;
        }
    }
}
```

The `FilenameFilter` could be more flexible; in a full-scale application, the list of files returned by the `FilenameFilter` would be chosen dynamically, possibly automatically, based on what you were working on. File Chooser dialogs implement this as well, allowing the user to select interactively from one of several sets of files to be listed. This is a great convenience in finding files, just as it is here in reducing the number of files that must be examined.

For the `listFiles()` method, there is an additional overload that accepts a `FileFilter`. The only difference is that `FileFilter`'s `accept` method is called with a `File` object, whereas `FileNameFilter`'s is called with a filename `String`.

See Also

See [Recipe 7.2](#) to display the results in your GUI. Chapter 11 of the *Java Cookbook* has more information on file and directory operations.

9.2 Default shared preferences consistency check

Federico Paolinelli

Problem

Android provides a very easy way to setup default preferences by defining a `PreferenceActivity` and providing it a resource file. What is not clear is how to perform checks on preferences given by the user

Solution

You can implement the

Example 9-4.

```
public void onSharedPreferenceChanged(SharedPreferences prefs, String key)
```

and perform the checks in its body. If the check fails you can restore a default value in the preference. You must be aware that even if the `SharedPreferences` will contain the right value, you won't see it displayed correctly. For this reason, you need to reload the preferences activity.

Discussion

If you have a default preference activity that implements `OnSharedPreferenceChangeListener`

Example 9-5.

```
public class MyPreferenceActivity extends PreferenceActivity implements OnSharedPreferenceChangeListener {  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        Context context = getApplicationContext();  
        prefs = PreferenceManager.getDefaultSharedPreferences(context);  
        addPreferencesFromResource(R.xml.userprefs);  
    }  
}
```

Your `PreferenceActivity` can implement the `onSharedPreferenceChanged` method.

This will be called after the change is committed, so every other change you perform will be permanent.

The idea is to check if you like the value, and otherwise put a default value / disable it.

To get the method notified, you have to register your activity as a valid listener. The better way is to register in `onResume` and unregister in `onPause`:

Example 9-6.

```
@Override
protected void onResume() {
    super.onResume();
    prefs.registerOnSharedPreferenceChangeListener(this);
}

@Override
protected void onPause() {
    super.onPause();
    prefs.unregisterOnSharedPreferenceChangeListener(this);
}
```

Now it's time to perform the consistency check. For example, if you have an option whose key is `MY_OPTION_KEY`,

Example 9-7.

```
public void onSharedPreferenceChanged(SharedPreferences prefs, String key) {
    SharedPreferences.Editor prefEditor = prefs.edit();

    if(key.equals(MY_OPTION_KEY)){
        String optionValue = prefs.getString(MY_OPTION_KEY, "");
        if(dontLikeTheValue(optionValue)){
            prefEditor.putString(MY_OPTION_KEY, "Default value");
            prefEditor.commit();
            reload();
        }
    }
    return;
}
```

Of course in this way the user will be surprised and will not know why you refused his option. You can then show an error dialog and perform the reload action after the user confirms the dialog.

Example 9-8.

```
private void showErrorDialog(String errorString){
    String okButtonString = context.getString(R.string.ok_name);
    AlertDialog.Builder ad = new AlertDialog.Builder(context);
    ad.setTitle(context.getString(R.string.error_name));
    ad.setMessage(errorString);
    ad.setPositiveButton(okButtonString, new OnClickListener() {
        public void onClick(DialogInterface dialog, int arg1) {
            reload();
        } } );
    ad.show();
}
```

```
    return;  
}
```

In this way the `iDontLikeTheValue` if becomes:

Example 9-9.

```
    if(dontLikeTheValue(optionValue)){  
        if(!GeneralUtils.isPhoneNumber(smsNumber)){  
            showErrorDialog("I dont like the option");  
            prefEditor.putString(MY_OPTION_KEY, "Default value");  
            prefEditor.commit();  
        }  
    }
```

What's still missing is the `reload()` function, but it's pretty obvious. It relaunches the activity using the same intent that fired it

Example 9-10.

```
private void reload(){  
    startActivity(getIntent());  
    finish();  
}
```

9.3 Advanced text search

Claudio Esperanca

Problem

How to build a data layer to store and search text data using wildcards or expressions like `and`, `or`, `not`, etc.

Solution

Using a SQLite Full Text Search 3 (FTS3) virtual table and match function from SQLite it's possible to build such mechanism.

Discussion

Following these steps you will be able to create an example android project with a data layer where you will be able to store and retrieve some data using the SQLite.

1. Create a new Android Project (AdvancedSearchProject)
2. # Select an API level equal or greater than 8
3. # Specify `AdvancedSearch` as the application name
4. # Use `com.androidcookbook.example.advancedsearch` as the package name
5. # Create an activity with the name `AdvancedSearchActivity`

6. # The Min SDK version should be 8 (for the android 2.2 - codename froyo)
7. Create a new Java class DbAdapter within the package `com.androidcookbook.example.advancedsearch` on the `src` folder.

To create the data layer for the example application, enter the following source code in the created file:

Example 9-11.

```
package com.androidcookbook.example.advancedsearch;

import java.util.LinkedList;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class DbAdapter {
    public static final String APP_NAME = "AdvancedSearch";
    private static final String DATABASE_NAME = "AdvancedSearch_db";
    private static final int DATABASE_VERSION = 1; // Our internal database version (e.g. to control upgrade)
    private static final String TABLE_NAME = "example_tbl";
    public static final String KEY_USERNAME = "username";
    public static final String KEY_FULLNAME = "fullname";
    public static final String KEY_EMAIL = "email";
    public static long GENERIC_ERROR = -1;
    public static long GENERIC_NO_RESULTS = -2;
    public static long ROW_INSERT_FAILED = -3;
    private final Context context;
    private DbHelper dbHelper;
    private SQLiteDatabase sqlDatabase;

    public DbAdapter(Context context) {
        this.context = context;
    }

    private static class DbHelper extends SQLiteOpenHelper {
        private boolean databaseCreated=false;
        DbHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }
        @Override
        public void onCreate(SQLiteDatabase db) {
            Log.d(APP_NAME, "Creating the application database");

            try{
                // Create the full text search 3 virtual table
                db.execSQL(
                    "CREATE VIRTUAL TABLE ["+TABLE_NAME+"] USING FTS3 (" +
                    "["+KEY_USERNAME+"] TEXT," +

```

```

        "["+KEY_FULLNAME+"] TEXT," +
        "["+KEY_EMAIL+"] TEXT" +
        ");"
    );
    this.databaseCreated = true;
} catch (Exception e) {
    Log.e(APP_NAME, "An error occurred while creating the database: "+e.toString(), e);
    this.deleteDatabaseStructure(db);
}
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    Log.d(APP_NAME, "Updating the database from the version " + oldVersion + " to " + newVersion + " ");
    this.deleteDatabaseStructure(db); // in this example we purge the previous data on upgrade
    this.onCreate(db);
}
public boolean databaseCreated(){
    return this.databaseCreated;
}
private boolean deleteDatabaseStructure(SQLiteDatabase db){
    try{
        db.execSQL("DROP TABLE IF EXISTS ["+TABLE_NAME+"];");

        return true;
    }catch (Exception e) {
        Log.e(APP_NAME, "An error occurred while deleting the database: "+e.toString(), e);
    }
    return false;
}
}

/**
 * Open the database; if the database can't be opened, try to create it
 *
 * @return {@link Boolean} true if the database was successfully opened/created, false otherwise
 * @throws {@link SQLException} if an error occurred
 */
public boolean open() throws SQLException {
    try{
        this.dbHelper = new DbHelper(this.context);
        this.sqlDatabase = this.dbHelper.getWritableDatabase();
        return this.sqlDatabase.isOpen();
    }catch (SQLException e) {
        throw e;
    }
}

/**
 * Close the database connection
 * @return {@link Boolean} true if the connection was terminated, false otherwise
 */
public boolean close() {
    this.dbHelper.close();
    return !this.sqlDatabase.isOpen();
}
}

```



```

/**
 * Check if the database is opened
 *
 * @return {@link Boolean} true if it was, false otherwise
 */
public boolean isOpen(){
    return this.sqlDatabase.isOpen();
}

/**
 * Check if the database was created
 *
 * @return {@link Boolean} true if it was, false otherwise
 */
public boolean databaseCreated(){
    return this.dbHelper.databaseCreated();
}

/**
 * Insert a new row on the table
 *
 * @param username {@link String} with the username
 * @param fullname {@link String} with the fullname
 * @param email {@link String} with the email
 * @return {@link Long} with the row id or ROW_INSERT_FAILED (bellow 0 value) on error
 */
public long insertRow(String username, String fullname, String email) {
    try{
        // Prepare the values
        ContentValues values = new ContentValues();
        values.put(KEY_USERNAME, username);
        values.put(KEY_FULLNAME, fullname);
        values.put(KEY_EMAIL, email);

        // Try to insert the row
        return this.sqlDatabase.insert(TABLE_NAME, null, values);
    }catch (Exception e) {
        Log.e(APP_NAME, "An error occurred while inserting the row: "+e.toString(), e);
    }
    return ROW_INSERT_FAILED;
}

/**
 * The search method
 * Uses the full text search 3 virtual table and the MATCH function from SQLite to search for data
 * @see http://www.sqlite.org/fts3.html to know more about the syntax
 * @param search {@link String} with the search expression
 * @return {@link LinkedList} with the {@link String} search results
 */
public LinkedList<String> search(String search) {
    LinkedList<String> results = new LinkedList<String>();
    Cursor cursor = null;

```

```

try{
    cursor = this.sqlDatabase.query(true, TABLE_NAME, new String[] { KEY_USERNAME, KEY_FULLNAME, KEY_EMAIL }, null, null, null, null, null, null);

    if(cursor!=null && cursor.getCount()>0 && cursor.moveToFirst()){
        int iUsername = cursor.getColumnIndex(KEY_USERNAME);
        int iFullname = cursor.getColumnIndex(KEY_FULLNAME);
        int iEmail = cursor.getColumnIndex(KEY_EMAIL);

        do{
            results.add(
                new String(
                    "Username: "+cursor.getString(iUsername) +
                    ", Fullname: "+cursor.getString(iFullname) +
                    ", Email: "+cursor.getString(iEmail)
                )
            );
        }while(cursor.moveToNext());
    }
}catch(Exception e){
    Log.e(APP_NAME, "An error occurred while searching for "+search+": "+e.toString(), e);
}finally{
    if(cursor!=null && !cursor.isClosed()){
        cursor.close();
    }
}

return results;
}
}

```

Now that the data layer is usable, the activity `AdvancedSearchActivity` can be used to test it.

1. To define the application strings, replace the contents of the `res/values/strings.xml` file:

Example 9-12.

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="label_search">Search</string>
    <string name="app_name">AdvancedSearch</string>
</resources>

```

2. The application layout can be set within the file `res/layout/main.xml`:

Example 9-13.

```

<?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"
    >
    <EditText

```

```

        android:text=""
        android:id="@+id/etSearch"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:singleLine="true"
    />
    <Button
        android:text="@string/label_search"
        android:id="@+id/btnSearch"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
    <TextView
        android:id="@+id/tvResults"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:text=""
        android:singleLine="false"
    />
</LinearLayout>

```

3. To finish, replace the contents of the `AdvancedSearchActivity.java` file by the following code:

Example 9-14.

```

package com.androidcookbook.example.advancedsearch;

import java.util.Iterator;
import java.util.LinkedList;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class AdvancedSearchActivity extends Activity {
    private DbAdapter dbAdapter;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        dbAdapter = new DbAdapter(this);
        dbAdapter.open();

        if(dbAdapter.databaseCreated()){
            dbAdapter.insertRow("test", "test example", "example_test@example.com");
            dbAdapter.insertRow("lorem", "lorem ipsum", "lorem.ipsum@example2.com");
            dbAdapter.insertRow("jdoe", "Jonh Doe", "j.doe@example.com");
        }

        Button button = (Button) findViewById(R.id.btnSearch);

```

```

final EditText etSearch = (EditText) findViewById(R.id.etSearch);
final TextView tvResults = (TextView) findViewById(R.id.tvResults);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        LinkedList<String> results = dbAdapter.search(etSearch.getText().toString());

        if(results.isEmpty()){
            tvResults.setText("No results found");
        }else{
            Iterator<String> i = results.iterator();
            tvResults.setText("");
            while(i.hasNext()){
                tvResults.setText(tvResults.getText()+i.next()+"\n");
            }
        }
    }
});
}
@Override
protected void onDestroy() {
    dbAdapter.close();
    super.onDestroy();
}
}

```

See Also

<http://www.sqlite.org/fts3.html> - to know more about the Full Text Search 3, including the search syntax

<http://code.google.com/p/localizeandroid/> - project with an implementation of this search mechanism

9.4 How to push string-values using Intent.putExtra()

Ulysses Levy

Problem

You need to pass some parameters into an activity while launching it.

Solution

A quick solution is to use Intent.putExtra() to push our data. And then use getIntent().getExtras().getString() to retrieve it.

Discussion

Push Data

Example 9-15.

```
import android.content.Intent;

...

Intent intent =
    new Intent(
        this,
        MyActivity.class );
intent.putExtra( "paramName", "paramValue" );
startActivity( intent );
```

The above code might be inside the main activity. "MyActivity.class" is the second Activity we want to launch; it must be explicitly included in your AndroidManifest.xml file.

Example 9-16.

```
<activity android:name=".MyActivity" />
```

Pull Data

Example 9-17.

```
import android.os.Bundle;

...

Bundle extras = getIntent().getExtras();
if (extras != null)
{
    String myParam = extras.getString("paramName");
}
else
{
    //..oops!
}
```

In this example, the above code would be inside your MyActivity.java file.

Gotchas

This method can only pass strings. So let's say you need to pass an ArrayList to your ListActivity; a possible workaround is to pass a comma-separated-string and then split it on the other side.

Alternative Solutions

Use SharedPreferences

See Also

(BROKEN XREF TO RECIPE -1 'http://mylifewithandroid.blogspot.com/2007/12/playing-with-intents.html '), (BROKEN XREF TO RECIPE -1 'http://developer.android.com/guide/appendix/faq/commontasks.html')

9.5 Retrieving data from a Sub-Activity back to your Main Activity

Ulysses Levy

Problem

Your main activity needs to retrieve data from a sub activity.

Solution

Use `startActivityForResult()`, and `onActivityResult()` in the main activity, and `setResult()` in the sub-activity.

Discussion

In this example we return a string from a Sub-Activity (`MySubActivity`) back to the Main Activity (`MyMainActivity`).

The first step is to "push" data from `MyMainActivity` via the Intent mechanism.

Example 9-18.

```
public class MyMainActivity extends Activity
{
    //..for logging..
    private static final String TAG = "MainActivity";

    //..The request code is supposed to be unique?..
    public static final int MY_REQUEST_CODE = 123;

    @Override
    public void onCreate( Bundle savedInstanceState )
    {
        ...
    }

    private void pushFxn()
    {
        Intent intent =
```

```

        new Intent(
            this,
            MySubActivity.class );

        startActivityForResult( intent, MY_REQUEST_CODE );
    }

    protected void onActivityResult(
        int requestCode,
        int resultCode,
        Intent pData)
    {
        if ( requestCode == MY_REQUEST_CODE )
        {
            if (resultCode == Activity.RESULT_OK )
            {
                final String zData = pData.getExtras().getString( MySubActivity.EXTRA_STRING_NAME );

                //..do something with our retrieved value..

                Log.v( TAG, "Retrieved Value zData is "+zData );
                //..logcats "Retrieved Value zData is returnValueAsString"
            }
        }
    }
}

```

Notes:

- The Main Activity's `onActivityResult()` gets called after `MySubActivity.finish()`.
- The retrieved value is technically an `Intent`, and so we could use it for more complex data (such as a **uri** to a google contact or something). However, in the above example code, we are only interested in a string value via `Intent.getExtras()`.
- The `requestCode` (**MY_REQUEST_CODE**) is supposed to be unique, and might be useful later. ie. `Activity.finishActivity(MY_REQUEST_CODE)`

The second major step is to "pull" data back from `MySubActivity` to `MyMainActivity`.

Example 9-19.

```

public class MySubActivity extends Activity
{
    public static final String EXTRA_STRING_NAME = "extraStringName";

    @Override
    public void onCreate( Bundle savedInstanceState )
    {
        ...
    }

    private void pullFxn()

```

```

{
    Intent iData = new Intent();
    iData.putExtra(
        EXTRA_STRING_NAME,
        "returnValueAsString" );

    setResult(
        android.app.Activity.RESULT_OK,
        iData );

    //..returns us to the parent "MyMainActivity"..
    finish();
}
}

```

Code Notes:

- Once again, Intents are used as data (ie. **"iData"**).
- `setResult()` requires a result code such as **RESULT_OK**.
- `finish()` essentially pushes the result from `setResult()`.

Other

- Technically, the data from `MySubActivity` doesn't get "pull"-ed until we're back on the other side with `MyMainActivity`. So arguably it is more similar to a 2nd "push".
- We don't have to use a public static final String variable for our "extra" field name, but I thought it was good style.

Use Case (informal)

In my app, I have a `ListActivity` with a `ContextMenu` (user long presses a selection to do something), and I wanted to let the Main-Activity know which row the user had selected for the `ContextMenu` action (atm, my app only has one action). I ended up using intent extras to pass the selected row's index as a string back to the parent activity; From there I could just convert the index back to an int and use it to identify the user row selection via `ArrayList.get(index)`. This worked for me, however I am sure there is another/better way.

See Also

Also see [Recipe 9.4](#)

[resultCode "gotcha"](#)

[startActivityForResultExample](#) (under "Returning a Result from a Screen")

[int\) Activity.startActivityForResult\(\)](#)

9.6 Getting total and free space on the SD card

Amir Alagic

Problem

You want to find out the amount of total and available space on the SD card

Solution

Use StatFs and Environment classes from the android.os package to find total and available space on the SD card.

Discussion

Here is code that obtains the information:

Example 9-20.

```
StatFs statFs = new StatFs(Environment.getExternalStorageDirectory().getPath());
double bytesTotal = (long) statFs.getBlockSize() * (long) statFs.getBlockCount();
double megTotal = bytesTotal / 1048576;
```

To get total space on the SD card use StatFs in the android.os package and as constructor parameter use Environment.getExternalStorageDirectory().getPath().

Then multiply block size with number of blocks on the SD card.

Example 9-21.

```
(long) statFs.getBlockSize() * (long) statFs.getBlockCount();
```

And to get size in megabytes then divide that with 1048576. To get free space on the SD card replace statFs.getBlockCount() with statFs.getAvailableBlocks().

Example 9-22.

```
(long) statFs.getBlockSize() * (long) statFs.getAvailableBlocks();
```

If you want to display the value with two decimal places you can use a DecimalFormat object from java.text.

Example 9-23.

```
DecimalFormat twoDecimalForm = new DecimalFormat("#.##");
```

9.7 Creating a SQLite database in an Android application.

Rachee Singh

Problem

Data persistence in the application.

Solution

Using SQLite to store the application data. This would involve inherit SQLiteOpenHelper class.

Discussion

In order to use SQLite databases in an Android application, it is necessary to inherit SQLiteOpenHelper class. This is an inbuilt class that helps load a database file. It checks for the existence of the database file and if it exists, it loads it otherwise it creates one.

Example 9-24.

```
public class SqlOpenHelper extends SQLiteOpenHelper {
```

The constructor for the SQLiteOpenHelper class takes in a few arguments: context, database name, CursorFactory object, version number.

Example 9-25.

```
    public static final String DBNAME = "tasksdb.sqlite";
    public static final int VERSION = 1;
    public static final String TABLE_NAME = "tasks";
    public static final String ID = "id";
    public static final String NAME = "name";

    public SqlOpenHelper(Context context) {
        super(context, DBNAME, null, VERSION);
    }
```

Creating a database in SQL uses the "create" statement:

Example 9-26.

```
CREATE TABLE <table-name> (column1 INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, column2 TEXT);
```

Example 9-27.

```
    public void onCreate(SQLiteDatabase db) {
        createDatabase(db);
    }

    private void createDatabase(SQLiteDatabase db) {
        db.execSQL("create table " + TABLE_NAME + "(" +
            ID + " integer primary key autoincrement not null, " +
            NAME + " text " +
            + ");");
    }
}
```

To get a handle on the SQL database you created, instantiate the class inheriting SQLiteOpenHelper:

Example 9-28.

```
SqlOpenHelper helper = new SqlOpenHelper(this);
SQLiteDatabase database= helper.getWritableDatabase();
```

Now, the SQLiteDatabase database can be used to load elements stored in the database, update and insert elements to it.

9.8 Retrieving data from a SQLite database.

Rachee Singh

Problem

Loading items from an existing SQLite database.

Solution

Using a Cursor object to iterate over the database and storing them.

Discussion

In order to iterate over items in a database, we require an object of the Cursor class. To query the database, we use the query method along with appropriate arguments, most importantly: the table name, the column names for which we are extracting values.

Example 9-29.

```
ArrayList<Food> foods = new ArrayList(this);
Cursor listCursor = database.query(TABLE_NAME, new String [] {ID, NAME}, null, null, null, null, String.form
listCursor.moveToFirst();
Food t;
if(! listCursor.isAfterLast()) {
    do {
        Long id = listCursor.getLong(0);
        String name= listCursor.getString(1);
        t = new Food(name);
        foods.add(t);
    } while (listCursor.moveToNext());
}
listCursor.close();
```

moveToFirst() method starts from the first item in the database and moveToNext moves the cursor to the next item. We keep checking until we have reached the end of the database. Each item of the database is added to an ArrayList.

9.9 Inserting values into a SQLite database.

Rachee Singh

Problem

Adding values into a SQLite database.

Solution

Using insert() method and passing an object of ContentValues class.

Discussion

ContentValues provides something like a key-value pair, so NAME would be the key and 'Mangoes' would be the value. This would insert a row in the database with value 'Mangoes' in it.

Example 9-30.

```
ContentValues values = new ContentValues();
values.put(NAME, "Mangoes");
```

After creating the values we insert it into the table using the insert method. SQLite returns the ID for that row in the database.

Example 9-31.

```
Long id = (database.insert(TABLE_NAME, null, values));
tasks.add(t);
```

'id' is the ID for the row that we inserted into the database.

9.10 Work With Dates in SQLite

Jonathan Fuerth

Problem

Android's embedded SQLite3 database supports date and time data directly, including some useful date and time arithmetic. However, getting these dates out of the database is troublesome: there is no Cursor.getDate() in the Android API.

Solution

Use SQLite's strftime() function to convert between SQLite timestamp format and the Java API's "milliseconds since the epoch" representation.

Discussion

This recipe demonstrates the advantages of using SQLite timestamps over storing raw milliseconds values in your database, and shows how to retrieve those timestamps from your database as `java.util.Date` objects.

Background

The usual representation for an absolute timestamp in Unix is `time_t`, which historically was just an alias for a 32-bit integer. This integer represented the date as the number of seconds elapsed since UTC 00:00 on January 1, 1970 (the Unix time *epoch*.) On systems where `time_t` is still a 32-bit integer, the clock will roll over partway through the year 2038.

Java adopted a similar convention, but with a few twists. The epoch remains the same, but the count is always stored in a 64-bit signed integer (the native Java `long` type) and the units are milliseconds rather than seconds. This method of timekeeping will not roll over for another 292 million years.

Android example code that deals with persisting dates and times tends to simply store and retrieve the raw *milliseconds since the epoch* values in the database. However, by doing this, it misses out on some useful features built in to SQLite.

The Advantages

There are several advantages to storing proper SQLite timestamps in your data: you can default timestamp columns to the current time using no Java code at all; you can perform calendar-sensitive arithmetic such as selecting the first day of a week or month, or adding a week to the value stored in the database; and you can extract just the date or time components and return those from your data provider.

All of these code-saving advantages come with two added bonuses: first, your data provider's API can stick to the Android convention of passing timestamps around as `long` values; second, all of this date manipulation is done in the natively-compiled SQLite code, so the manipulations don't incur the garbage collection overhead of creating multiple `java.util.Date` or `java.util.Calendar` objects.

The Code

Without further ado, here's how to do it.

First, create a table that defines a column of type `timestamp`.

```
CREATE TABLE current_list ( item_id INTEGER NOT NULL, added_on TIME-  
STAMP NOT NULL DEFAULT current_timestamp, added_by VARCHAR(50) NOT  
NULL, quantity INTEGER NOT NULL, units VARCHAR(50) NOT NULL, CON-  
STRAINT current_list_pk PRIMARY KEY (item_id) );
```

Note the default value for the `added_on` column. Whenever you insert a row into this table, SQLite will automatically fill in the current time (accurate to the second) for the new record.

```
sqlite> insert into current_list (item_id, added_by, quantity, units) ...> values (1, 'fuertth', 1, 'EA'); sqlite> select * from current_list where item_id = 1; 1|2010-05-14 23:10:26|fuertth|1|EA sqlite>
```

See how the current date was inserted automatically? This is one of the advantages you get from working with SQLite timestamps.

How about the other advantages?

Select just the date part, forcing the time back to midnight:

```
sqlite> select item_id, date(added_on,'start of day') ...> from current_list where item_id = 1; 1|2010-05-14 sqlite>
```

Or adjust the date to the Monday of the following week:

```
sqlite> select item_id, date(added_on,'weekday 1') ...> from current_list where item_id = 1; 1|2010-05-17 sqlite>
```

Or the Monday before:

```
sqlite> select item_id, date(added_on,'weekday 1',' -7 days') ...> from current_list where item_id = 1; 1|2010-05-10 sqlite>
```

These examples are just the tip of the iceberg. You can do a lot of useful things with your timestamps once SQLite recognizes them as such.

Last, but not least, you must be wondering how to get these dates back into your Java code. The trick is to press another of SQLite's date functions into service - this time `strftime()`. Here is a Java method that fetches a row from the `current_list` table we've been working with:

Example 9-32.

```
Cursor cursor = database.rawQuery(
    "SELECT item_id AS id," +
    " (strftime('%s', added_on) * 1000) AS added_on," +
    " added_by, quantity, units" +
    " FROM current_list", new String[0]);
long millis = cursor.getLong(cursor.getColumnIndexOrThrow("added_on"));
Date addedOn = new Date(millis);
```

That's it: using `strftime`'s `%s` format, you can select timestamps directly into your Cursor as Java *milliseconds since the epoch* values. Client code will be none the wiser, except that your content provider will be able to do date manipulations for free that would take significant amounts of Java code and extra object allocations.

See Also

[SQLite's documentation for its date and time functions](#)

9.11 Parsing JSON using the Jackson Parser

Wagied Davids

Problem

Parse JSON using Jackson parser

Solution

Jackson (<http://jackson.codehaus.org/>) is a really fast streaming JSON parser and generator. It also offers full node-based Tree Model, as well as full OJM (Object/Json Mapper) data binding functionality.

Required Downloads available from <http://jackson.codehaus.org/Download> jackson-core-asl-1.8.0.jar jackson-jaxrs-1.8.0.jar jackson-mapper-asl-1.8.0.jar jackson-mrbean-1.8.0.jar jackson-xc-1.8.0.jar

Discussion

File: recipes-all.json

Example 9-33.

```
{ "recipes":
  [
    {
      "name": "Recipe 1",
      "id": "8aecfd9b2fa26e83012fa298c2a50017",
      "recipe": "1 Lorem ipsum...",
      "image": "/malibu-server/recipe/getImage/8aecfd9b2fa26e83012fa298c2a50017"
    },
    {
      "name": "Recipe 2",
      "id": "8aecfd9b2fa26e83012fa298c2a90018",
      "recipe": "2 Lorem ipsum...",
      "image": "/malibu-server/recipe/getImage/8aecfd9b2fa26e83012fa298c2a90018"
    },
    {
      "name": "Recipe 3",
      "id": "8aecfd9b2fa26e83012fa298c2ae0019",
      "recipe": "3 Lorem ipsum...",
      "image": "/malibu-server/recipe/getImage/8aecfd9b2fa26e83012fa298c2ae0019"
    }
  ]
}
```

File: main.xml

Example 9-34.

```
<?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:id="@+id/textview" android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Data Model files: File: A: Recipe.java

Example 9-35.

```
public class Recipe
{
    public String name;
    public String id;
    public String recipe;
    public String image;
}
```

File B: Recipes.java

Example 9-36.

```
import java.util.ArrayList;
import java.util.HashMap;

public class Recipes extends HashMap<String, ArrayList<Recipe>>
{
    //empty
}
```

File: Main.java

Example 9-37.

```
public class Main extends Activity
{
    private static final String tag = Main.class.getName();

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(com.android.jackson.json.R.layout.main);
        TextView textview = (TextView) this.findViewById(com.android.jackson.json.R.id.textview);

        StringBuffer strBuffer = new StringBuffer();

        MockRecipesController mockRecipesController = new MockRecipesController();
        mockRecipesController.init();
        for (Recipe recipe : mockRecipesController.findAll())
```



```

    {
        Log.d(tag, "Name: " + recipe.name);
        Log.d(tag, "ID: " + recipe.id);
        Log.d(tag, "Recipe: " + recipe.recipe);
        Log.d(tag, "Image: " + recipe.image);

        strBuffer.append("Name: " + recipe.name + "\n");
        strBuffer.append("ID: " + recipe.id + "\n");
        strBuffer.append("Recipe: " + recipe.recipe + "\n");
        strBuffer.append("Image: " + recipe.image + "\n");
    }

    // Finally
    textView.setText(strBuffer.toString());
}
}
}

```

File: MockRecipesController.java

Example 9-38.

```

package com.malibu.mock;

import java.io.IOException;
import java.util.ArrayList;

import org.codehaus.jackson.JsonFactory;
import org.codehaus.jackson.JsonParseException;
import org.codehaus.jackson.JsonParser;
import org.codehaus.jackson.map.ObjectMapper;

import com.malibu.models.Recipe;
import com.malibu.models.Recipes;

public class MockRecipesController
{
    private final String json = "{ \"recipes\": [\n" + " [ \n" + " { \n" + "     \"name\": \"Recipe 1\", \n"

    private ObjectMapper objectMapper = null;
    private JsonFactory jsonFactory = null;
    private JsonParser jp = null;
    private ArrayList<Recipe> recipes = null;
    private Recipes mRecipes = null;

    public MockRecipesController()
    {
        objectMapper = new ObjectMapper();
        jsonFactory = new JsonFactory();
    }

    public void init()
    {
        try
        {

```

```

        jp = jsonFactory.createJsonParser(json);
        mRecipes = objectMapper.readValue(jp, Recipes.class);
        recipes = mRecipes.get("recipes");
    }
    catch (JsonParseException e)
    {
        e.printStackTrace();
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
}

public ArrayList<Recipe> findAll()
{
    return recipes;
}

public Recipe findById(int id)
{
    return recipes.get(id);
}
}

```

9.12 Parsing an XML document using the DOM API

Ian Darwin

Problem

You have data in XML, and you want to transform it into something useful in your application.

Solution

Android provides a fairly good clone of the standard DOM API used in the Java Standard Edition. Using the DOM API instead of writing your own parsing code just makes sense.

Discussion

This is the code that parses the XML document containing the list of Recipes in the Android Cookbook, as discussed in [Recipe 11.2](#). The input file has a single *recipes* root element, followed by a sequence of *recipe* elements, each with an *id* and a *title* with textual content.

The code creates a DOMDocumentBuilderFactory, which can be tailored, for example, to make Schema-aware parsers. In real code this could be created in a static initializer instead of recreating it each time. The DocumentBuilderFactory is used to create a

Document Builder, a.k.a. Parser. The Parser expects to be reading from an `InputStream`, so we convert the data which we have in `String` form into an array of bytes and construct a `ByteArrayInputStream`. Again in real life, you would probably want to combine this code with the Web Service consumer so you could simply get the input stream from the network connection and read the XML directly into the Parser, instead of saving it as a `String` and then wrapping that in a converter as we do here.

Once the elements are parsed, we convert the Document into an array of Data (the singular of Data is Datum, so the class is called Datum) by calling the DOM API methods such as `getDocumentElement()`, `getChildNodes()`, and `getNodeValue()`. Since the DOM API was not invented by Java people, it doesn't use the standard Collections API but has its own collections like `NodeList`. In DOM's defense, the same or similar APIs are used in a really wide variety of programming language, so it can be said to be as much a standard as Java's Collections.

Here's the code:

Example 9-39.

```
/** Convert the list of Recipes in the String result from the
 * web service into an ArrayList of Datum.
 * @throws ParserConfigurationException
 * @throws IOException
 * @throws SAXException
 */
public static ArrayList<Datum> parse(String input) throws Exception {

    final ArrayList<Datum> results = new ArrayList<Datum>(1000);
    final DocumentBuilderFactory dbFactory =
        DocumentBuilderFactory.newInstance();
    final DocumentBuilder parser = dbFactory.newDocumentBuilder();

    final Document document =
        parser.parse(new ByteArrayInputStream(input.getBytes()));

    Element root = document.getDocumentElement();
    NodeList recipesList = root.getChildNodes();
    for (int i = 0; i < recipesList.getLength(); i++) {
        Node recipe = recipesList.item(i);
        NodeList fields = recipe.getChildNodes();
        String id = ((Element) fields.item(0)).getNodeValue();
        String title =
            ((Element) fields.item(1)).getNodeValue();
        Datum d = new Datum(Integer.parseInt(id), title);
        results.add(d);
    }
    return results;
}
```

In changing this code from Java SE to Android, the only change we had to make was to use `getNodeValue()` in the retrieval of `id` and `title` instead of Java SE's `getTextContent()`; So the API really is very close.

See Also

The Web Service is in [Recipe 11.2](#). There is much more on XML in the *Java Cookbook* chapter on XML.

9.13 Parsing an XML document using an XmlPullParser

Johan Pelgrim

Problem

You have data in XML, and you want to transform it into something useful in your application.

Solution

Apart from processing XML using DOM or SAX the Android framework also provides an implementation of the XmlPullParser interface provided in the XML Pull v1 API.

Discussion

Introduction

XmlPull v1 API is a simple to use XML pull parsing API that was designed for simplicity and very good performance both in constrained environment such as defined by J2ME and on server side when used in J2EE application servers. XML pull parsing allows incremental (sometimes called streaming) parsing of XML where application is in control - the parsing can be interrupted at any given moment and resumed when application is ready to consume more input.

Parsing XML with the XmlPullParser

The code below parses the XML document containing the list of Recipes in the Android Cookbook, as discussed in [Recipe 11.2](#) and [Recipe 9.12](#). The input file has a single recipes root element, followed by a sequence of recipe elements, each with an id and a title with textual content.

First we get an instance of an XmlPullParserFactory by calling its static `newInstance()` method. Basically this scans the classpath for instances of XmlPullParserFactory and XmlPullParser. If it cannot find any instances this method throws an `XmlPullParserException`. We get an instance of an XmlPullParser by calling the `newPullParser()` factory method. We then pass the recipe list URL via the `setInput(InputStream inputStream, String inputEncoding)` method. The call to `setInput` resets the parser state and sets the event type to the initial value `START_DOCUMENT`. Also note that we don't need to first retrieve the URL's content with the `converse` method, like was done in the [Recipe 11.2](#) and [Recipe 9.12](#) recipes.

Parsing XML input with an `XmlPullParser` means we are processing parser *events*. Simple events can be of the following type: `START_DOCUMENT`, `END_DOCUMENT`, `START_TAG`, `END_TAG` and `TEXT`. (You might notice that these closely mimic the SAX callback event handler methods). Once we have passed our URL to the `setInput()` method we are ready for processing these events.

The first event is of type `START_DOCUMENT`. We process the input until we encounter the `END_DOCUMENT` tag. We advance to the next event by calling the `next()` method. (Note: You can even process more events by calling the `nextToken()` method, but that is out of scope here).

The code simply keeps on advancing to the next event until it encounters a `START_TAG`. In this case we retrieve the element's local name by calling the `getName()` method. When namespace processing is disabled, the raw name is returned. We store the tag name in a local variable `currentTag`, as a bread crumb. (Note: When a start element contains attributes you can extract them via the `getAttributeValue(String namespace, String name)` method, again out of scope here). Now we simply fall through the loop and advance to the next event.

Once we encounter a `TEXT` event we check whether the `currentTag` is "id" or "title". If this is the case we retrieve the text contents by calling the `getText()` method and assign it to the appropriate local variable. We keep on doing this until we encounter a recipe `END_TAG` event. In this case we simply create a new `Datum` object with the previously created `id` and `title` variables.

Example 9-40.

```
public static ArrayList<Datum> parse(String url) throws IOException, XmlPullParserException {
    final ArrayList<Datum> results = new ArrayList<Datum>(1000);

    XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
    factory.setNamespaceAware(true);
    XmlPullParser xpp = factory.newPullParser();

    URL input = new URL(url);
    xpp.setInput(input.openStream(), null);

    int eventType = xpp.getEventType();
    String currentTag = null;
    Integer id = null;
    String title = null;
    while (eventType != XmlPullParser.END_DOCUMENT) {
        if (eventType == XmlPullParser.START_TAG) {
            currentTag = xpp.getName();
        } else if (eventType == XmlPullParser.TEXT) {
            if ("id".equals(currentTag)) {
                id = Integer.valueOf(xpp.getText());
            }
            if ("title".equals(currentTag)) {
                title = xpp.getText();
            }
        }
    }
}
```

```

        } else if (eventType == XmlPullParser.END_TAG) {
            if ("recipe".equals(xpp.getName())) {
                results.add(new Datum(id, title));
            }
        }
        eventType = xpp.next();
    }
    return results;
}

```

Making it more strict

We can rewrite the parse method to make it a bit more strict. In this case we use the `require()` method to verify the expected XML structure. Once we are on the `id` or `title` `START_TAG` event we call `nextText()` to retrieve the elements text content and advance to the `END_TAG` event immediately after.

Example 9-41.

```

public static ArrayList<Datum> parse(String url) throws IOException, XmlPullParserException {
    final ArrayList<Datum> results = new ArrayList<Datum>(1000);

    XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
    factory.setNamespaceAware(true);
    XmlPullParser xpp = factory.newPullParser();

    URL input = new URL(url);
    xpp.setInput(input.openStream(), null);

    xpp.nextTag();
    xpp.require(XmlPullParser.START_TAG, null, "recipes");
    while (xpp.nextTag() == XmlPullParser.START_TAG) {
        xpp.require(XmlPullParser.START_TAG, null, "recipe");

        xpp.nextTag();
        xpp.require(XmlPullParser.START_TAG, null, "id");
        Integer id = Integer.valueOf(xpp.nextText());
        xpp.require(XmlPullParser.END_TAG, null, "id");

        xpp.nextTag();
        xpp.require(XmlPullParser.START_TAG, null, "title");
        String title = xpp.nextText();
        xpp.require(XmlPullParser.END_TAG, null, "title");

        xpp.nextTag();
        xpp.require(XmlPullParser.END_TAG, null, "recipe");

        results.add(new Datum(id, title));
    }
    xpp.require(XmlPullParser.END_TAG, null, "recipes");

    return results;
}

```

Both methods return the same results. The recipe's downloadable source code uses the retrieved list of Datum objects to fill a `ListActivity`. When you click on a list item you are redirected to the corresponding recipe's web page.

Processing static XML resources

You can easily process static XML resources with an `XmlPullParser`. Simply call the `getXml()` method via your context's `getResources()` method and you will receive an instance of `XmlResourceParser`. This basically is an implementation of `XmlPullParser` with an extra convenience method to close the input resource, so you can use the above described techniques to process your static XML resources as well!

Conclusion

The `XmlPullParser` is the parser of choice for many developers basically because of its simplicity. If you want speed you should pick SAX. DOM is about twice as slow as SAX. Parsing XML with the `XmlPullParser` is somewhere in the middle between SAX and DOM.

Note

Don't forget to add the `android.permission.INTERNET` Permission to your `AndroidManifest.xml` or you will not be able to access any web connections.

See Also

[Recipe 11.2](http://developer.android.com/reference/org/xmlpull/v1/XmlPullParser.html) [Recipe 9.12](http://developer.android.com/reference/org/xmlpull/v1/XmlPullParserFactory.html) [Recipe 2.17](http://developer.android.com/reference/android/content/res/XmlResourceParser.html) <http://developer.android.com/reference/org/xmlpull/v1/XmlPullParser.html> <http://developer.android.com/reference/org/xmlpull/v1/XmlPullParserFactory.html> <http://developer.android.com/reference/android/content/res/XmlResourceParser.html>

Source Download URL

The source code for this example may be downloaded from this URL: <https://github.com/downloads/jpelgrim/androidcookbook/RecipeList.zip>

9.14 Accessing data from a file shipped with the App rather than in the filesystem

Rachee Singh

Problem

You need to access data stored in a file in `/res/raw` directory.

Solution

Using the `getResources()` and `openRawResource()` method to read from the sample file.

Discussion

We wish to read information from a file in the android application. So we will require to put the relevant file in the `res/raw` directory (we need to create the directory since it is not present by default). Then using the `InputStreamReader` and `BufferedReader`, we will read the file. Then we extract the `String` from the `BufferedReader` using the `readLine` method. Eclipse asks to enclose the `readLine` function within a `try-catch` block since there is a possibility of it throwing an `IOException`.

The file included in `/res/raw` is named 'samplefile'.

Example 9-42.

```
InputStreamReader is = new InputStreamReader(this.getResources().openRawResource(R.raw.samplefile));
BufferedReader reader = new BufferedReader(is);
StringBuilder finalText = new StringBuilder();
String line;
try {
    while ((line = reader.readLine()) != null) {
        finalText.append(line);
    }
} catch (IOException e) {
    e.printStackTrace();
}
fileTextView = (TextView)findViewById(R.id.fileText);
fileTextView.setText(finalText.toString());
```

After reading the entire string, we set it to the `TextView` in the activity.

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LMWJjYjQwMjYtNDVlMi00Y2M5LTk1MmItMTc3OGNhNWZiNjNh&hl=en_US

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LNTNmNjjiZjctN2I4MC00NDRiLWIzN2YtN2Q3MmM0ZjEwODY3&hl=en_US

9.15 Adding a Contact

Ian Darwin

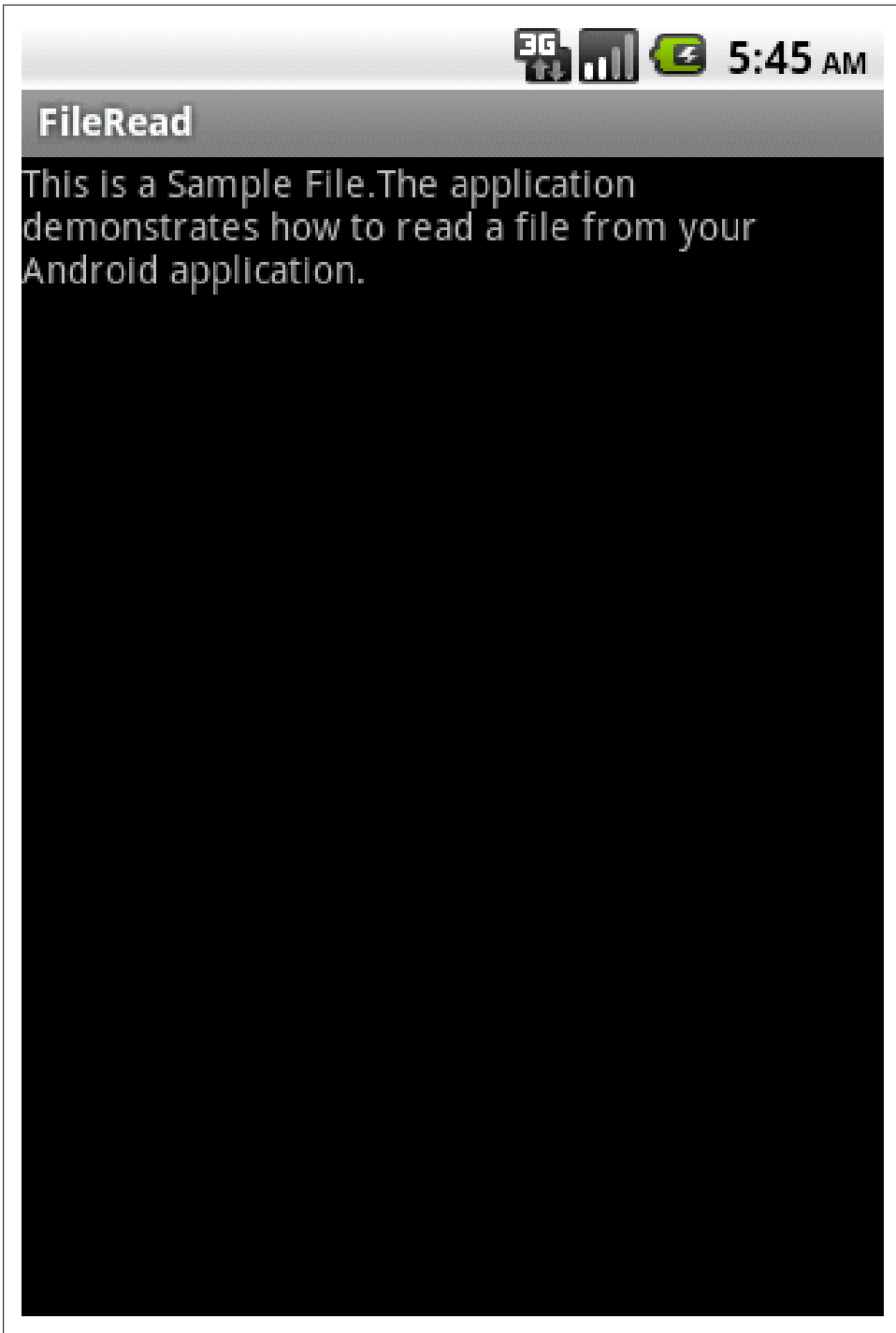


Figure 9-1.

Problem

You have some contact information that you want to save for use in the Contacts application.

Solution

Set up a list of operations for batch insert, and tell the persistence manager to run it.

Discussion

The Contacts database is, to be sure, "flexible". It has to adapt to many different kinds of accounts and contact management uses, with different types of data. And it is, as a result, somewhat complicated.

Note: In current versions, the classes named `Contact` (and by extension all its inner classes and interfaces) are deprecated, meaning "don't use in new development". The classes and interfaces that take their place have names beginning with (the somewhat cumbersome, and somewhat tongue-twisty) `ContactsContract`.

We'll start with the simplest case of adding a person's contact information. We want to insert this information - which we either got from the user or by finding it on the network someplace:

Name:	Jon Smith
Home Phone	416-555-5555
Work Phone	416-555-6666
Email	jon@jonsmith.domain

First we have to determine which `Account` to associate the data with (see (BROKEN XREF TO RECIPE -1 'Accounts')). For now we will use a fake `Account` name ("darwinian" is both an adjective, and my name, so we'll use that).

For each of the four fields, we'll need to create an `Account Operation`.

We add all five operations to a `List`, and pass that into `getContentResolver().applyBatch()`.

Here is the code for the `addContact()` method.

Example 9-43.

```
private void addContact() {
    final String ACCOUNT_NAME = "darwinian"
    String name = "Jon Smith";
    String homePhone = "416-555-5555";
    String workPhone = "416-555-6666";
    String email = "jon@jonsmith.domain";

    // Use new-style batch operations: Build List of ops then call applyBatch
```

```

try {
    ArrayList<ContentProviderOperation> ops = new ArrayList<ContentProviderOperation>();
    AuthenticatorDescription[] types = accountManager.getAuthenticatorTypes();
    ops.add(ContentProviderOperation.newInsert(
        ContactsContract.RawContacts.CONTENT_URI).withValue(
            ContactsContract.RawContacts.ACCOUNT_TYPE, types[0].type)
        .withValue(ContactsContract.RawContacts.ACCOUNT_NAME, ACCOUNT_NAME)
        .build());
    ops.add(ContentProviderOperation
        .newInsert(ContactsContract.Data.CONTENT_URI)
        .withValueBackReference(ContactsContract.Data.RAW_CONTACT_ID, 0)
        .withValue(ContactsContract.Data.MIMETYPE,
            ContactsContract.CommonDataKinds.StructuredName.CONTENT_ITEM_TYPE)
        .withValue(ContactsContract.CommonDataKinds.StructuredName.DISPLAY_NAME, name)
        .build());
    ops.add(ContentProviderOperation.newInsert(
        ContactsContract.Data.CONTENT_URI).withValueBackReference(
            ContactsContract.Data.RAW_CONTACT_ID, 0).withValue(
            ContactsContract.Data.MIMETYPE,
            ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE)
        .withValue(ContactsContract.CommonDataKinds.Phone.NUMBER,
            homePhone).withValue(
            ContactsContract.CommonDataKinds.Phone.TYPE,
            ContactsContract.CommonDataKinds.Phone.TYPE_HOME)
        .build());
    ops.add(ContentProviderOperation.newInsert(
        ContactsContract.Data.CONTENT_URI).withValueBackReference(
            ContactsContract.Data.RAW_CONTACT_ID, 0).withValue(
            ContactsContract.Data.MIMETYPE,
            ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE)
        .withValue(ContactsContract.CommonDataKinds.Phone.NUMBER,
            workPhone).withValue(
            ContactsContract.CommonDataKinds.Phone.TYPE,
            ContactsContract.CommonDataKinds.Phone.TYPE_WORK)
        .build());
    ops.add(ContentProviderOperation.newInsert(
        ContactsContract.Data.CONTENT_URI).withValueBackReference(
            ContactsContract.Data.RAW_CONTACT_ID, 0).withValue(
            ContactsContract.Data.MIMETYPE,
            ContactsContract.CommonDataKinds.Email.CONTENT_ITEM_TYPE)
        .withValue(ContactsContract.CommonDataKinds.Email.DATA, email)
        .withValue(ContactsContract.CommonDataKinds.Email.TYPE,
            ContactsContract.CommonDataKinds.Email.TYPE_HOME)
        .build());

    getContentResolver().applyBatch(ContactsContract.AUTHORITY, ops);

    Toast.makeText(this, getString(R.string.addContactSuccess),
        Toast.LENGTH_LONG).show();
} catch (Exception e) {

    Toast.makeText(this, getString(R.string.addContactFailure),
        Toast.LENGTH_LONG).show();
    Log.e(LOG_TAG, getString(R.string.addContactFailure), e);
}

```

```
} }  
}
```

The resulting contact shows up in the Contact manager (although, temporarily, in this version, you have to Search for it).

9.16 Reading Contact Data

Ian Darwin

Problem

You need to extract details, such as a phone number or email address, from the Contacts database.

Solution

Use an Intent to let the user pick one contact. Use a ContentResolver to create an SQLite Query for the chosen contact. Use SQLite and pre-defined constants in the dunningly-named ContactContract class to retrieve the parts you want. Be aware that the Contacts database was designed for generality, not for simplicity.

Discussion

This code is from TabbyText, my SMS Text Message sender for Tablets. The user has already picked the given contact (using the Contact app; see [Recipe 4.2](#)). In this code we want to extract the Mobile number and save it in a text field in the current Activity, so the user can post-edit it if need be, or even reject it, so we just set it on an EditText once we find it.

Finding it turns out to be the hard part. We start with a Query that we get from the Content Provider, to extract the ID field for the given contact. Information like phone numbers and emails are in their own tables, so we need a second query, to feed in the Id as part of the "select" part of the query. This query gives a list of the contact's phone numbers. We iterate through this, taking each valid phone number and setting it on the EditText.

A further elaboration would restrict this to only selecting the Mobile number (Contacts allows both Home Fax and Work Fax, but only one Mobile number, at least as of Honeycomb 3.2).

Example 9-44.

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    if (requestCode == REQ_GET_CONTACT) {  
        switch(resultCode) {  
            case Activity.RESULT_OK:
```

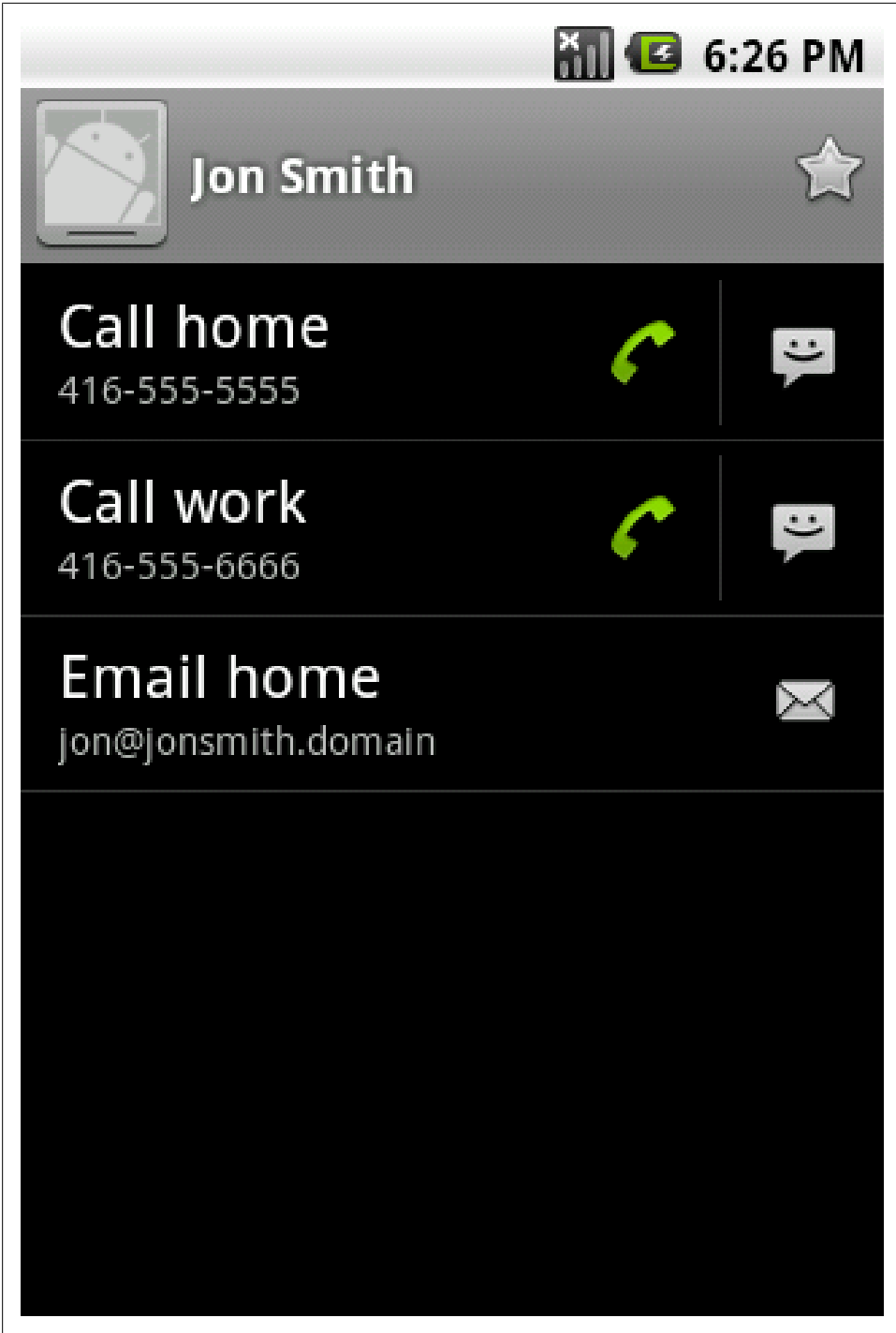


Figure 9-2.

```

// The Contacts API is about the most complex to use.
// First we have to retrieve the Contact, since we only get its URI from the Intent
Uri resultUri = data.getData(); // e.g., content://contacts/people/123
Cursor cont = getContentResolver().query(resultUri, null, null, null, null);
if (!cont.moveToNext()) { // expect 001 row(s)
    Toast.makeText(this, "Cursor contains no data", Toast.LENGTH_LONG).show();
    return;
}
int columnIndexForId = cont.getColumnIndex(ContactsContract.Contacts._ID);
String contactId = cont.getString(columnIndexForId);
int columnIndexForHasPhone = cont.getColumnIndex(ContactsContract.Contacts.HAS_PHONE_NUMBER);
boolean hasAnyPhone = Boolean.parseBoolean(cont.getString(columnIndexForHasPhone));
if (!hasAnyPhone) {
    Toast.makeText(this, "Selected contact seems to have no phone numbers ", Toast.LENGTH_LO
}

// Now we have to do another query to actually get the numbers!
Cursor numbers = getContentResolver().query(
    ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
    null,
    ContactsContract.CommonDataKinds.Phone.CONTACT_ID + "=" + contactId, // "selection",
    null, null);
// XXX still need to restrict to Mobile number!
while (numbers.moveToNext()) {
    String aNumber = numbers.getString(numbers.getColumnIndex(ContactsContract.CommonDataKin
    System.out.println(aNumber);
    number.setText(aNumber);
}
if (cont.moveToNext()) {
    System.out.println("WARNING: More than one contact returned from picker!");
}
numbers.close();
cont.close();
break;
case Activity.RESULT_CANCELED:
    // nothing to do here
    break;
default:
    Toast.makeText(this, "Unexpected resultCode: " + resultCode, Toast.LENGTH_LONG).show();
    break;
}
}
super.onActivityResult(requestCode, resultCode, data);
}

```

Source Download URL

The source code for this example may be downloaded from this URL: <http://projects.darwinsys.com/TabbyText-src.zip>

9.17 Parsing JSON using JSONObject

Rachee Singh

Problem

Many websites provide data in JSON. Many applications require to parse JSON and provide that data in the application.

Solution

Using inbuilt classes like JSONObject the process of parsing JSON is simplified in Android.

Discussion

We use a method to generate JSON code for sample purposes. In a real application you would obtain the JSON from some web source. In this method we make use of a JSONObject class object to put in values and then to return the corresponding String (using toString() method). Creating an object of type JSONObject throws a JSONException, so we enclose the code in a try-catch block.

Example 9-45.

```
private String getJsonString() {
    JSONObject string = new JSONObject();
    try {
        string.put("name", "John Doe");
        string.put("age", new Integer(25));
        string.put("address", "75 Ninth Avenue 2nd and 4th Floors New York, NY 10011");
        string.put("phone", "8367667829");
    } catch (JSONException e) {
        e.printStackTrace();
    }
    return string.toString();
}
```

We need to instantiate an object of class JSONObject that takes the JSON string as an argument. In this case, the JSON string is being obtained from the getJsonString method. From the jsonObject we extract the information and print it in a TextView.

Example 9-46.

```
try {
    String jsonString = getJsonString();
    JSONObject jsonObject = new JSONObject(jsonString);
    String name = jsonObject.getString("name");
    String age = jsonObject.getString("age");
    String address = jsonObject.getString("address");
    String phone = jsonObject.getString("phone");
    String jsonText = name + "\n" + age + "\n" + address + "\n" + phone;
    json = (TextView) findViewById(R.id.json);
    json.setText(jsonText);
} catch (JSONException e) {
    e.printStackTrace();
}
```

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LZDYxN2E3NTItMjE3Yy00YjE2LThjY2UtMGE2MTIyM2I0YjUx&hl=en_US

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LNTlhMWM3YTMtMjU4YS00MjhlLTg4YTYtNzljMzBiNjFlNjFh&hl=en_US

Telephone Applications

10.1 Introduction: Telephone Applications

Ian Darwin

Discussion

Android began as a platform for cellular telephone handsets, so it is no surprise that Android Apps are very capable of dealing with the phone. You can write apps that dial the phone, or that guide the user to do so. You can write apps that verify or modify the number the user is calling (for example, to add a long-distance dialing prefix). You can also send and receive SMS (short message service, a.k.a. Text Messages). Of course, both of these capabilities are discussed assuming your device is telephony equipped. Nowadays, a great many Android tablets are WiFi-only, and do not have "3G" or even 2G telephone/SMS capabilities. For these devices, other capabilities such as SMS via Internet and VOIP (Voice Over IP, usually SIP) have to be used.

This chapter covers most of these topics; a few will be found elsewhere in this book.

10.2 Do something when the phone rings

Johan Pelgrim

Problem

You want to act on an incoming phone call and do something with the incoming number.

Solution

This can be achieved by implementing a Broadcast receiver and listening for a `TelephonyManager.ACTION_PHONE_STATE_CHANGED` action.

Discussion

If you want to do something when the phone rings you have to implement a *broadcast receiver* which listens for the `TelephonyManager.ACTION_PHONE_STATE_CHANGED` intent action. This is a broadcast intent action indicating that the call state (cellular) on the device has changed.

1. Create a class `IncomingCallInterceptor` which extends `BroadcastReceiver`.
2. Override the `onReceive` method to handle incoming broadcast messages.
3. The `EXTRA_STATE` intent extra in this case indicates the new call state.
4. If (and only if) the new state is `RINGING`, a second intent extra `EXTRA_INCOMING_NUMBER` provides the incoming phone number as a `String`.
5. We extract the number information from the `EXTRA_INCOMING_NUMBER` intent extra.

Note: Additionally you can act on a state change to `OFFHOOK` or `IDLE` when the user picks up the phone or ends/rejects the phone call respectively.

Example 10-1.

```
package nl.codestone.cookbook.incomingcallinterceptor;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.telephony.TelephonyManager;
import android.widget.Toast;

public class IncomingCallInterceptor extends BroadcastReceiver { // 1

    @Override
    public void onReceive(Context context, Intent intent) { // 2
        String state = intent.getStringExtra(TelephonyManager.EXTRA_STATE); // 3
        String msg = "Phone state changed to " + state;

        if (TelephonyManager.EXTRA_STATE_RINGING.equals(state)) { // 4
            String incomingNumber = intent.getStringExtra(TelephonyManager.EXTRA_INCOMING_NUMBER); // 5
            msg += ". Incoming number is " + incomingNumber;

            // TODO This would be a good place to "Do something when the phone rings" ;-

        }

        Toast.makeText(context, msg, Toast.LENGTH_LONG).show();

    }
}
```

6. We have to register our `IncomingCallInterceptor` as a `<receiver>` within the `<application>` element in the `AndroidManifest.xml` file.

7. We register an `<intent-filter>` ...
8. and an `<action value` which registers our receiver to listen for `TelephonyManager.ACTION_PHONE_STATE_CHANGED` broadcast messages.
9. Finally we have to register a `<uses-permission>` so we are allowed to listen to phone state changes.

Example 10-2.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="nl.codestone.cookbook.incomingcallinterceptor"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-sdk android:minSdkVersion="3" />

    <application android:icon="@drawable/icon" android:label="Incoming Call Interceptor">

        <receiver android:name="IncomingCallInterceptor"                // 6
            <intent-filter>                                           // 7
                <action android:name="android.intent.action.PHONE_STATE"/> // 8
            </intent-filter>
        </receiver>

    </application>

    <uses-permission android:name="android.permission.READ_PHONE_STATE"/> // 9

</manifest>
```

If all is well you should see something like this when the phone rings:

What happens if two receivers listen for phone state changes?

In general, a *broadcast message* is just that, a message which is sent out to many receivers at the same time. This is the case for *normal broadcast* messages which is used to send out the `ACTION_PHONE_STATE_CHANGED` intent as well. All receivers of the broadcast are run in an undefined order, often at the same time and for that reason *order* is not applicable.

In other cases the system sends out *ordered broadcast* which is described in more detail in the Recipe [Recipe 10.3](#).

Final notes

When your *BroadcastReceiver* does not finish within 10 seconds the Android framework will show the infamous Application Not Responding (ANR) dialog, giving your users the possibility to kill your program. If you need to do some processing which takes longer than 10 seconds implement a *Service* and call the service method.



Figure 10-1.

It is also not advised to start an activity from a *BroadcastReceiver*, as it will spawn a new screen that will steal focus from whatever application the user is currently has running. If your application has something to show the user in response to an Intent broadcast, it should do so using the Notification Manager.

See Also

Recipe [Recipe 10.3](http://developer.android.com/reference/android/content/BroadcastReceiver.html) <http://developer.android.com/reference/android/content/BroadcastReceiver.html> http://developer.android.com/reference/android/telephony/TelephonyManager.html#ACTION_PHONE_STATE_CHANGED

Source Download URL

The source code for this example may be downloaded from this URL: <https://github.com/downloads/jpelgrim/androidcookbook/IncomingCallInterceptor.zip>

10.3 Process outgoing calls

Johan Pelgrim

Problem

You want to block certain calls, or alter the phone number about to be called.

Solution

Listen for the Intent.ACTION_NEW_OUTGOING_CALL broadcast action and set the result data of the broadcast receiver to the new number.

Discussion

If you want to intercept a call before it is about to be placed you can implement a broadcast receiver and listen for the Intent.ACTION_NEW_OUTGOING_CALL action. This recipe is in essence similar to the recipe [Recipe 10.2](#), but it is more interesting since we can actually manipulate the phone number in this case!

Here are the steps:

1. Create an `OutgoingCallInterceptor` class which extends the `BroadcastReceiver`
2. Override the `onReceive` method.
3. Extract the phone number which the user originally intended to call via the `Intent.EXTRA_PHONE_NUMBER` intent extra.
4. Replace this number by calling `setResultData` with the new number as the String argument.

Once the broadcast is finished, the result data is used as the actual number to call. If the result data is `null`, no call will be placed at all!

Example 10-3.

```
package nl.codestone.cookbook.outgoingcallinterceptor;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class OutgoingCallInterceptor extends BroadcastReceiver { // 1

    @Override
    public void onReceive(Context context, Intent intent) { // 2
        final String oldNumber = intent.getStringExtra(Intent.EXTRA_PHONE_NUMBER); // 3
        this.setResultData("0123456789"); // 4
        final String newNumber = this.getResultData();
        String msg = "Intercepted outgoing call. Old number " + oldNumber + ", new number " + newNumber;
        Toast.makeText(context, msg, Toast.LENGTH_LONG).show();
    }
}
```

5. We have to register our `OutgoingCallInterceptor` as a `<receiver>` within the `<application>` element in the `AndroidManifest.xml` file.
6. We add an `<intent-filter>` element within this `<receiver>` declaration and add a `android:priority` of 1.
7. We add an `<action>` element within the `<intent-filter>` to only receive `Intent.ACTION_NEW_OUTGOING_CALL` intent actions.
8. And we have to hold the `PROCESS_OUTGOING_CALLS` permission to receive this intent so we register a `<uses-permission>` to `PROCESS_OUTGOING_CALLS` right below the `<application>` element.

Example 10-4.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="nl.codestone.cookbook.outgoingcallinterceptor"
    android:versionCode="1" android:versionName="1.0">
    <uses-sdk android:minSdkVersion="3" />

    <application android:icon="@drawable/icon" android:label="Outgoing Call Interceptor">

        <receiver android:name="OutgoingCallInterceptor" // 5
            <intent-filter android:priority="1"> // 6
                <action android:name="android.intent.action.NEW_OUTGOING_CALL" /> // 7
            </intent-filter>
        </receiver>

    </application>

    <uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" /> // 8

</manifest>
```

Now, when you try to dial the number 11111 you will actually be forwarded to 0123456789 instead!

What happens if two receivers process outgoing calls?

As was stated before the `Intent.ACTION_NEW_OUTGOING_CALL` is an *ordered broadcast* and is a protected intent that can only be sent by the system. *Ordered broadcast* messages come with three additional features compared to *normal broadcast* messages.

1. You can use the `<intent-filter>` element's `android:priority` attribute to influence your position in the sending mechanism. The `android:priority` is an integer indicating which parent (receiver) has higher priority in processing the incoming broadcast message. The higher the number, the higher the priority and the sooner that receiver can process the broadcast message.
2. You can propagate a *result* to the next receiver by calling the `setResultData` method, and
3. You can completely abort the broadcast by calling the `abortBroadcast()` method so that it won't be passed to other receivers.

Note: According to the API any `BroadcastReceiver` receiving the `Intent.ACTION_NEW_OUTGOING_CALL` must *not abort* the broadcast by calling the `abortBroadcast()` method. Doing so does not present any errors, but apparently some system receivers still want to have a go at the broadcast message. Emergency calls **cannot** be intercepted using this mechanism, and other calls cannot be modified to call emergency numbers using this mechanism.

It is perfectly acceptable for multiple receivers to process the outgoing call in turn: for example, a parental control application might verify that the user is authorized to place the call at that time, then a number-rewriting application might add an area code if one was not specified.

In case two receivers are defined with an equal `android:priority` attribute they will be run in an arbitrary order (according to the API). However, in practice, when they both reside in the same `AndroidManifest.xml` file, it *looks* like the order in which the receivers are defined determines the order in which they will receive the broadcast message.

Furthermore, if two receivers are defined with an equal `android:priority` attribute but they are defined in a different `AndroidManifest.xml` file (i.e. they belong to a different application) it *looks* like the broadcast receiver, which was *installed* first is *registered* first and thus will be the one which is allowed to process the message first. But again, don't count on it!

If you want to have a shot at being the very first to process a message you can use the maximum integer value (2147483647). Even though the API this still does not guarantee you will be first, you will have a pretty good change though!

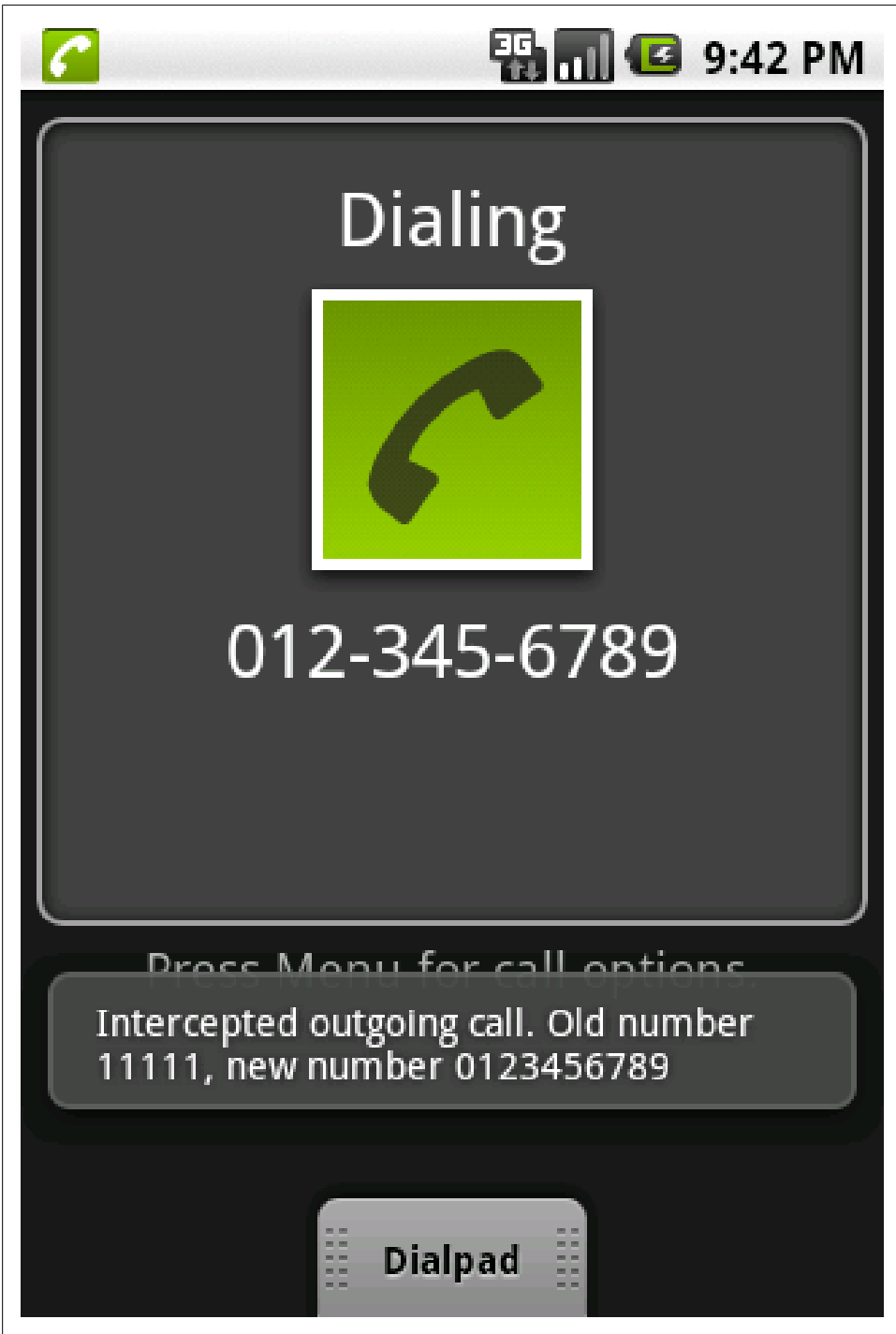


Figure 10-2.

Other applications could have intercepted the phone number before us. If you are pretty sure you want to take action on the original number you can use the `EXTRA_PHONE_NUMBER` intent extra as described above and completely ignore the result from the receiver before you. If you simply want to fall in line and pick up where another broadcast receiver has left off you can retrieve the intermediary phone number via the `getResultData()` method.

For consistency, any receiver whose purpose is to prohibit phone calls should have a priority of 0, to ensure it will see the final phone number to be dialed. Any receiver whose purpose is to rewrite phone numbers to be called should have a *positive* priority. Negative priorities are reserved for the system for this broadcast; using them may cause problems.

See Also

Recipe 10.2, http://developer.android.com/reference/android/content/Intent.html#ACTION_NEW_OUTGOING_CALL

Source Download URL

The source code for this example may be downloaded from this URL: <https://github.com/downloads/jpelgrim/androidcookbook/OutgoingCallInterceptor.zip>

10.4 Dialing the phone

Ian Darwin

Problem

You want to dial the phone from within an application, without worrying about details of telephony.

Solution

Start an Intent to dial the phone.

Discussion

One of the beauties of Android is the ease with which applications can re-use other applications, without being tightly coupled to the details (or even name) of the other program, using the Intent mechanism. For example, to call the phone, you only need to create and start an Intent with the action of `DIAL` and the "URI" of "tel" + the number you want to dial. Thus, a basic dialer can be as simple as this:

Example 10-5.

```
public class Main extends Activity {  
    String phoneNumber = "555-1212";
```

```

String intentStr = "tel:" + phoneNumber;

/** Standard creational callback.
 * Just dial the phone
 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    Intent intent = new Intent("android.intent.action.DIAL",
        Uri.parse(intentStr));

    startActivity(intent);
}
}

```

You need to have the permission `android.permission.CALL_PHONE` to use this code. The user will see this screen, and users know to press the green Phone button to let the call proceed.

Typically in real life you would not hard-code the number. In other circumstances you might want the user to call a number from the phone's Contacts list.

10.5 Sending single or multipart SMS messages

Colin Wilcox

Problem

A simple way to send either a single part or a multipart (handling the message concatenation UDH) from a single entry point

Solution

Use the `SMSManager`.

Discussion

Example 10-6.

```

public class SendSMS
{
    SMSManager iSMSManager = null;
    ArrayList<String> iFragmenList = null;

    SendSMS ()
    {
        iSMSManager = SMSManager.getDefault ();
    }
}

```

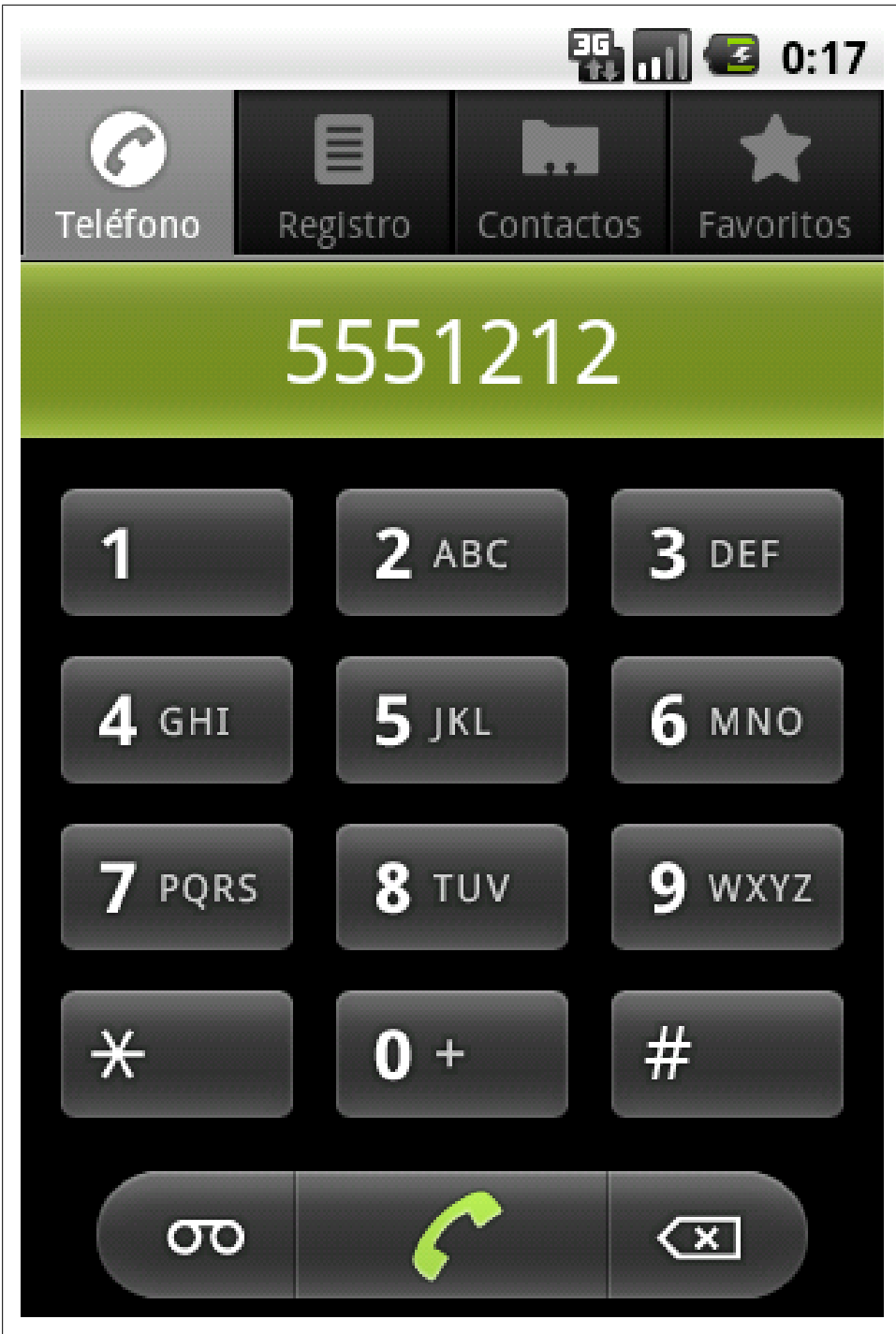


Figure 10-3.

```

public boolean sendSMSMessage (String aDestination, String aMessageText)
{
    if (iSMSManager == null)
        {
            return (false);
        }

        int fragmentCount = 0;

        iFragmentManager = iSMSManager.divideMessage (aMessageText);
        fragmentCount = iFragmentManager.Count ();
        if (fragmentCount > 1)
            {
                iSMSManager.SendMultipartMessage (aDestinationAddress, null, iFragmentManager, null, null, null);
            }
            else
            {
                iSMSManager.SendTextMessage (aDestinationAddress, null, aMessageText, null, null, null);
            }

            return true;
        }
}

// end of file

```

10.6 Receiving an SMS in an Android Application.

Rachee Singh

Problem

Enabling your application to receive an SMS.

Solution

Using a broadcast receiver to listen for incoming SMSs and then extracting the message from them.

Discussion

When an Android device receives a message, a broadcast intent is fired (the intent also includes the SMS that is sent). The application can register to receive these intents. The intent has an action `android.provider.Telephony.SMS_RECEIVED`. The application designed to receive SMSs should include the `RECEIVE_SMS` permission in the manifest:

Example 10-7.

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

When a message is received, the `onReceive` method (overridden) is called. Within this method, the message can be processed. From the intent that is received, the sms message has to be extracted using the `get()` method. The `BroadcastReceiver` with the extracting the message part looks something like this:

Example 10-8.

```
public class invitationSMSreciever extends BroadcastReceiver {

    public void onReceive(Context context, Intent intent) {

        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        String message = "";
        if(bundle != null) {
            Object[] pdus = (Object[]) bundle.get("pdus");
            msgs = new SmsMessage[pdus.length];

            for(int i=0; i<msgs.length;i++) {
                msgs[i] = SmsMessage.createFromPdu((byte[]) pdus[i]);
                message = msgs[i].getMessageBody();
                Toast.makeText(context,message,Toast.LENGTH_SHORT).show();
            }

        }

    }

}
```

The code makes a toast with the contents of the SMS sent.

To register the `invitationSMSreciever` class for receiving the SMSs, add the following code in the manifest:

Example 10-9.

```
<receiver android:name=".invitationSMSreciever"
    android:enabled="true">
    <intent-filter>
        <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</receiver>
```

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LMjk0YjJiZTgtZGI5ZC00Mjk3LTk2MGUtMjhkOGYzNmFmYWZ&hl=en_US&authkey=CMWZvskL

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LM2U0NjQ2OTMtMTdjMC00ZGFILWI1MzgtNzFhNjFiOWM0MGI4&hl=en_US&authkey=CLLju5EB

10.7 Using Emulator Controls to send SMS to the Emulator.

Rachee Singh

Problem

For testing SMS based applications sending SMS to the emulator.

Solution

Emulator Control in the DDMS perspective of Eclipse allows the functionality of sending SMSs to the emulator.

Discussion

To test if your application responds to incoming SMSs, we require to send SMS to the emulator. The DDMS perspective of Eclipse provides this function. In the Emulator Control tab, go to 'Telephony Actions' and provide a phone number. This number is any random number you would want to send an SMS. Select the SMS radio button. The message content that you wish to send can be typed in the message box.

10.8 Android TelephonyManager.

Pratik Rupwal

Problem

I want to obtain network related and telephonic information on my phone. How to do this?

Solution

Android TelephonyManager can be used to obtain the different statistics about network status and telephonic information.

Discussion

Android TelephonyManager provides information about the android telephony system. It assists in collecting different information like Cell Location, IMEI Number, Network Provider Information and more.

Below code covers many of the facilities provided by the android TelephonyManager:

Example 10-10.

```
import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.telephony.CellLocation;
import android.telephony.NeighboringCellInfo;
import android.telephony.PhoneStateListener;
import android.telephony.ServiceState;
import android.telephony.TelephonyManager;
import android.telephony.gsm.GsmCellLocation;
import android.util.Log;
import android.widget.ImageView;
import android.widget.ProgressBar;
import android.widget.TextView;

public class PhoneStateSample extends Activity {

    private static final String APP_NAME = "SignalLevelSample";
    private static final int EXCELLENT_LEVEL = 75;
    private static final int GOOD_LEVEL = 50;
    private static final int MODERATE_LEVEL = 25;
    private static final int WEAK_LEVEL = 0;

    private static final int INFO_SERVICE_STATE_INDEX = 0;
    private static final int INFO_CELL_LOCATION_INDEX = 1;
    private static final int INFO_CALL_STATE_INDEX = 2;
    private static final int INFO_CONNECTION_STATE_INDEX = 3;
    private static final int INFO_SIGNAL_LEVEL_INDEX = 4;
    private static final int INFO_SIGNAL_LEVEL_INFO_INDEX = 5;
    private static final int INFO_DATA_DIRECTION_INDEX = 6;
    private static final int INFO_DEVICE_INFO_INDEX = 7;

    private static final int[] info_ids= {
        R.id.serviceState_info,
        R.id.celllocation_info,
        R.id.callState_info,
        R.id.connectionState_info,
        R.id.signalLevel,
        R.id.signalLevelInfo,
        R.id.dataDirection,
        R.id.device_info
    };

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```



```

        setContentView(R.layout.main);
        startSignalLevelListener();
        displayTelephonyInfo();
    }

    @Override
    protected void onPause()
    {
        super.onPause();
        stopListening();
    }

    @Override
    protected void onResume()
    {
        super.onResume();
        startSignalLevelListener();
    }

    @Override
    protected void onDestroy()
    {
        stopListening();
        super.onDestroy();
    }

    private void setTextViewText(int id,String text) {
        ((TextView)findViewById(id)).setText(text);
    }
    private void setSignalLevel(int id,int infoId,int level){
        int progress = (int) (((float)level)/31.0) * 100;
        String signalLevelString =getSignalLevelString(progress);
        ((ProgressBar)findViewById(id)).setProgress(progress);
        ((TextView)findViewById(infoId)).setText(signalLevelString);
        Log.i("signalLevel ","" + progress);
    }

    private String getSignalLevelString(int level) {
        String signalLevelString = "Weak";
        if(level > EXCELLENT_LEVEL)    signalLevelString = "Excellent";
        else if(level > GOOD_LEVEL)    signalLevelString = "Good";
        else if(level > MODERATE_LEVEL) signalLevelString = "Moderate";
        else if(level > WEAK_LEVEL)    signalLevelString= "Weak";
        return signalLevelString;
    }

    private void stopListening(){
        TelephonyManager tm = (TelephonyManager)
getSystemService(TELEPHONY_SERVICE);
        tm.listen(phoneStateListener,
PhoneStateListener.LISTEN_NONE);
    }

    private void setDataDirection(int id, int direction){
        int resid = getDataDirectionRes(direction);
    }

```

```

        ((ImageView)findViewById(id)).setImageResource(resid);
    }
    private int getDataDirectionRes(int direction){
        int resid = R.drawable.data_none;

        switch(direction)
        {
            case TelephonyManager.DATA_ACTIVITY_IN:
                resid = R.drawable.data_in; break;
            case TelephonyManager.DATA_ACTIVITY_OUT:
                resid = R.drawable.data_out; break;
            case TelephonyManager.DATA_ACTIVITY_INOUT:
                resid = R.drawable.data_both; break;
            case TelephonyManager.DATA_ACTIVITY_NONE:
                resid = R.drawable.data_none; break;
            default: resid = R.drawable.data_none; break;
        }
        return resid;
    }
    private void startSignalLevelListener() {
        TelephonyManager tm = (TelephonyManager)
        getSystemService(TELEPHONY_SERVICE);
        int events = PhoneStateListener.LISTEN_SIGNAL_STRENGTH | PhoneStateListener.LISTEN_DATA_ACTIVITY |
        PhoneStateListener.LISTEN_CELL_LOCATION|PhoneStateListener.LISTEN_CALL_STATE |
        PhoneStateListener.LISTEN_CALL_FORWARDING_INDICATOR |
        PhoneStateListener.LISTEN_DATA_CONNECTION_STATE |
        PhoneStateListener.LISTEN_MESSAGE_WAITING_INDICATOR |
        PhoneStateListener.LISTEN_SERVICE_STATE;
        tm.listen(phoneStateListener, events);
    }
    private void displayTelephonyInfo(){
        TelephonyManager tm = (TelephonyManager)
        getSystemService(TELEPHONY_SERVICE);
        GsmCellLocation loc = (GsmCellLocation)tm.getCellLocation();
        int cellid = loc.getCellId();
        int lac = loc.getLac();
        String deviceid = tm.getDeviceId();
        String phonenumber = tm.getLine1Number();
        String softwareversion = tm.getDeviceSoftwareVersion();
        String operatorname = tm.getNetworkOperatorName();
        String simcountrycode = tm.getSimCountryIso();
        String simoperator = tm.getSimOperatorName();
        String simserialno = tm.getSimSerialNumber();
        String subscriberid = tm.getSubscriberId();
        String networktype = getNetworkTypeString(tm.getNetworkType());
        String phonetype = getPhoneTypeString(tm.getPhoneType());
        logString("CellID: " + cellid);
        logString("LAC: " + lac);
        logString("Device ID: " + deviceid);
        logString("Phone Number: " + phonenumber);
        logString("Software Version: " + softwareversion);
        logString("Operator Name: " + operatorname);
        logString("SIM Country Code: " + simcountrycode);
        logString("SIM Operator: " + simoperator);
        logString("SIM Serial No.: " + simserialno);
    }

```

```

logString("Subscriber ID: " + subscriberid);
String deviceinfo = "";
deviceinfo += ("CellID: " + cellid + "\n");
deviceinfo += ("LAC: " + lac + "\n");
deviceinfo += ("Device ID: " + deviceid + "\n");
deviceinfo += ("Phone Number: " + phonenumber + "\n");
deviceinfo += ("Software Version: " + softwareversion + "\n");
deviceinfo += ("Operator Name: " + operatorname + "\n");
deviceinfo += ("SIM Country Code: " + simcountrycode + "\n");
deviceinfo += ("SIM Operator: " + simoperator + "\n");
deviceinfo += ("SIM Serial No.: " + simserialno + "\n");
deviceinfo += ("Subscriber ID: " + subscriberid + "\n");
deviceinfo += ("Network Type: " + networktype + "\n");
deviceinfo += ("Phone Type: " + phonetype + "\n");
List<NeighboringCellInfo> cellinfo =tm.getNeighboringCellInfo();
if(null != cellinfo){
    for(NeighboringCellInfo info: cellinfo){
        deviceinfo += ("\tCellID: " + info.getCid() +", RSSI: " + info.getRssi() + "
    }
}
setTextViewText(info_ids[INFO_DEVICE_INFO_INDEX],deviceinfo);
}
private String getNetworkTypeString(int type){
    String typeString = "Unknown";
    switch(type)
    {
        case TelephonyManager.NETWORK_TYPE_EDGE:typeString = "EDGE"; break;
        case TelephonyManager.NETWORK_TYPE_GPRS:typeString = "GPRS"; break;
        case TelephonyManager.NETWORK_TYPE_UMTS:typeString = "UMTS"; break;
        default:
            typeString = "UNKNOWN"; break;
    }
    return typeString;
}
private String getPhoneTypeString(int type){
    String typeString = "Unknown";
    switch(type)
    {
        case TelephonyManager.PHONE_TYPE_GSM: typeString = GSM"; break;
        case TelephonyManager.PHONE_TYPE_NONE: typeString = UNKNOWN"; break;
        default:typeString = "UNKNOWN"; break;
    }
    return typeString;
}
private int logString(String message) {
    return Log.i(APP_NAME,message);
}

private final PhoneStateListener phoneStateListener = new PhoneStateListener(){

    @Override
    public void onCallForwardingIndicatorChanged(boolean cfi)
    {
        Log.i(APP_NAME, "onCallForwardingIndicatorChanged " +cfi);
        super.onCallForwardingIndicatorChanged(cfi);
    }
}

```

```

    }

    @Override
    public void onCallStateChanged(int state, String incomingNumber)
    {
        String callState = "UNKNOWN";
        switch(state)
        {
            case TelephonyManager.CALL_STATE_IDLE:callState = "IDLE"; break;
            case TelephonyManager.CALL_STATE_RINGING:callState = "Ringing (" + incomingNumber + ")"; break;
            case TelephonyManager.CALL_STATE_OFFHOOK: callState = "Offhook"; break;
        }
        setTextViewText(info_ids[INFO_CALL_STATE_INDEX],callState);
        Log.i(APP_NAME, "onCallStateChanged " + callState);
        super.onCallStateChanged(state, incomingNumber);
    }
    @Override
    public void onCellLocationChanged(CellLocation location)
    {
        String locationString = location.toString();
        setTextViewText(info_ids[INFO_CELL_LOCATION_INDEX],locationString);

        Log.i(APP_NAME, "onCellLocationChanged " +
locationString);
        super.onCellLocationChanged(location);
    }

    @Override
    public void onDataActivity(int direction)
    {
        String directionString = "none";
        switch(direction)
        {
            case TelephonyManager.DATA_ACTIVITY_IN:
                directionString = "IN"; break;
            case TelephonyManager.DATA_ACTIVITY_OUT:
                directionString = "OUT"; break;
            case TelephonyManager.DATA_ACTIVITY_INOUT:
                directionString = "INOUT";
                break;
            case TelephonyManager.DATA_ACTIVITY_NONE:
                directionString = "NONE"; break;
            default: directionString = "UNKNOWN: " +
direction; break;
        }

        setDataDirection(info_ids[INFO_DATA_DIRECTION_INDEX],direction);
        Log.i(APP_NAME, "onDataActivity " +
directionString);
        super.onDataActivity(direction);
    }

    @Override
    public void onDataConnectionStateChanged(int state)
    {

```

```

        String connectionState = "Unknown";
        switch(state)
        {
            case TelephonyManager.DATA_CONNECTED:
connectionState = "Connected"; break;
            case TelephonyManager.DATA_CONNECTING:
connectionState = "Connecting"; break;
            case TelephonyManager.DATA_DISCONNECTED:
connectionState = "Disconnected"; break;
            case TelephonyManager.DATA_SUSPENDED:
connectionState = "Suspended"; break;
            default:
                connectionState = "Unknown: " +
state; break;
        }

setTextViewText(info_ids[INFO_CONNECTION_STATE_INDEX],connectionState);

        Log.i(APP_NAME, "onDataConnectionStateChanged " +
connectionState);

        super.onDataConnectionStateChanged(state);
    }

    @Override
    public void onMessageWaitingIndicatorChanged(boolean mwi)
    {
        Log.i(APP_NAME, "onMessageWaitingIndicatorChanged
" + mwi);
        super.onMessageWaitingIndicatorChanged(mwi);
    }

    @Override
    public void onServiceStateChanged(ServiceState
serviceState)
    {
        String serviceStateString = "UNKNOWN";
        switch(serviceState.getState())
        {
            case ServiceState.STATE_IN_SERVICE:
serviceStateString = "IN SERVICE"; break;
            case ServiceState.STATE_EMERGENCY_ONLY:
serviceStateString = "EMERGENCY ONLY"; break;
            case ServiceState.STATE_OUT_OF_SERVICE:
serviceStateString = "OUT OF SERVICE"; break;
            case ServiceState.STATE_POWER_OFF:
serviceStateString = "POWER OFF"; break;
            default:
                serviceStateString = "UNKNOWN";
break;
        }
    }

```

```

setTextViewText(info_ids[INFO_SERVICE_STATE_INDEX],serviceStateString);

        Log.i(APP_NAME, "onServiceStateChanged " +
serviceStateString);

        super.onServiceStateChanged(serviceState);
    }

    @Override
    public void onSignalStrengthChanged(int asu)
    {
        Log.i(APP_NAME, "onSignalStrengthChanged " +
asu);
setSignalLevel(info_ids[INFO_SIGNAL_LEVEL_INDEX],info_ids[INFO_SIGNAL_LEV
EL_INFO_INDEX],asu);
        super.onSignalStrengthChanged(asu);
    }
};
}

```

Below code goes in 'main.xml' file:

Example 10-11.

```

<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:scrollbarStyle="insideOverlay"
    android:scrollbarAlwaysDrawVerticalTrack="false">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
            <TextView android:text="Service State"
style="@style/labelStyleRight"/>
            <TextView android:id="@+id/serviceState_info"
style="@style/textStyle"/>
        </LinearLayout>
        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
            <TextView android:text="Cell Location"
style="@style/labelStyleRight"/>
            <TextView android:id="@+id/cellLocation_info"
style="@style/textStyle"/>

```

```

        </LinearLayout>
        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
            <TextView android:text="Call State"
style="@style/labelStyleRight"/>
            <TextView android:id="@+id/callState_info"
style="@style/textStyle"/>
        </LinearLayout>
        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
            <TextView android:text="Connection State"
style="@style/labelStyleRight"/>
            <TextView android:id="@+id/connectionState_info"
style="@style/textStyle"/>
        </LinearLayout>
        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
            <TextView android:text="Signal Level"
style="@style/labelStyleRight"/>
            <LinearLayout
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_weight="0.5"
                android:orientation="horizontal">
                <ProgressBar
android:id="@+id/signalLevel" style="@style/progressStyle"/>
                <TextView
android:id="@+id/signalLevelInfo" style="@style/textSmallStyle"/>
            </LinearLayout>
        </LinearLayout>
        <LinearLayout
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
            <TextView android:text="Data"
style="@style/labelStyleRight"/>
            <ImageView android:id="@+id/dataDirection"
style="@style/imageStyle"/>
        </LinearLayout>
        <TextView android:id="@+id/device_info"
style="@style/labelStyleLeft"/>
    </LinearLayout>
</ScrollView>

```

Our code uses some UI styles which are declared in this file named 'styles.xml':

Example 10-12.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <style name="labelStyleRight">
    <item name="android:layout_width">fill_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_weight">0.5</item>
    <item name="android:textSize">15dip</item>
    <item name="android:textStyle">bold</item>
    <item name="android:layout_margin">10dip</item>
    <item name="android:gravity">center_vertical|right</item>
  </style>

  <style name="labelStyleLeft">
    <item name="android:layout_width">fill_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_weight">0.5</item>
    <item name="android:textSize">15dip</item>
    <item name="android:textStyle">bold</item>
    <item name="android:layout_margin">10dip</item>
    <item name="android:gravity">center_vertical|left</item>
  </style>

  <style name="textStyle">
    <item name="android:layout_width">fill_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_weight">0.5</item>
    <item name="android:textSize">15dip</item>
    <item name="android:textStyle">bold</item>
    <item name="android:layout_margin">10dip</item>
    <item name="android:gravity">center_vertical|left</item>
  </style>

  <style name="textSmallStyle">
    <item name="android:layout_width">fill_parent</item>
    <item name="android:layout_height">fill_parent</item>
    <item name="android:layout_weight">0.5</item>
    <item name="android:textSize">10dip</item>
    <item name="android:layout_margin">10dip</item>
    <item name="android:gravity">center_vertical|left</item>
  </style>

  <style name="progressStyle">
    <item name="android:layout_width">fill_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_margin">10dip</item>
    <item name="android:layout_weight">0.5</item>
    <item name="android:indeterminateOnly">false</item>
    <item name="android:minHeight">20dip</item>
    <item name="android:maxHeight">20dip</item>
    <item name="android:progress">15</item>
    <item name="android:max">100</item>
    <item name="android:gravity">center_vertical|left</item>
    <item name="android:progressDrawable">@android:drawable/progress_horizontal</item>
  </style>
</resources>
```



```

em>
    <item
name="android:indeterminateDrawable">@android:drawable/progress_indetermi
nate_horizontal</item>
    </style>

    <style name="imageStyle">
        <item name="android:layout_width">fill_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:layout_weight">0.5</item>
        <item name="android:src">@drawable/icon</item>
        <item name="android:scaleType">fitStart</item>
        <item name="android:layout_margin">10dip</item>
        <item name="android:gravity">center_vertical|left</item>
    </style>
</resources>

```

The application uses following android permission which needs to be added in 'AndroidManifest.xml' file of your project.

Example 10-13.

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

The application also uses some images for indicating the data communication state as no data communication, incoming data communication, outgoing data communication and both ways data communication. These images are respectively named as 'data_none.png', 'data_in.png', 'data_out.png' and 'data_both.png'. Please add some icons with above names in 'res/drawable' folder of your project structure.

Networked Applications

11.1 Introduction: Networking

Ian Darwin

Discussion

lorem ipsum dolor. This is intended to be representative of the "chapter introductions" that appear at the start of each chapter of an O'Reilly Cookbook and give general guidance and an introduction to the chapter.

Choose Your Protocol Wisely

While Java makes it easy to create network connections on any protocol, experience shows that HTTP (and HTTPS) are the most universal. If you use a custom protocol talking to your own server, there are some users who will not be able to access your server. Bear in mind that in some countries, high-speed data (aka 3G) is either not yet available or is very expensive, whereas GPRS/EDGE is less expensive and more widely available. Most GPRS service providers only allow HTTP/HTTPS connections, often through a WAP proxy. That said, there may be things you need to do that can't be done via HTTP, for example because their protocol demands a different port number (SIP over port 5000, for example). But do try to make HTTP your first choice when you can - you'll include more customers.

11.2 Using a RESTful Web Service

Ian Darwin

Problem

You need to access a RESTful Web Service.

Solution

You can either use the "standard" Java URL and URLConnection objects, or you can use the Android-provided Apache HttpClient library to code at a slightly higher level or to use HTTP methods other than GET and POST.

Discussion

REST was originally intended as an architectural description of the early Web, in which GET requests were used and in which the URL fully specified (Represented) the State of the request. Today RESTful web services are those which eschew the overhead of XML SOAP, WSDL and (usually) XML Schema, and simply send URLs that contain all the information needed to perform the request (or almost all, there is often a POST body sent for some types of requests). For example, to support an Android client that allows off-line editing of Recipes for this Android Cookbook, there is a (draft) Web Service which allows you to view the list of Recipes (you send an HTTP GET request ending in /recipe/list), to view the details of one Recipe (HTTP GET ending in /recipe/NNN where NNN is the primary key of the entry, gotten from the list request above), and later upload your revised version of the Recipe using HTTP POST to /recipe/NNN with the POST body containing the revised Recipe in the same XML document format as the "get Recipe" operation downloads it.

By the way, the RESTful Service used by these examples is implemented in server-side Java using the JAX-RS API, provided by JBoss Seam using RestEasy. The code for the service is available in the Cookbook CVS (subject to O'Reilly approval).

Using URL and URLConnection

Android's developers wisely preserved a lot of the Java standard API, including some widely-used classes for networking, so as to make it easy to port existing code. The converse() method shown below uses a URL and URLConnection from java.net to do a GET, and is extracted from an example in the networking chapter of my *Java Cookbook*. Comments in this version show what you'd need to change to do a POST.

Example 11-1.

```
public static String converse(String host, int port, String path) throws IOException {
    URL url = new URL("http", host, port, path);
    URLConnection conn = url.openConnection();
    // This does a GET; to do a POST, add conn.setDoInput(true);
    conn.setDoOutput(true);
    conn.setAllowUserInteraction(true);

    conn.connect();

    // To do a POST, you'd write to conn.getOutputStream();

    StringBuilder sb = new StringBuilder();
    BufferedReader in = new BufferedReader(
```

```

        new InputStreamReader(conn.getInputStream()));
String line;
while ((line = in.readLine()) != null) {
    sb.append(line);
}
in.close();
return sb.toString();
}

```

The invocation of this method in, say, your `onResume()` or `onCreate()` method, can be as simple as the following, which gets the list of Recipes from this Android Cookbook:

Example 11-2.

```

String host = "androidcookbook.net";
String path = "/seam/resource/rest/recipe/list";
String ret = converse(host, 80, path);

```

Using HTTPClient

Android supports the Apache HTTPClient library which is widely used for communicating at a slightly higher level than the `URLConnection`. I've used it in my [PageUnit web test framework](#). `HttpClient` also lets you use other HTTP methods that are common in RESTful services, such as PUT, DELETE, etc. (the `URLConnection` object used above, by contrast, only supports GET and POST(?)). Here's the same `converse` method coded for a GET using `HttpClient`:

Example 11-3.

```

public static String converse(String host, int port, String path, String postBody) throws IOException {
    HttpHost target = new HttpHost(host, port);
    HttpClient client = new DefaultHttpClient();
    HttpGet get = new HttpGet(path);
    HttpEntity results = null;
    try {
        HttpResponse response=client.execute(target, get);
        results = response.getEntity();
        return EntityUtils.toString(results);
    } catch (Exception e) {
        throw new RuntimeException("Web Service Failure");
    } finally {
        if (results!=null)
            try {
                results.consumeContent();
            } catch (IOException e) {
                // empty, Checked exception but don't care
            }
    }
}

```

Usage will be exactly the same as for the `URLConnection`-based version.

The Results

In the present version of the web service, the return value comes back as an XML document, which you'd need to parse to display in a List. If there is enough interest, we might add a JSON version as well.

Note

Don't forget to add the *android.permission.INTERNET* Permission to your *Android-Manifest.xml* or you will not be able to access any web connections.

See Also

(BROKEN XREF TO RECIPE -1 'Using a SOAP Web Service'), [Recipe 9.12](#), (BROKEN XREF TO RECIPE -1 'Displaying a List')

11.3 Extracting Information from Unstructured Text using Regular Expressions

Ian Darwin

Problem

You want to get information from another source, but they don't make it available as information, only as a viewable web page.

Solution

Use `java.net` to download the HTML page, and use Regular Expressions to extract the information from the page.

Discussion

If you aren't already a big fan of regular expressions, well, you should be. And maybe this recipe will help interest you in learning regex technology.

Suppose that I, as a published author, want to track how my book is selling in comparison to others. This information can be obtained for free just by clicking on the page for my book on any of the major bookseller sites, reading the sales rank number off the screen, and typing the number into a file—but that's too tedious. As I wrote in one of my earlier books, "computers get paid to extract relevant information from files; people should not have to do such mundane tasks." This program uses the Regular Expressions API and, in particular, newline matching to extract a value from an HTML page on the Amazon.com web site. It also reads from a URL object (see [Recipe 11.2](#)). The pattern to look for is something like this (bear in mind that the HTML may change at any time, so I want to keep the pattern fairly general):

Example 11-4.

```
(bookstore name here) Sales Rank:  
# 26,252
```

As the pattern may extend over more than one line, I read the entire web page from the URL into a single long string using a private convenience routine `readerToString()` instead of the more traditional line-at-a-time paradigm. The value is extracted from the regular expression, converted to an integer value, and returned. The longer version of this code in the Java Cookbook would also plot a graph using an external program. The complete program is shown in this example.

Example 11-5.

```
// Part of class BookRank  
public static int getBookRank(String isbn) throws IOException {  
    // The RE pattern - digits and commas allowed  
    final String pattern = "Rank:</b> #([\\d,]+)";  
    final Pattern r = Pattern.compile(pattern);  
  
    // The url -- must have the "isbn=" at the very end, or otherwise  
    // be amenable to being appended to.  
    final String url = "http://www.amazon.com/exec/obidos/ASIN/" + isbn;  
  
    // Open the URL and get a Reader from it.  
    final BufferedReader is = new BufferedReader(new InputStreamReader(  
        new URL(url).openStream()));  
    // Read the URL looking for the rank information, as  
    // a single long string, so can match RE across multi-lines.  
    final String input = readerToString(is);  
  
    // If found, append to sales data file.  
    Matcher m = r.matcher(input);  
    if (m.find()) {  
        // Group 1 is digits (and maybe ', 's) that matched; remove comma  
        return Integer.parseInt(m.group(1).replace(",", ""));  
    } else {  
        throw new RuntimeException(  
            "Pattern not matched in `" + url + "'!");  
    }  
}
```

See Also

As mentioned, using the regex API is vital to being able to deal with semi-structured data that you will meet in real life. Chapter Four of the *Java Cookbook* is all about regex, as is Jeffrey Friedl's comprehensive *Mastering Regular Expressions*.

Source Download URL

The source code for this example may be downloaded from this URL: <http://javacook.darwinsys.com/javasrc/regex/BookRank.java>

11.4 Parsing RSS/ATOM feeds parsing with ROME

Wagied Davids

Problem

You want to parse RSS/Atom feeds.

Solution

Based on ROME (<https://rome.dev.java.net/>) a Java-based RSS syndication feed parser. It has some nifty features such as HTTP conditional GETs, ETags and GZip compression. It also covers a wide range of formats from RSS 0.90, RSS 2.0, and Atom 0.3 & 1.0.

1. Modify your AndroidManifest.xml to allow for internet browsing. 2. Download the appropriate JAR files. rome-0.9.jar jdom-1.0.jar

File: AndroidManifest.xml

Example 11-6.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.pfizer.android"
  android:versionCode="1"
  android:versionName="1.0">
  <application
    android:icon="@drawable/icon"
    android:label="@string/app_name">
    <activity
      android:name=".AndroidRss"
      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-permission
      android:name="android.permission.INTERNET"></uses-permission>
</manifest>
```

File: main.xml

Example 11-7.

```
<?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">
```

```

<TableLayout
    android:id="@+id/table"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:stretchColumns="0">
    <TableRow
        android:id="@+id/top_add_entry_row"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent">

        <EditText
            android:id="@+id/rssURL"
            android:hint="Enter RSS URL"
            android:singleLine="true"
            android:maxLines="1"
            android:maxLength="220dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        </EditText>
        <Button
            android:id="@+id/goButton"
            android:text="Go"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content">
        </Button>
    </TableRow>
</TableLayout>

<!-- Mid Panel -->
<ListView
    android:id="@+id/ListView"
    android:layout_weight="1"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
</ListView>

<Button
    android:id="@+id/clearButton"
    android:text="Clear"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
</Button>
</LinearLayout>

```

File: AndroidRss.java

Example 11-8.

```

import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

```



```

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

import com.sun.syndication.feed.synd.SyndEntry;
import com.sun.syndication.feed.synd.SyndFeed;
import com.sun.syndication.io.FeedException;
import com.sun.syndication.io.SyndFeedInput;
import com.sun.syndication.io.XmlReader;

public class AndroidRss extends Activity
{
    private static final String tag="AndroidRss ";
    private int selectedItemIndex = 0;
    private final ArrayList list = new ArrayList();
    private EditText text;
    private ListView listView;
    private Button goButton;
    private Button clearButton;
    private ArrayAdapter adapter = null;

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

        text = (EditText) this.findViewById(R.id.rssURL);
        goButton = (Button) this.findViewById(R.id.goButton);
        goButton.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {
                String rss = text.getText().toString().trim();
                getRSS(rss);
            }
        });

        clearButton = (Button) this.findViewById(R.id.clearButton);
        clearButton.setOnClickListener(new OnClickListener()
        {
            @Override
            public void onClick(View v)
            {

```

```

        adapter.clear();
        adapter.notifyDataSetChanged();
    }
});

listView = (ListView) this.findViewById(R.id.ListView);
listView.setOnItemClickListener(new OnItemClickListener()
{
    @Override
    public void onItemClick(AdapterView parent, View view, int position, long duration)
    {
        selectedItemIndex = position;
        Toast.makeText(getApplicationContext(), "Selected " + adapter.getItem(position),
        Toast.LENGTH_SHORT).show();
    }
});

adapter = new ArrayAdapter(this, R.layout.dataview, R.id.ListItemView);
listView.setAdapter(adapter);
}

private void getRSS(String rss)
{
    URL feedUrl;
    try
    {
        Log.d("DEBUG", "Entered:" + rss);
        feedUrl = new URL(rss);

        SyndFeedInput input = new SyndFeedInput();
        SyndFeed feed = input.build(new XmlReader(feedUrl));
        List entries = feed.getEntries();
        Toast.makeText(this, "#Feeds retrieved: " + entries.size(), Toast.LENGTH_SHORT);

        Iterator iterator = entries.listIterator();
        while (iterator.hasNext())
        {
            SyndEntry ent = (SyndEntry) iterator.next();
            String title = ent.getTitle();
            adapter.add(title);
        }
        adapter.notifyDataSetChanged();
    }
    catch (MalformedURLException e)
    {
        e.printStackTrace();
    }
    catch (IllegalArgumentException e)
    {
        e.printStackTrace();
    }
    catch (FeedException e)
    {
        e.printStackTrace();
    }
}

```

```

        e.printStackTrace();
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
}

private void clearTextFields()
{
    Log.d(tag, "clearTextFields()");
    this.text.setText("");
}
}

```

Discussion

TODO

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.filefactory.com/file/b3f3435/n/AndroidRss.zip>

11.5 Using MD5 to Digest Free Text

Colin Wilcox

Problem

Sometime it is necessary to save or send free text in an encrypted form. Android provides a standard Java MD5 class to allow plaintext to be replaced with an MD5 digest of the original text. This is a one-way digest that is not believed to be easily reversible (if you need that, use Java Cryptography).

Solution

Below is a simple function that takes a free text string and digests it using MD5, returning the encrypted string as a return value

Discussion

Example 11-9.

```

public static String md5(String s)
{
    try
    {
        // Create MD5 Hash
        MessageDigest digest = java.security.MessageDigest.getInstance("MD5");
    }
}

```



Figure 11-1.

```

        digest.update(s.getBytes());
        byte messageDigest[] = digest.digest();
        // Create Hex String
        StringBuffer hexString = new StringBuffer();
        for (int i = 0; i < messageDigest.length; i++)
        {
            hexString.append(Integer.toHexString(0xFF & messageDigest[i]));
        }
        return hexString.toString();
    }
    catch (NoSuchAlgorithmException e)
    {
        e.printStackTrace();
    }
}
return "";
}

```

11.6 Converting text into hyperlinks

Rachee Singh

Problem

You need to make webpage URLs into hyper-links in a TextView of your Android app

Solution

Use the autolink property for a TextView

Discussion

You are setting the URL: 'www.google.com' as part of the text in a TextView but you want this text to be a hyper-link so that the user can open the web page in a browser by clicking on it. To achieve this, add a property to the TextView:

Example 11-10.

```
android:autoLink = "all"
```

Now, in the Activity's code you can set any text to the TextView and all the URLs will be converted to hyper-links!

Example 11-11.

```
linkText = (TextView)findViewById(R.id.link);
linkText.setText("The link is: www.google.com");
```

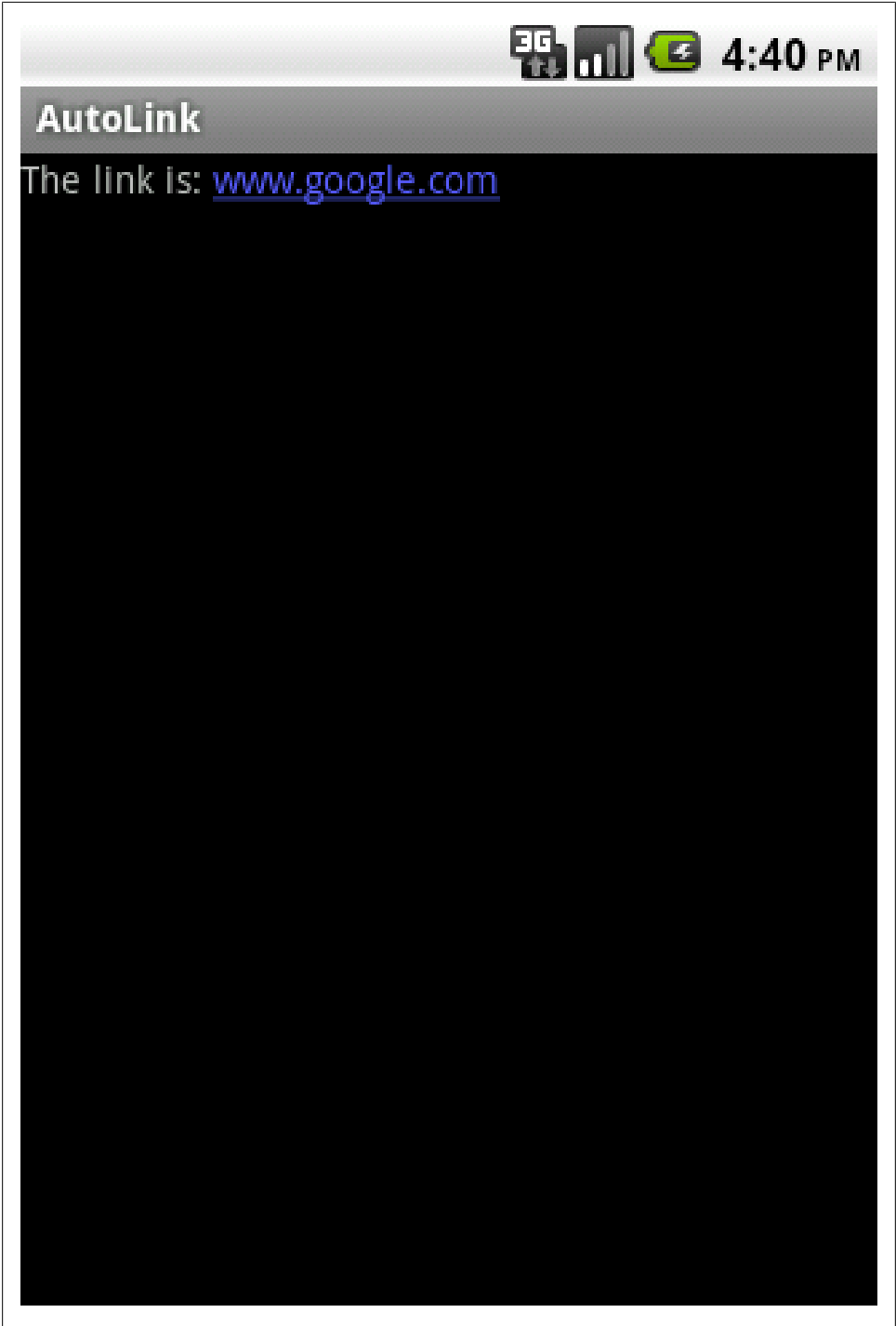


Figure 11-2.

11.7 Accessing a web page through your Android application

Rachee Singh

Problem

Opening a website within your application.

Solution

Embed a WebView component in the layout and use it to display the web page.

Discussion

In the application, since we wish to access a web page, we need to add the Internet permission into the manifest file:

Example 11-12.

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

The component which is used to display a web page within an application is called a WebView. We add a WebView to the XML layout:

Example 11-13.

```
<WebView  
android:id="@+id/webview"  
android:layout_height="fill_parent"  
android:layout_width="fill_parent"/>
```

In the Java code for the activity that displays the web page, we obtain a handle onto the WebView using findViewById() method. On the WebView we use the loadUrl method to provide it the URL of the web site we wish to open in the application.

Example 11-14.

```
WebView webview = (WebView)findViewById(R.id.webview);  
webview.loadUrl("http://google.com");
```

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LN2JhMDFjZTUtY2IwZS00NzkyLWFhNjItMzhiZWRIYTQxMWNm&hl=en_US

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LYWVjZTA1ZjAtY2EwNC00Y2Q1LWFhNmEtZmQxNWEwNjg4OWM3&hl=en_US

11.8 Customizing a WebView

Rachee Singh

Problem

You need to customize the WebView opened by your application.

Solution

Using the WebSettings class for accessing inbuilt functions for customizing the browser.

Discussion

As discussed in the recipe titled [Recipe 11.7](#), to open a web page in an Android application we use a WebView component. Then to load a URL in the WebView we use:

Example 11-15.

```
webView.loadUrl("http://www.google.com/");
```

Now, to customize the browser to suit the user's needs there are multiple actions that can be taken. We need to instantiate the WebSettings class:

Example 11-16.

```
WebSettings webSettings = webView.getSettings();
```

Here are a few things that can be done using WebSettings:

1. Tell the WebView to block network images:

Example 11-17.

```
webSettings.setBlockNetworkImage (true);
```

2. Set the default font size in the browser:

Example 11-18.

```
webSettings.setDefaultFontSize(25);
```

3. Set whether the WebView supports zoom:

Example 11-19.

```
webSettings.setSupportZoom(true);
```

4. Tell the WebView to enable javascript execution:

Example 11-20.

```
webSettings.setJavaScriptEnabled(true);
```

5. Store whether the WebView is saving password.

Example 11-21.

```
webSettings.setSavePassword(false);
```

6. Store whether the WebView is saving form data.

Example 11-22.

```
webSettings.setSaveFormData(false);
```

Many more methods of this kind are available. For more information, see the Android developers page on the topic.

Gaming and Animation

12.1 Introduction: Gaming and Animation

Ian Darwin

Discussion

Gaming is obviously an important part of what people used to use "computers" for and now use mobile devices for, and Android is a perfectly capable contender in the graphics arena, providing support for OpenGL ES.

If you want to use some advanced gaming features without having to write a lot of code, you're in luck, as there are many "games development" frameworks in existence today. Many of them are primarily or exclusively for desktops. The ones shown below are known to be usable on Android; if you find others, please add a Comment to this Recipe.

Table 12-1. Android Game Frameworks

Name	Open Source?	Cost	URL
AndEngine	Y	\$0	http://www.andengine.org/
Corona SDK	?	\$199+/year	http://www.anscamobile.com/corona/
Flixel	Y	\$0	http://flixel.org/index.html
ForPlay	Y		
libgdx	Y	\$0	http://code.google.com/p/libgdx/
rokon	Y	0	http://www.rokonandroid.com/
Shiva 3d	N	E169.00+ ea for editor and server	http://www.stonetrip.com/
Unity	N	\$400+	http://unity3d.com/unity/publishing/android.html

You will need to compare the functions that each offers before committing to use one or another in your project.

12.2 Android Game Programming - Introduction to Flixel-Android

Wagied Davids

Problem

Game design

Solution

File: Main.java

Example 12-1.

```
import android.app.Activity;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.Window;
import android.view.WindowManager;

public class Main extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);

        // ORIENTATION
        // setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);

        setContentView(new GameView(this, R.class));
    }
}
```

File: GameView.java

Example 12-2.

```
import org.flixel.FlxEGame;
import org.flixel.FlxEGameView;
import android.content.Context;

public class GameView extends FlxEGameView
{
    public GameView(Context context, Class<? extends Object> resource)
    {
        super(new FlxEGame(400, 240, SimpleJumper.class, context, resource), context);
    }
}
```

```
}  
}
```

File: Droid.java

Example 12-3.

```
import org.flixel.FlxC;
import org.flixel.FlxCSound;
import org.flixel.FlxCSprite;

public class Droid extends FlxCSprite
{
    private final FlxCSound sound = new FlxCSound();

    public Droid(int X, int Y)
    {
        super(X, Y);
        loadGraphic(R.drawable.player, true, true);
        maxVelocity.x = 100; // walking speed
        acceleration.y = 400; // gravity
        drag.x = maxVelocity.x * 4; // deceleration (sliding to a stop)

        // tweak the bounding box for better feel
        width = 8;
        height = 10;

        offset.x = 3;
        offset.y = 3;

        addAnimation("idle", new int[] { 0 }, 0, false);
        addAnimation("walk", new int[] { 1, 2, 3, 0 }, 12);
        addAnimation("walk back", new int[] { 3, 2, 1, 0 }, 10, true);
        addAnimation("flail", new int[] { 1, 2, 3, 0 }, 18, true);
        addAnimation("jump", new int[] { 4 }, 0, false);
    }

    @Override
    public void update()
    {
        // Smooth slidey walking controls
        acceleration.x = 0;
        if (FlxC.dpad.pressed("LEFT")) acceleration.x -= drag.x;
        if (FlxC.dpad.pressed("RIGHT")) acceleration.x += drag.x;

        if (onFloor)
        {
            // Jump controls
            if (FlxC.dpad.justTouched("UP"))
            {
                sound.loadEmbedded(R.raw.jump);
                sound.play();

                velocity.y = -acceleration.y * 0.51f;
                play("jump");
            }
        }
    }
}
```

```

        } // Animations
    else if (velocity.x > 0)
    {
        play("walk");
    }
    else if (velocity.x < 0)
    {
        play("walk_back");
    }
    else play("idle");
}
else if (velocity.y < 0) play("jump");
else play("flail");

// Default object physics update
super.update();
}
}

```

Discussion

Flixel is an Actionscript-based game framework developed by Adam ("Atomic") Salzman <http://flixel.org/index.html>.

Due to the tremendous work of Wing Eraser a Java-based port has been created <http://code.google.com/p/flixel-android/>, which closely resembles the AS3-based Flixel in programming paradigm.

In this tutorial, we will be creating a simple jumper game, containing a few entities, a droid, and pusher and a few elevators. Each entity is declared as a separate class containing its own asset resources, and listeners for digital touchpad events.

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.filefactory.com/file/cc0a426/n/SimpleJumper.zip>

12.3 Introduction to Game Programming using AndEngine (Android-Engine)

Wagied Davids

Problem

Game design and programming using the AndEngine game framework

Solution

Example 12-4.

```
import org.anddev.andengine.engine.Engine;
import org.anddev.andengine.engine.camera.Camera;
import org.anddev.andengine.engine.options.EngineOptions;
import org.anddev.andengine.engine.options.EngineOptions.ScreenOrientation;
import org.anddev.andengine.engine.options.resolutionpolicy.RatioResolutionPolicy;
import org.anddev.andengine.entity.Entity;
import org.anddev.andengine.entity.primitive.Rectangle;
import org.anddev.andengine.entity.scene.Scene;
import org.anddev.andengine.entity.scene.Scene.IOnAreaTouchListener;
import org.anddev.andengine.entity.scene.Scene.IOnSceneTouchListener;
import org.anddev.andengine.entity.scene.Scene.ITouchArea;
import org.anddev.andengine.entity.shape.Shape;
import org.anddev.andengine.entity.sprite.AnimatedSprite;
import org.anddev.andengine.entity.sprite.Sprite;
import org.anddev.andengine.entity.util.FPSLogger;
import org.anddev.andengine.extension.physics.box2d.PhysicsConnector;
import org.anddev.andengine.extension.physics.box2d.PhysicsFactory;
import org.anddev.andengine.extension.physics.box2d.PhysicsWorld;
import org.anddev.andengine.extension.physics.box2d.util.Vector2Pool;
import org.anddev.andengine.input.touch.TouchEvent;
import org.anddev.andengine.opengl.texture.Texture;
import org.anddev.andengine.opengl.texture.TextureOptions;
import org.anddev.andengine.opengl.texture.region.TextureRegion;
import org.anddev.andengine.opengl.texture.region.TextureRegionFactory;
import org.anddev.andengine.opengl.texture.region.TiledTextureRegion;
import org.anddev.andengine.sensor.accelerometer.AccelerometerData;
import org.anddev.andengine.sensor.accelerometer.IAccelerometerListener;
import org.anddev.andengine.ui.activity.BaseGameActivity;

import android.hardware.SensorManager;
import android.util.DisplayMetrics;

import com.badlogic.gdx.math.Vector2;
import com.badlogic.gdx.physics.box2d.Body;
import com.badlogic.gdx.physics.box2d.BodyDef.BodyType;
import com.badlogic.gdx.physics.box2d.FixtureDef;

public class SimplePool extends BaseGameActivity implements IAccelerometerListener, IOnSceneTouchListener, I
{
    private Camera mCamera;
    private Texture mTexture;
    private Texture mBallYellowTexture;
    private Texture mBallRedTexture;
    private Texture mBallBlackTexture;
    private Texture mBallBlueTexture;
    private Texture mBallGreenTexture;
    private Texture mBallOrangeTexture;
    private Texture mBallPinkTexture;
    private Texture mBallPurpleTexture;
    private Texture mBallWhiteTexture;
```

```

private TiledTextureRegion mBallYellowTextureRegion;
private TiledTextureRegion mBallRedTextureRegion;
private TiledTextureRegion mBallBlackTextureRegion;
private TiledTextureRegion mBallBlueTextureRegion;
private TiledTextureRegion mBallGreenTextureRegion;
private TiledTextureRegion mBallOrangeTextureRegion;
private TiledTextureRegion mBallPinkTextureRegion;
private TiledTextureRegion mBallPurpleTextureRegion;
private TiledTextureRegion mBallWhiteTextureRegion;

private Texture mBackgroundTexture;
private TextureRegion mBackgroundTextureRegion;

private PhysicsWorld mPhysicsWorld;

private float mGravityX;
private float mGravityY;
private Scene mScene;

private final int mFaceCount = 0;

private final int CAMERA_WIDTH = 720;
private final int CAMERA_HEIGHT = 480;

@Override
public Engine onLoadEngine()
{
    DisplayMetrics dm = new DisplayMetrics();
    getWindowManager().getDefaultDisplay().getMetrics(dm);

    this.mCamera = new Camera(0, 0, CAMERA_WIDTH, CAMERA_HEIGHT);
    return new Engine(new EngineOptions(true, ScreenOrientation.LANDSCAPE, new RatioResolutionPolicy(CAM
}

@Override
public void onLoadResources()
{
    this.mTexture = new Texture(64, 64, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
    this.mBallBlackTexture = new Texture(64, 64, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
    this.mBallBlueTexture = new Texture(64, 64, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
    this.mBallGreenTexture = new Texture(64, 64, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
    this.mBallOrangeTexture = new Texture(64, 64, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
    this.mBallPinkTexture = new Texture(64, 64, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
    this.mBallPurpleTexture = new Texture(64, 64, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
    this.mBallYellowTexture = new Texture(64, 64, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
    this.mBallRedTexture = new Texture(64, 64, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
    this.mBallWhiteTexture = new Texture(64, 64, TextureOptions.BILINEAR_PREMULTIPLYALPHA);

    TextureRegionFactory.setAssetBasePath("gfx/");
    mBallYellowTextureRegion = TextureRegionFactory.createTiledFromAsset(this.mBallYellowTexture, this,
    mBallRedTextureRegion = TextureRegionFactory.createTiledFromAsset(this.mBallRedTexture, this, "ball_
    mBallBlackTextureRegion = TextureRegionFactory.createTiledFromAsset(this.mBallBlackTexture, this, "b
    mBallBlueTextureRegion = TextureRegionFactory.createTiledFromAsset(this.mBallBlueTexture, this, "bal
    mBallGreenTextureRegion = TextureRegionFactory.createTiledFromAsset(this.mBallGreenTexture, this, "b

```

```

mBallOrangeTextureRegion = TextureRegionFactory.createTiledFromAsset(this.mBallOrangeTexture, this,
mBallPinkTextureRegion = TextureRegionFactory.createTiledFromAsset(this.mBallPinkTexture, this, "bal
mBallPurpleTextureRegion = TextureRegionFactory.createTiledFromAsset(this.mBallPurpleTexture, this,
mBallWhiteTextureRegion = TextureRegionFactory.createTiledFromAsset(this.mBallWhiteTexture, this, "b

this.mBackgroundTexture = new Texture(512, 1024, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
this.mBackgroundTextureRegion = TextureRegionFactory.createFromAsset(this.mBackgroundTexture, this,

this.enableAccelerometerSensor(this);

this.mEngine.getTextureManager().loadTextures(this.mBackgroundTexture, mBallYellowTexture, mBallRedT
}

@Override
public Scene onLoadScene()
{
    this.mEngine.registerUpdateHandler(new FPSLogger());

    this.mPhysicsWorld = new PhysicsWorld(new Vector2(0, SensorManager.GRAVITY_EARTH), false);

    this.mScene = new Scene();
    this.mScene.attachChild(new Entity());

    this.mScene.setBackgroundEnabled(false);
    this.mScene.setOnSceneTouchListener(this);
    Sprite background = new Sprite(0, 0, this.mBackgroundTextureRegion);
    background.setWidth(CAMERA_WIDTH);
    background.setHeight(CAMERA_HEIGHT);
    background.setPosition(0, 0);
    this.mScene.getChild(0).attachChild(background);

    final Shape ground = new Rectangle(0, CAMERA_HEIGHT, CAMERA_WIDTH, 0);
    final Shape roof = new Rectangle(0, 0, CAMERA_WIDTH, 0);
    final Shape left = new Rectangle(0, 0, 0, CAMERA_HEIGHT);
    final Shape right = new Rectangle(CAMERA_WIDTH, 0, 0, CAMERA_HEIGHT);

    final FixtureDef wallFixtureDef = PhysicsFactory.createFixtureDef(0, 0.5f, 0.5f);
    PhysicsFactory.createBoxBody(this.mPhysicsWorld, ground, BodyType.StaticBody, wallFixtureDef);
    PhysicsFactory.createBoxBody(this.mPhysicsWorld, roof, BodyType.StaticBody, wallFixtureDef);
    PhysicsFactory.createBoxBody(this.mPhysicsWorld, left, BodyType.StaticBody, wallFixtureDef);
    PhysicsFactory.createBoxBody(this.mPhysicsWorld, right, BodyType.StaticBody, wallFixtureDef);

    this.mScene.attachChild(ground);
    this.mScene.attachChild(roof);
    this.mScene.attachChild(left);
    this.mScene.attachChild(right);

    this.mScene.registerUpdateHandler(this.mPhysicsWorld);
    this.mScene.setOnAreaTouchListener(this);

    return this.mScene;
}

@Override
public void onLoadComplete()

```



```

    {
        setupBalls();
    }

@Override
public boolean onAreaTouched(final TouchEvent pSceneTouchEvent, final ITouchArea pTouchArea, final float
{
    if (pSceneTouchEvent.isActionDown())
    {
        final AnimatedSprite face = (AnimatedSprite) pTouchArea;
        this.jumpFace(face);
        return true;
    }

    return false;
}

@Override
public boolean onSceneTouchEvent(final Scene pScene, final TouchEvent pSceneTouchEvent)
{
    if (this.mPhysicsWorld != null)
    {
        if (pSceneTouchEvent.isActionDown())
        {
            // this.addFace(pSceneTouchEvent.getX(),
            // pSceneTouchEvent.getY());
            return true;
        }
    }
    return false;
}

@Override
public void onAccelerometerChanged(final AccelerometerData pAccelerometerData)
{
    this.mGravityX = pAccelerometerData.getX();
    this.mGravityY = pAccelerometerData.getY();

    final Vector2 gravity = Vector2Pool.obtain(this.mGravityX, this.mGravityY);
    this.mPhysicsWorld.setGravity(gravity);
    Vector2Pool.recycle(gravity);
}

private void setupBalls()
{
    final AnimatedSprite[] balls = new AnimatedSprite[9];

    final FixtureDef objectFixtureDef = PhysicsFactory.createFixtureDef(1, 0.5f, 0.5f);

    AnimatedSprite redBall = new AnimatedSprite(10, 10, this.mBallRedTextureRegion);
    AnimatedSprite yellowBall = new AnimatedSprite(20, 20, this.mBallYellowTextureRegion);
    AnimatedSprite blueBall = new AnimatedSprite(30, 30, this.mBallBlueTextureRegion);
    AnimatedSprite greenBall = new AnimatedSprite(40, 40, this.mBallGreenTextureRegion);
    AnimatedSprite orangeBall = new AnimatedSprite(50, 50, this.mBallOrangeTextureRegion);

```

```

AnimatedSprite pinkBall = new AnimatedSprite(60, 60, this.mBallPinkTextureRegion);
AnimatedSprite purpleBall = new AnimatedSprite(70, 70, this.mBallPurpleTextureRegion);
AnimatedSprite blackBall = new AnimatedSprite(70, 70, this.mBallBlackTextureRegion);
AnimatedSprite whiteBall = new AnimatedSprite(70, 70, this.mBallWhiteTextureRegion);

balls[0] = redBall;
balls[1] = yellowBall;
balls[2] = blueBall;
balls[3] = greenBall;
balls[4] = orangeBall;
balls[5] = pinkBall;
balls[6] = purpleBall;
balls[7] = blackBall;
balls[8] = whiteBall;

for (int i = 0; i < 9; i++)
{
    Body body = PhysicsFactory.createBoxBody(this.mPhysicsWorld, balls[i], BodyType.DynamicBody, obj
    this.mPhysicsWorld.registerPhysicsConnector(new PhysicsConnector(balls[i], body, true, true));

    balls[i].animate(new long[] { 200, 200 }, 0, 1, true);
    balls[i].setUserData(body);
    this.mScene.registerTouchArea(balls[i]);
    this.mScene.attachChild(balls[i]);
}

private void jumpFace(final AnimatedSprite face)
{
    final Body faceBody = (Body) face.getUserData();

    final Vector2 velocity = Vector2Pool.obtain(this.mGravityX * -50, this.mGravityY * -50);
    faceBody.setLinearVelocity(velocity);
    Vector2Pool.recycle(velocity);
}
}

```

Discussion

AndEngine is a game engine framework (<http://www.andengine.org/>) designed for producing games on Android. Originally developed by Nicholas Gramlich, it has some advanced features for producing awesome games.

In this example, I have designed a simple pool game with physics capabilities, such that effects of the accelerometer are taken into account but also touch events. When touching a specific billiard ball, and pulling down on it will cause it to shoot into other balls, with the collision detection taken care of.

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.filefactory.com/file/cc0a5b1/n/SimplePool.zip>

Social Networking

13.1 Facebook Integration

Wagied Davids

Problem

Integrate an Android application with Facebook

Solution

Steps involved:

1. Apply for a Facebook Application ID (APP_ID)
2. Include the Facebook APP_ID in your Android Application

Example 13-1.

```
Facebook mFacebook = new Facebook(getResources().getString(R.string.FACEBOOK_ID_TEST));
```

3. Make a call to Facebook for single-sign on authorization
4. In your onActivityResult() function

Example 13-2.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data)
{
    super.onActivityResult(requestCode, resultCode, data);
    mFacebook.authorizeCallback(requestCode, resultCode, data);
}
```

That's the basics (full code available in zipped Android-project).

Essential URLs: Facebook mobile: <http://developers.facebook.com/docs/guides/mobile/> Facebook Android Developers site: <https://github.com/facebook/facebook-android-sdk>

Discussion

File: facebook_login.xml

Example 13-3.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent" android:background="@color/white"
    android:gravity="center_horizontal">

    <com.facebook.android.LoginButton
        android:id="@+id/login" android:src="@drawable/login_button"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_margin="30dp" />

    <TextView android:id="@+id/txt" android:text="@string/hello"
        android:textColor="@color/black" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <Button android:id="@+id/uploadButton" android:text="@string/FACEBOOK_UPLOAD"
        android:visibility="invisible" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:paddingRight="20dp"
        android:paddingLeft="20dp" android:layout_margin="20dp" />

    <Button android:id="@+id/requestButton" android:text="@string/REQUEST"
        android:visibility="invisible" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:paddingRight="20dp"
        android:paddingLeft="20dp" android:layout_margin="20dp" />

    <Button android:id="@+id/postButton" android:text="@string/FACEBOOK_WALL_POST"
        android:visibility="invisible" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:paddingRight="20dp"
        android:paddingLeft="20dp" android:layout_margin="20dp" />

    <Button android:id="@+id/deletePostButton" android:text="@string/FACEBOOK_DELETE_POST"
        android:visibility="invisible" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:paddingRight="20dp"
        android:paddingLeft="20dp" android:layout_margin="20dp" />

</LinearLayout>
```

File: Main.java

Example 13-4.

```
package com.facebook.android;

import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
```

```

import java.net.URL;

import org.json.JSONException;
import org.json.JSONObject;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

import com.facebook.android.Facebook.DialogListener;
import com.facebook.android.SessionEvents.AuthListener;
import com.facebook.android.SessionEvents.LogoutListener;

public class Main extends Activity implements OnClickListener
{
    private static final String tag = "Main";
    private LoginButton mLoginButton;
    private TextView mText;
    private Button mRequestButton;
    private Button mPostButton;
    private Button mDeleteButton;
    private Button mUploadButton;

    private Facebook mFacebook;
    private AsyncFacebookRunner mAsyncRunner;

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        Log.d(tag, getResources().getString(R.string.CREATING_VIEW));
        mFacebook = new Facebook(getResources().getString(R.string.FACEBOOK_ID_TEST));
        setContentView(R.layout.facebook_login_view);

        mLoginButton = (LoginButton) this.findViewById(R.id.login);
        mText = (TextView) Main.this.findViewById(R.id.txt);
        mRequestButton = (Button) findViewById(R.id.requestButton);
        mPostButton = (Button) findViewById(R.id.postButton);
        mDeleteButton = (Button) findViewById(R.id.deletePostButton);
        mUploadButton = (Button) findViewById(R.id.uploadButton);

        mFacebook.authorize(this, new DialogListener()
        {
            @Override
            public void onComplete(Bundle values)
            {
            }

            @Override
            public void onFacebookError(FacebookError error)

```

```

        {
        }

        @Override
        public void onError(DialogError e)
        {
        }

        @Override
        public void onCancel()
        {
        }
    });

    //
    mAsyncRunner = new AsyncFacebookRunner(mFacebook);

    SessionStore.restore(mFacebook, this);
    SessionEvents.addAuthListener(new SampleAuthListener());
    SessionEvents.addLogoutListener(new SampleLogoutListener());
    mLoginButton.init(this, mFacebook);

    mRequestButton.setOnClickListener(this);

    mRequestButton.setVisibility(mFacebook.isSessionValid() ? View.VISIBLE : View.INVISIBLE);

    mUploadButton.setOnClickListener(this);

    mUploadButton.setVisibility(mFacebook.isSessionValid() ? View.VISIBLE : View.INVISIBLE);

    mPostButton.setOnClickListener(this);

    mPostButton.setVisibility(mFacebook.isSessionValid() ? View.VISIBLE : View.INVISIBLE);

    }
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data)
    {
        super.onActivityResult(requestCode, resultCode, data);
        mFacebook.authorizeCallback(requestCode, resultCode, data);
    }

    //
    public class SampleAuthListener implements AuthListener
    {

        @Override
        public void onAuthSucceed()
        {
            mText.setText("You have logged in! ");
            mRequestButton.setVisibility(View.VISIBLE);
            mUploadButton.setVisibility(View.VISIBLE);
            mPostButton.setVisibility(View.VISIBLE);
        }
    }

```

```

        @Override
        public void onAuthFail(String error)
        {
            mText.setText("Login Failed: " + error);
        }
    }

    public class SampleLogoutListener implements LogoutListener
    {
        @Override
        public void onLogoutBegin()
        {
            mText.setText("Logging out...");
        }

        @Override
        public void onLogoutFinish()
        {
            mText.setText("You have logged out! ");
            mRequestButton.setVisibility(View.INVISIBLE);
            mUploadButton.setVisibility(View.INVISIBLE);
            mPostButton.setVisibility(View.INVISIBLE);
        }
    }

    public class SampleRequestListener extends BaseRequestListener
    {
        @Override
        public void onComplete(final String response, final Object state)
        {
            try
            {
                // process the response here: executed in background thread
                Log.d("Facebook-Example", "Response: " + response.toString());
                JSONObject json = Util.parseJson(response);
                final String name = json.getString("name");

                // then post the processed result back to the UI thread
                // if we do not do this, an runtime exception will be generated
                // e.g. "CalledFromWrongThreadException: Only the original
                // thread that created a view hierarchy can touch its views."
                Main.this.runOnUiThread(new Runnable()
                {
                    @Override
                    public void run()
                    {
                        mText.setText("Hello there, " + name + "!");
                    }
                });
            }
            catch (JSONException e)
            {
                Log.w("Facebook-Example", "JSON Error in response");
            }
        }
    }

```



```

        catch (FacebookError e)
        {
            Log.w("Facebook-Example", "Facebook Error: " + e.getMessage());
        }
    }
}

public class SampleUploadListener extends BaseRequestListener
{
    @Override
    public void onComplete(final String response, final Object state)
    {
        try
        {
            // process the response here: (executed in background thread)
            Log.d("Facebook-Example", "Response: " + response.toString());
            JSONObject json = Util.parseJson(response);
            final String src = json.getString("src");

            // then post the processed result back to the UI thread
            // if we do not do this, an runtime exception will be generated
            // e.g. "CalledFromWrongThreadException: Only the original
            // thread that created a view hierarchy can touch its views."
            Main.this.runOnUiThread(new Runnable()
            {
                @Override
                public void run()
                {
                    mText.setText("Hello there, photo has been uploaded at \n" +
                        src);
                }
            });
        }
        catch (JSONException e)
        {
            Log.w("Facebook-Example", "JSON Error in response");
        }
        catch (FacebookError e)
        {
            Log.w("Facebook-Example", "Facebook Error: " + e.getMessage());
        }
    }
}

public class WallPostRequestListener extends BaseRequestListener
{
    @Override
    public void onComplete(final String response, final Object state)
    {
        Log.d("Facebook-Example", "Got response: " + response);
        String message = "<empty>";
        try
        {
            JSONObject json = Util.parseJson(response);

```

```

        message = json.getString("message");
    }
    catch (JSONException e)
    {
        Log.w("Facebook-Example", "JSON Error in response");
    }
    catch (FacebookError e)
    {
        Log.w("Facebook-Example", "Facebook Error: " + e.getMessage());
    }
    final String text = "Your Wall Post: " + message;
    Main.this.runOnUiThread(new Runnable()
    {
        @Override
        public void run()
        {
            mText.setText(text);
        }
    });
}

public class WallPostDeleteListener extends BaseRequestListener
{
    @Override
    public void onComplete(final String response, final Object state)
    {
        if (response.equals("true"))
        {
            Log.d("Facebook-Example", "Successfully deleted wall post");
            Main.this.runOnUiThread(new Runnable()
            {
                @Override
                public void run()
                {
                    mDeleteButton.setVisibility(View.INVISIBLE);
                    mText.setText("Deleted Wall Post");
                }
            });
        }
        else
        {
            Log.d("Facebook-Example", "Could not delete wall post");
        }
    }
}

public class SampleDialogListener extends BaseDialogListener
{
    @Override
    public void onComplete(Bundle values)
    {
        final String postId = values.getString("post_id");
    }
}

```

```

        if (postId != null)
        {
            Log.d("Facebook-Example", "Dialog Success! post_id=" + postId);
            mAsyncRunner.request(postId, new WallPostRequestListener());
            mDeleteButton.setOnClickListener(new OnClickListener()
            {
                @Override
                public void onClick(View v)
                {
                    mAsyncRunner.request(postId, new Bundle(), "DELETE", new Wal
                });
            });
            mDeleteButton.setVisibility(View.VISIBLE);
        }
    else
    {
        Log.d("Facebook-Example", "No wall post made");
    }
}

/*
 * (non-Javadoc)
 *
 * @see android.view.View.OnClickListener#onClick(android.view.View)
 */
@Override
public void onClick(View v)
{
    if (v == mLoginButton)
    {
    }

    if (v == mRequestButton)
    {
        mAsyncRunner.request("me", new SampleRequestListener());
    }

    if (v == mUploadButton)
    {
        Bundle params = new Bundle();
        params.putString("method", "photos.upload");

        URL uploadFileUrl = null;
        try
        {
            uploadFileUrl = new URL("http://www.facebook.com/images/devsite/iphone_conne
        }
        catch (MalformedURLException e)
        {
            e.printStackTrace();
        }
    }
}

```

```

        {
            HttpURLConnection conn = (HttpURLConnection) uploadFileUrl.openConnection();
            conn.setDoInput(true);
            conn.connect();
            int length = conn.getContentLength();

            byte[] imgData = new byte[length];
            InputStream is = conn.getInputStream();
            is.read(imgData);
            params.putByteArray("picture", imgData);
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }

        mAsyncRunner.request(null, params, "POST", new SampleUploadListener(), null);
    }

    if (v == mPostButton)
    {
        mFacebook.dialog(Main.this, "feed", new SampleDialogListener());
    }
    if (v == mDeleteButton)
    {
    }
}
}
}

```

Binary Download URL

The executable code for this example may be downloaded from this URL: <http://www.filefactory.com/file/cbd661e/n/Facebook-Example.zip>

13.2 Social Networking Integration using Http

Shraddha Shrivagi

Problem

Instead of diving into the API you can simply add Social networking support.

Solution

For Facebook, Twitter and LinkedIn integration, just follow 3 simple steps to get started:

1. Get the logo's for Facebook, Twitter and Linked-in
2. Create image buttons for each of them

3. Implement the event handler

Discussion

Here goes the detailed explanation:

1. Get the logo:

Just download the logo from the site, or use a web search engine.

2. Create image buttons for each of them

Example 13-5.

```
<!-- Facebook button -->
<ImageView android:src="@drawable/icon_facebook"
    android:layout_width="28dip"
    android:layout_height="28dip" android:id="@+id/facebookBtn"
    android:clickable="true"
    android:onClick="facebookBtnClicked" />

<!-- Twitter button -->
<ImageView android:src="@drawable/icon_twitter"
    android:clickable="true"
    android:layout_width="30dip" android:layout_height="28dip" android:id="@+id/twitterBtn"
    android:layout_marginLeft="3dp" android:layout_marginRight="3dp"
    android:onClick="twitterBtnClicked"
    />

<!-- LinkedIn button -->
<ImageView android:src="@drawable/icon_linkedin"
    android:layout_width="28dip"
    android:layout_height="30dip" android:clickable="true"
    android:id="@+id/linkedinBtn"
    android:onClick="linkedinBtnClicked"
    />
```

3. Implement the click event

Example 13-6.

```
/*URL used here is for application to which I want user to redirect and comment about here I am
 * using http://goo.gl/eRAD9 as the URL. But you can use URL of your app. Take the URL from Market
 * and shorten the URL with bit.ly or Google URL shortener
 * */

public void facebookBtnClicked(View v)
{
    Toast.makeText(this, "Facebook Loading...\n Please make sure you are connected to internet.", Toast.
    String url="http://m.facebook.com/sharer.php?u=http%3A%2F%2Fgoo.gl%2FeRAD9";
    Intent i = new Intent(Intent.ACTION_VIEW);
    i.setData(Uri.parse(url));
    startActivity(i);
}
```



Figure 13-1.

```

public void twitterBtnClicked(View v)
{
    Toast.makeText(this, "Twitter Loading... \n Please make sure you are connected to internet.", Toast.
    /**/
    String url = "http://www.twitter.com/share?text=Checkout+This+Demo+http://goo.gl/eRAD9+";
    Intent i = new Intent(Intent.ACTION_VIEW);
    i.setData(Uri.parse(url));
    startActivity(i);
}
public void linkedinBtnClicked(View v)
{
    Toast.makeText(this, "Linked-In Loading... \n Please make sure you are connected to internet", Toast.
    String url="http://www.linkedin.com/shareArticle?url=http%3A%2F%2Fgoo.gl%2FeRAD9&mini=true&source=Sa
    Intent intent=new Intent(Intent.ACTION_VIEW);
    intent.setData(Uri.parse(url));
    startActivity(intent);
}
}

```

This is how in 3 simple steps you can get a Social Networking feature for your application.

13.3 Loading a user's Twitter timeline (using JSON)

Rachee Singh

Problem

You want to load the Twitter timeline of a user in an Android application.

Solution

Since timeline information is public, you don't need to deal with Twitter's authentication. You can just use `HttpGet` to obtain the data from the user's Twitter page in JSON format. Then, the JSON can be processed to obtain the tweets by the user.

Discussion

`HttpGet` is used to obtain data from the twitter page of the Times of India (a newspaper). From the response obtained after executing the request contains data from the twitter page in JSON format. We check for the status code and unless the code is 200, the request could not fetch the data. From the response we obtain the JSON and put it into the `StringBuilder` object. The `getTwitterTimeline()` method returns the `String` that contains the data in JSON format.

Example 13-7.

```

public String getTwitterTimeline() {
    StringBuilder builder = new StringBuilder();
    HttpClient client = new DefaultHttpClient();
    HttpGet httpGet = new HttpGet("http://twitter.com/statuses/user_timeline/timesofindia.json");
}

```

```

try {
    HttpResponse response = client.execute(httpGet);
    StatusLine statusLine = response.getStatusLine();
    int statusCode = statusLine.getStatusCode();
    if (statusCode == 200) {
        HttpEntity entity = response.getEntity();
        InputStream content = entity.getContent();
        BufferedReader reader = new BufferedReader(new InputStreamReader(content));
        String line;
        while ((line = reader.readLine()) != null) {
            builder.append(line);
        }
    } else {
        //Couldn't obtain the data
    }
} catch (ClientProtocolException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
}
return builder.toString();
}
}

```

Now we process the JSON returned from the `getTwitterTimeline()` method in the standard way, using the `getString()` method. These we insert into a `TextView`.

Example 13-8.

```

String twitterTimeline = getTwitterTimeline();
try {
    String tweets = "";
    JSONArray jsonArray = new JSONArray(twitterTimeline);
    for (int i = 0; i < jsonArray.length(); i++) {
        JSONObject jsonObject = jsonArray.getJSONObject(i);
        int j = i+1;
        tweets += "*** " + j + " ***\n";
        tweets += "Date:" + jsonObject.getString("created_at") + "\n";
        tweets += "Post:" + jsonObject.getString("text") + "\n\n";
    }
    json= (TextView)findViewById(R.id.json);
    json.setText(tweets);
} catch (JSONException e) {
    e.printStackTrace();
}
}

```

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LZDE3MzIxNmYtMDU3Yy00OTZjLTk2NTgtMDBiNTZiYjdlYzlm&hl=en_US

TwitterJSON

*** 14 ***

Date: Mon Jul 18 15:21:54 +0000 2011

Post: Lokayukta asks President to censure Sheila Dikshit <http://toi.in/u1WgLb>

*** 15 ***

Date: Mon Jul 18 14:31:54 +0000 2011

Post: Gavaskar, Kapil, Sachin, Sehwag in ICC's all-time Test XI <http://toi.in/wifHKZ>

*** 16 ***

Date: Mon Jul 18 11:12:36 +0000 2011

Post: Darjeeling tripartite pact signed for Gorkhaland Territorial Administration <http://toi.in/1umZ7Y>

*** 17 ***

Date: Mon Jul 18 10:21:46 +0000 2011

Post: UP poll: Holy Ganga becomes a hot political commodity <http://toi.in/vgciyb>

*** 18 ***

Date: Mon Jul 18 07:52:25 +0000 2011

Post: Now, Gujarati encyclopaedia, lexicon on your mobile phone <http://toi.in/bIBAbb>

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LZGYyNmYwYjktYjQwNS00NjYwLTg1MTktOGY2ZGQ3ODZlYjZj&hl=en_US

Location and Map Applications

14.1 Introduction: Location-Aware Applications

Ian Darwin

Discussion

Not that long ago, GPS devices were either unavailable, expensive, or cumbersome. Today, almost every smartphone has a GPS receiver, and many digital cameras do too. GPS is well on its way to becoming truly ubiquitous in devices. The organizations that provide map data have not been unaware of this trend. Indeed, [OpenStreetMap](#) exists and provides its "free, editable map of the world" in part because of the rise of consumer GPS devices - most of its map was made by enthusiasts. Google gets much of its data from commercial mapping services, but in Android, Google has been very driven by the availability of GPS receivers in Android devices. This chapter thus concentrates on the ins and outs of using Google Maps and OpenStreetmap in Android devices.

14.2 Getting Location Information

Ian Darwin

Problem

You want to know where you are.

Solution

Android provides two levels of locational position. If you need to know fairly precisely where you are, you can use the "FINE" resolution, which is GPS-based. If you only need to know roughly where you are, you can use the "COARSE" resolution, which is based on the location of the cell phone tower your phone is talking to.

Discussion

Here is the setup portion of the code. This is part of *jpstrack*, a mapping application for *OpenStreetMap*. For mapping purpose the GPS is a must, so I only ask for the FINE resolution.

Example 14-1.

```
// Part of jpstrack Main.java
LocationManager mgr =
    (LocationManager) getSystemService(LOCATION_SERVICE);
for (String prov : mgr.getAllProviders()) {
    Log.i(LOG_TAG, getString(R.string.provider_found) + prov);
}

// GPS setup
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_FINE);
List<String> providers = mgr.getProviders(criteria, true);
if (providers == null || providers.size() == 0) {
    Log.e(JPSTRACK, getString(R.string.cannot_get_gps_service));
    Toast.makeText(this, "Could not open GPS service",
        Toast.LENGTH_LONG).show();
    return;
}
String preferred = providers.get(0); // first == preferred
```

After this setup, when you actually want to start the GPS sending you location data, you have to call the `LocationManager.requestLocationUpdates` with the name of the provider you looked up previously, the minimum time between updates (in milliseconds), the minimum distance between updates (in meters), and an instance of the `LocationListener` interface. You stop updates by `removeUpdates` with the previously-passed-in `LocationListener`. In *jpstrack* the code looks like this:

Example 14-2.

```
@Override
protected void onResume() {
    super.onResume();
    if (preferred != null) {
        mgr.requestLocationUpdates(preferred,
            MIN_SECONDS * 1000,
            MIN_METRES, this);
    }
}

@Override
protected void onPause() {
    super.onPause();
    if (preferred != null) {
        mgr.removeUpdates(this);
    }
}
```

Finally, the `LocationListener`'s `onLocationChanged()` method is called when the location changes, and this is where you do something with the location information.

Example 14-3.

```
@Override
public void onLocationChanged(Location location) {
    long time = location.getTime();
    double latitude = location.getLatitude();
    double longitude = location.getLongitude();
    // do something with latitude and longitude (and time?)...
}
```

There are a few other methods in `LocationListener` but they aren't required to do anything.

What you do with the location data depends on your application, of course. In `jpstrack` I save it into a track file with hand-written XML-writing code. Commonly you would use it to update your position on a map, or upload it to a location service. There's no limit to what you can do with it.

Source Download URL

The source code for this example may be downloaded from this URL: <http://www.darwinsys.com/jpstrack/>

Binary Download URL

The executable code for this example may be downloaded from this URL: <http://www.darwinsys.com/jpstrack/jpstrack.android.apk>

14.3 Access GPS information anywhere in your application

Pratik Rupwal

Problem

You need access the GPS location in any class of your application.

Solution

Add a class (say 'MyLocationListener') which implements 'LocationListener' interface, create an instance of this class where you want to access the GPS information and use it for the relevant requirement.

Discussion

Below class 'MyLocationListener' implements 'LocationListener'

Example 14-4.

```
public class MyLocationListener implements LocationListener
{
    @Override
    public void onLocationChanged(Location loc)
    {
        loc.getLatitude();
        loc.getLongitude();
        /** The above 'Location' object 'loc' can be used for accessing GPS information, how
        in an application to perform all the GPS information related tasks in this overridden
        data accessibility.
        for example in an application providing the information of shopping malls n
        so app accesses the names of malls according to the user's location and displays the
        app displays the different offers given by that mall.
        In this example the location of the user decides the mall's name to be fetc
        database handler which is private member of the class hosting the view to display li
        handler can't be accessible in this overridden method hence this operation cannot be
    }

    @Override
    public void onProviderDisabled(String provider)
    {
    }

    @Override
    public void onProviderEnabled(String provider)
    {
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras)
    {
    }
}
} // End of Class MyLocationListener.
```

Add the above class file in package of your application, its instance can be used as below for accessing GPS information in any class.

Example 14-5.

```
public class AccessGPS extends Activity
{
    //declaration of required objects

    LocationManager mlocManager;
    LocationListener mlocListener;
    Location lastKnownLocation;
    Double latitude, longitude;
    .....
    .....

    protected void onCreate(Bundle savedInstanceState)
```

```

{
    .....
    .....
    //instantiating objects for accessing GPS information.

    mlocListener = new MyLocationListener();

    //reuest for location updates

    mlocManager.requestLocationUpdates( LocationManager.GPS_PROVIDER, 0, 0, mlocListener);
    locationProvider=LocationManager.GPS_PROVIDER;
    .....
    .....

    // Access the last identified location

    lastKnownLocation = mlocManager.getLastKnownLocation(locationProvider);

    // The above object can be used for accessing GPS data as below

    latitude=lastKnownLocation.getLatitude();
    longitude=lastKnownLocation.getLongitude();

    // The above GPS data can be used for carrying out the operations specific to the location.
    .....
    .....
}
}

```

14.4 Mocking GPS Coordinates On A Device

Emaad Manzoor

Problem

You need to demonstrate your application, but are scared it might choke trying to triangulate your GPS coordinates. Or you'd like to simulate being in a place you're not.

Solution

Attach a mock location provider to your LocationManager object, then attach mock coordinates to the mock location provider.

Discussion

Write The setMockLocation Method

This function is what you will eventually use in your application to set mock GPS coordinates on the device.

Example 14-6.

```
private void setMockLocation(double latitude, double longitude, float accuracy) {
    lm.addTestProvider (LocationManager.GPS_PROVIDER,
        "requiresNetwork" == "",
        "requiresSatellite" == "",
        "requiresCell" == "",
        "hasMonetaryCost" == "",
        "supportsAltitude" == "",
        "supportsSpeed" == "",
        "supportsBearing" == "",
        android.location.Criteria.POWER_LOW,
        android.location.Criteria.ACCURACY_FINE);

    Location newLocation = new Location(LocationManager.GPS_PROVIDER);

    newLocation.setLatitude(latitude);
    newLocation.setLongitude(longitude);
    newLocation.setAccuracy(accuracy);

    lm.setTestProviderEnabled(LocationManager.GPS_PROVIDER, true);

    lm.setTestProviderStatus(LocationManager.GPS_PROVIDER,
        LocationProvider.AVAILABLE,
        null, System.currentTimeMillis());

    lm.setTestProviderLocation(LocationManager.GPS_PROVIDER, newLocation);
}
```

What's Happening? Add a mock provider to the LocationManager lm: The *addTestProvider* method of the *LocationManager* class enables the creation and configuration of a mock location provider.

Create a new location: The *Location* object allows you to set its *Latitude*, *Longitude* and *Accuracy*.

Activate the mock provider: Set a mock *enabled* value for the LocationManager, set a mock *status* and then set a mock *location*.

Use The setMockLocation Method

To use the method, you must create a *LocationManager* object as you usually would, and then invoke the method with your coordinates.

Example 14-7.

```
LocationManager lm = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

lm.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, new LocationListener() {
    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {}
    @Override
    public void onProviderEnabled(String provider) {}
});
```

```
        @Override
        public void onProviderDisabled(String provider) {}
        @Override
        public void onLocationChanged(Location location) {}
    });

    /* Set a mock location for debugging purposes */
    setMockLocation(15.387653, 73.872585, 500);
```

Note: You may need to restart the device after using the mock GPS to re-enable the real device GPS

Example Application Usage

Find Me X: This Android application takes in a search query of the form "churches in Vasco Goa" (*place_type* in *locality city*) and returns results augmented with their distance from the user. The location in this application is mocked to be BITS - Pilani Goa Campus, Goa, India.

See Also

[Recipe 14.2](#)

Source Download URL

The source code for this example may be downloaded from this URL: <https://github.com/emaadmanzoor/findmex>

14.5 Geocoding and Reverse Geocoding

Nidhin Jose Davis

Problem

How to Geocode and Reverse Geocode on Android

Solution

The solution is to use Geocoder class.

Discussion

Geocoding is the process of finding the geographical coordinates (latitude and longitude) of given address or location.

Reverse Geocoding as you might have guessed is the opposite of Geocoding. In this case a pair of latitude and longitude is converted into an address or location.

In order to Geocode or Reverse Geocode the first thing to do is to import the proper package.

Example 14-8.

```
import android.location.Geocoder;
```

The Geocoding or Reverse Geocoding should not be done on the UI thread as it might cause the system to display an Application Not Responding (ANR) dialog to the user. So it has to be done in a separate thread.

To Geocode

Example 14-9.

```
Geocoder gc = new Geocoder(context);

if(gc.isPresent()){
    List<Address> list = gc.getFromLocationName("1600 Amphitheatre Parkway, Mountain View, CA", 1);

    Address address = list.get(0);

    double lat = address.getLatitude();
    double lng = address.getLongitude();
}
```

To Reverse Geocode

Example 14-10.

```
Geocoder gc = new Geocoder(context);

if(gc.isPresent()){
    List<Address> list = gc.getFromLocation(37.42279, -122.08506,1);

    Address address = list.get(0);

    StringBuffer str = new StringBuffer();
    str.append("Name: " + address.getLocality() + "\n");
    str.append("Sub-Admin Area: " + address.getSubAdminArea() + "\n");
    str.append("Admin Area: " + address.getAdminArea() + "\n");
    str.append("Country: " + address.getCountryName() + "\n");
    str.append("Country Code: " + address.getCountryCode() + "\n");

    String strAddress = str.toString();
}
```

14.6 Getting ready for Google Maps development

Johan Pelgrim

Problem

You want to get ready to include Google `MapView` layout elements in your Android app

Solution

Use the Google Maps API library, a `MapView` layout element and the `MapActivity`.

Discussion

Let's dig right in by creating an Android project which displays a default map

Setting up an AVD which makes use of the Google API SDK libraries.

When you create a new Android project you have to indicate which minimum SDK-version your app needs and which SDK-version you target. Since we will be using the Google Maps API we have to make sure we have an AVD with those libraries pre-installed. If you don't work with Google Maps inside your project it is fine to work with an AVD without the Google Maps APIs. In our case we are dependent on them.

Make sure you have an AVD with a build target of "Google APIs - 1.5 - API level 3".

Create a new Android project which targets "Google APIs - 1.5 - API level 3"

Creating a `MapTest` project which targets the "Google APIs - 1.5 - API level 3" and uses `minSdkVersion 3`. Let the Android Project wizard create a `MapTest` activity for you. Click finish.

The `MapView` element can only live inside a `MapActivity`, so make sure the `MapTest` activity extends that class. A `MapActivity` must implement the `isRouteDisplayed()` method. This method is required for some accounting from the Maps service to see if you're currently displaying any route information. In this example, we are not. We still have to implement the method, but it's ok to simply return `false` for now. To be able to zoom in the map we can set the build-in zoom controls to `true` by calling the `setBuiltInZoomControls` method on the `MapView` object.

Example 14-11.

```
package nl.codestone.cookbook.maptest;

import android.os.Bundle;

import com.google.android.maps.MapActivity;

public class MapTest extends <tt>MapActivity</tt> {

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

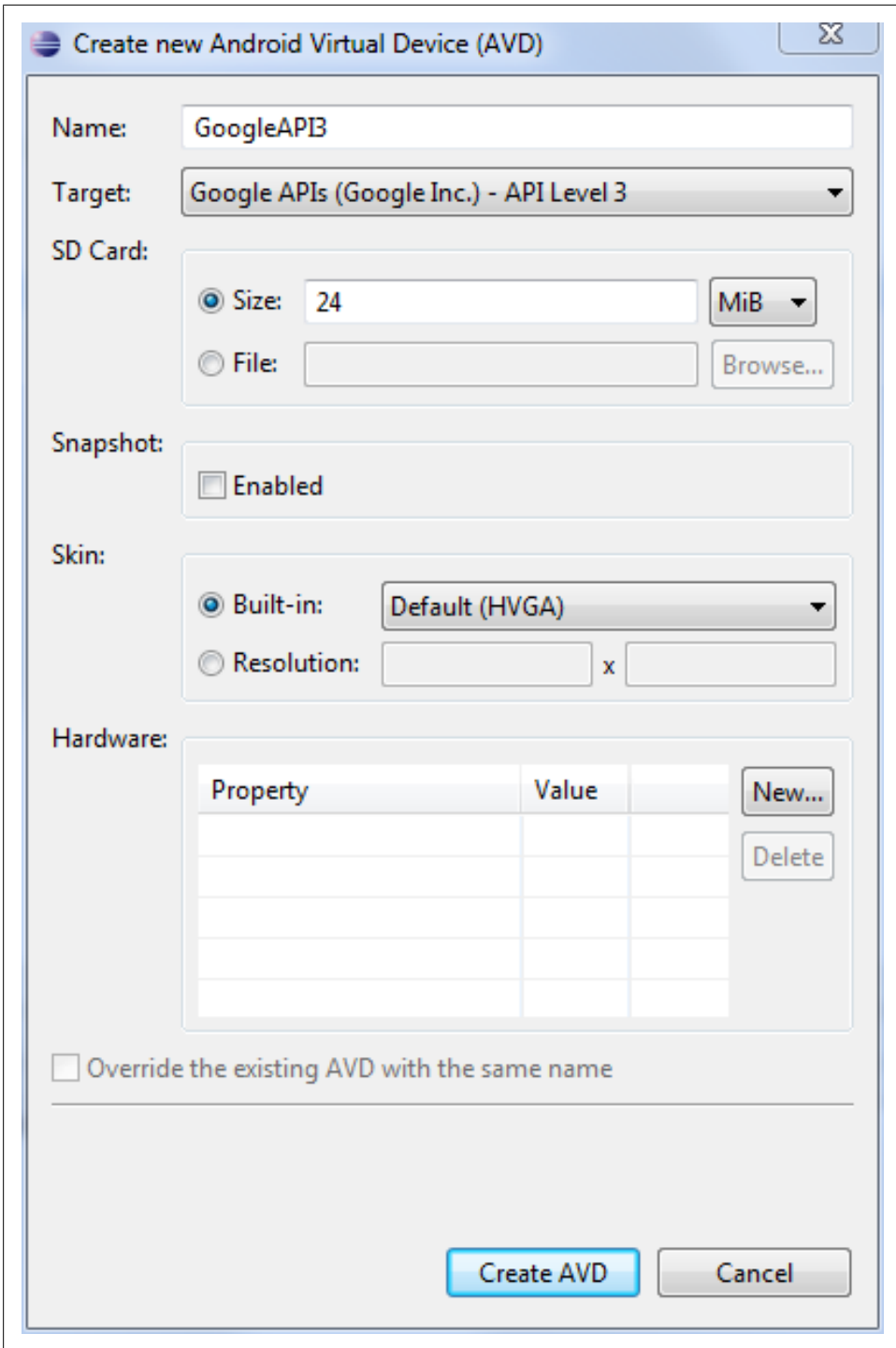


Figure 14-1.

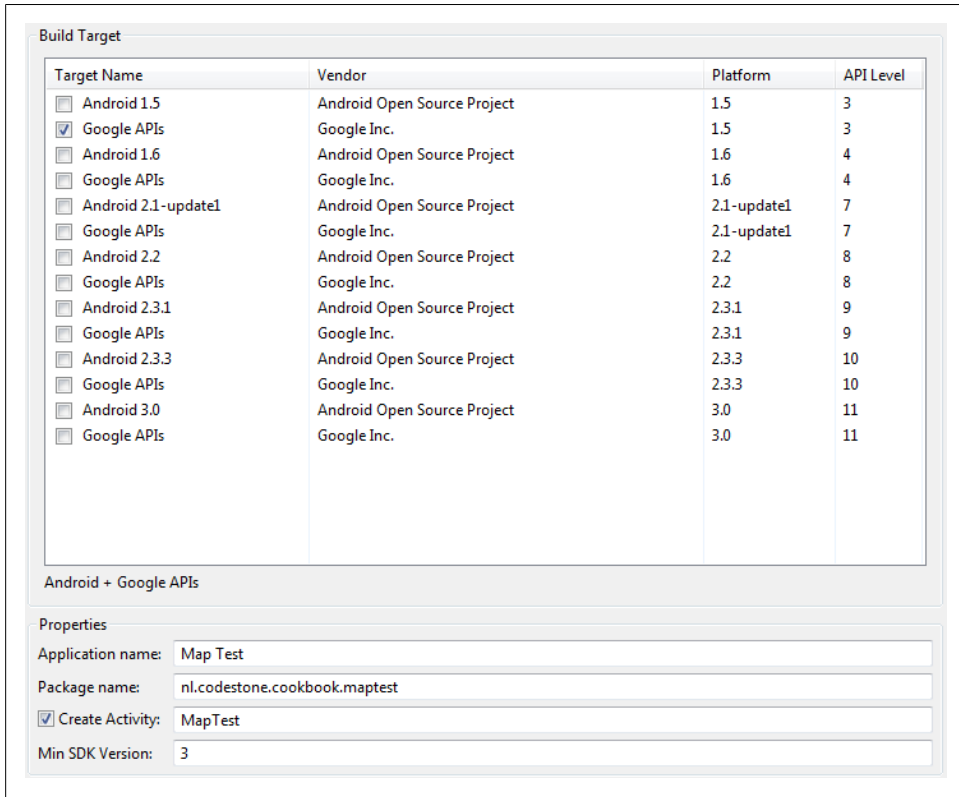


Figure 14-2.

```

MapView mapView = (MapView) findViewById(R.id.mapview);
mapview.setBuiltInZoomControls(true);

}

@Override
protected boolean isRouteDisplayed() {
    return false;
}
}

```

Adding the MapView element to your layout file

Open the `res/layout/main.xml` file. Delete the `TextView` element and replace it with a `MapView` element

Example 14-12.

```

<com.google.android.maps.MapView
    android:id="@+id/mapview"

```

```
android:layout_width="fill_parent"  
android:layout_height="fill_parent"  
android:apiKey="your_api_key_here"  
android:clickable="true"  
</>
```

Some highlights here

- The `MapView` is not part of the standard `com.android.view` package so we have to include the full package name in this element.
- We have to set the `android:clickable` attribute to `true` to be able to drag the map and zoom in and out.
- Your `MapView` object has to be configured with a personalized Google Maps API key in a special attribute `android:apiKey` on the `MapView` definition. This key can be obtained by registering your MD5 hash from the keystore you sign your apps with (or the `debug.keystore` during your development cycle).

Registering the Google Maps API key

A full description on how to register a Google Maps API key is given here: <http://code.google.com/android/add-ons/google-apis/mapkey.html>

This section extracts the minimal steps to get such a key. If you get stuck please refer to the full description by Google.

Android applications have to be signed with a certificate. These certificates are kept in a keystore. For your commercial apps you have to work with a private (self-signed) certificate which is imported in a keystore. When you create and deploy Android applications in your development environment a `debug.keystore` is used to sign your applications. This debug keystore is located in a `.android` directory in your user-directory. You need your private `androiddebugkey` key entry's fingerprint (MD5 hash) to register for a Google Maps API key.

Open a command shell and change to the `.android` directory which is located in your user directory (e.g. `cd ~/.android` in unix-like environments)

Issue the following command: `keytool -list -alias androiddebugkey -keystore debug.keystore -storepass android`

You will be presented with something like this:

Example 14-13.

```
androiddebugkey, 29-mrt-2011, PrivateKeyEntry,  
Certificate fingerprint (MD5): 2E:54:39:DB:33:E7:D6:3A:9E:18:3D:7F:FB:6D:BC:8D
```

Copy the bit after `Certificate fingerprint (MD5):` to your clipboard and go to this page to sign up for a Google Maps API key

<http://code.google.com/android/maps-api-signup.html>

You'll receive a key like this:

18Qcs3h-Sq518A7L56bjLwY1gwxgeMYF9Rp_0Cg

Copy and paste this key in the `android:apiKey` attribute in the `MapView` element in your `res/main.xml` layout file. If you are instantiating a `MapView` directly from code, you should pass the Maps API Key in the `MapView` constructor.

Tip: You can always regenerate the key as described in the above steps, so there's no need to keep this key somewhere safe. On the other hand, you'd better make a copy of the keystore you use for signing your personal apps!

Necessary changes in the `AndroidManifest.xml` file

- You have to register a `<uses-permission android:name="android.permission.INTERNET" />` in your `AndroidManifest.xml` to be able to get Map tiles information from the internet. These map tiles are automatically cached in your `apps-data` directory, so you don't have to do anything extra for that.
- The Google Maps classes are not standard, so you have to indicate you use the `com.google.android.maps` library in your `AndroidManifest.xml` file.

Example `AndroidManifest.xml` file

Example 14-14.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="nl.codestone.cookbook.maptest"
    android:versionCode="1"
    android:versionName="1.0">
    <uses-sdk android:minSdkVersion="3" />

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

    <application android:icon="@drawable/icon" android:label="@string/app_name">

        <activity android:name=".MapTest"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <uses-library android:name="com.google.android.maps" />

    </application>
</manifest>
```

There's another file called `default.properties` which contains the build target (level) of your app. This file is automatically generated when you created this project, so no need to change anything. It is good to know that the build target level is defined here

if you decide to increase or decrease it at some point. You can either change the level in this file or do it via the project properties dialog in Eclipse.

Example 14-15.

```
target=Google Inc.:Google APIs:3
```

That's it! Start your AVD and run your Android Application. If all's well you should see a map of North and South America which you can drag around and zoom into!

Check List

We end this Recipe with a check list which you can use for quickly setting up projects for the other Google Maps recipes:

- Use an AVD which makes use of the Google API SDK libraries.
- Your `Activity` should extend the `MapActivity` class.
- You must implement the `isRouteDisplayed()` method. The default -- let it return `false` -- is fine in most cases.
- Set the build-in zoom controls to true by calling the `setBuiltInZoomControls` method on the `MapView` object.
- Added the full package name to the `MapView` element in your layout file (i.e. `com.google.android.maps.MapView`)
- Add your Google Maps API key to the `android:apiKey` attribute on the `MapView` element.
- If you are instantiating a `MapView` directly from code, you should pass the Google Maps API Key directly in the `MapView` constructor.
- Set the `android:clickable` attribute on the `MapView` element to `true` to be able to drag the map and zoom in and out.
- Register a `<uses-permission android:name="android.permission.INTERNET" />` as a child of the `manifest` element in your `AndroidManifest.xml`
- Register a `<uses-library android:name="com.google.android.maps" />` as a child of the `application` element in your `AndroidManifest.xml`

See Also

Google APIs project on Google Code - <http://code.google.com/android/add-ons/google-apis> Google API key signup page - <http://code.google.com/android/maps-api-signup.html>

Source Download URL

The source code for this example may be downloaded from this URL: <https://github.com/downloads/jpelgrim/androidcookbook/MapTest.zip>



Figure 14-3.

14.7 Using Google Maps in your Android App

Oscar Salguero

Problem

Sometimes the steps needed to add a Google Map and show the User's location to Android Apps is not clear as water on the Internet and Documentation.

Solution

To be completed later...

Discussion

To be completed later...

14.8 How to show your current location in a map

Enrique Diaz

Problem

You want to see where you are in a map using the built-in GPS in Android devices

Solution

We can take advantage of the Google APIs and Location-Based Services to allow people to find their friends, find relevant places and many more. In this snippet of code, we can read the GPS location and see where we are located in a map called from the internet.

Discussion

This example of code shows how to create a MapActivity to show where are we located. Also, we're going to add some permissions to allow our project read the latitude and longitude based in the Google Maps API.

Step 1 Create a new project called MyCurrentLocation with an Activity called MyCurrentLocation. Because we need to obtain the Latitude and Longitude from the GPS, we need to add the following permissions in the Android Manifest:

Example 14-16.

```
&lt;?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.androidtitlan.mycurrentlocation"
    android:versionCode="1"
    android:versionName="1.0">
```

```

<application android:icon="@drawable/icon" android:label="@string/app_name">
  <activity android:name=".MyCurrentLocation"
    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="4" />

<uses-permission android:name="android.permission.INTERNET"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
</manifest>

```

14.9 To Add Device's current location to Google Maps

Rachee Singh

Problem

Adding Current location of the device on Google maps.

Solution

Using MyLocationOverlay class, current location of the device can be depicted on the Map.

Discussion

Add the following permissions to the Android Manifest File:

Example 14-17.

```

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

```

Adding a mapView to your application, these few lines of code should be present in the XML layout. The Id of the map view is 'map'.

Example 14-18.

```

<com.google.android.maps.MapView
  android:id="@+id/map"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:layout_below="@id/map_location_button"
  android:layout_above="@+id/use_this_location_button"
  android:clickable="true"
  android:apiKey = "Your API Key Should be placed here"/>

```

In the Java class for the activity which displays the map View, add a field:

Example 14-19.

```
private MyLocationOverlay myLocationOverlay;
```

Also, get a handle to the map view defined in the XML and add a MyLocationOverlay. After that call the invalidate() method.

Example 14-20.

```
mapView = (MapView)findViewById(R.id.map);
myLocationOverlay = new MyLocationOverlay(this, mapView);
mapView.getOverlays().add(myLocationOverlay);
mapView.invalidate();
```

To prevent depletion of battery, in the onPause method of the class, disableMyLocation() method should be called.

Example 14-21.

```
@Override
protected void onPause() {
    super.onPause();
    myLocationOverlay.disableMyLocation();
}

@Override
protected void onResume() {
    super.onResume();
    myLocationOverlay.enableMyLocation();
}
```

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LZGU1ZmIzYjUtZTY3OS00MjczLWIxNDAtNzY4NjI5ZWJmMzZj&hl=en_US&authkey=CNb-xe8C

14.10 Draw a location marker on a Google MapView

Johan Pelgrim

Problem

You have a geo location and you want to display it on a Google MapView view object

Solution

Create an instance of `Overlay`, draw your marker in it and add it to the `MapView` overlays. Animate to the given geo point.

Discussion

Create a new project called "Location on Map" and use the "Getting Ready for Google Maps Development" recipe to set it up correctly (or simply use the MapTest code from that recipe). If all's well you should have an onCreate like this:

Example 14-22.

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

    MapView mapView = (MapView) findViewById(R.id.mapview);
    mapView.setBuiltInZoomControls(true);
}
```

We are going to make this app a little bit more interesting. First we are going to set the view-type to *satellite* so we are shown some more recognizable terrain information.

Example 14-23.

```
mapView.setSatellite(true);
```

Run your application to see the effect.

You can add traffic information by calling `setTraffic` but that works best with map information, not terrain information.

Ok, we can drag- and zoom around on this map, but let's automatically animate to a certain geo location. First of all create a private field called `geoPoint` and set it to some geo location. Note that the `GeoPoint` constructor takes integer arguments for the latitude and longitude values and not floating points! You can convert a floating point latitude longitude pair by multiplying it by 1 million or `1E6` in Java terms.

Example 14-24.

```
GeoPoint geoPoint = new GeoPoint( (int) (52.334822 * 1E6), (int) (4.668907 * 1E6));
```

We need a handle to the `MapView's MapController` to set the zoom-level and animate to a given `GeoPoint`

Example 14-25.

```
MapController mc = mapView.getController();
mc.setZoom(18);
mc.animateTo(geoPoint);
```

Pretty easy. Fire up the application to see what we've done here. Play around with the zoom-level. What is the minimal value you can set? What is the maximum value?

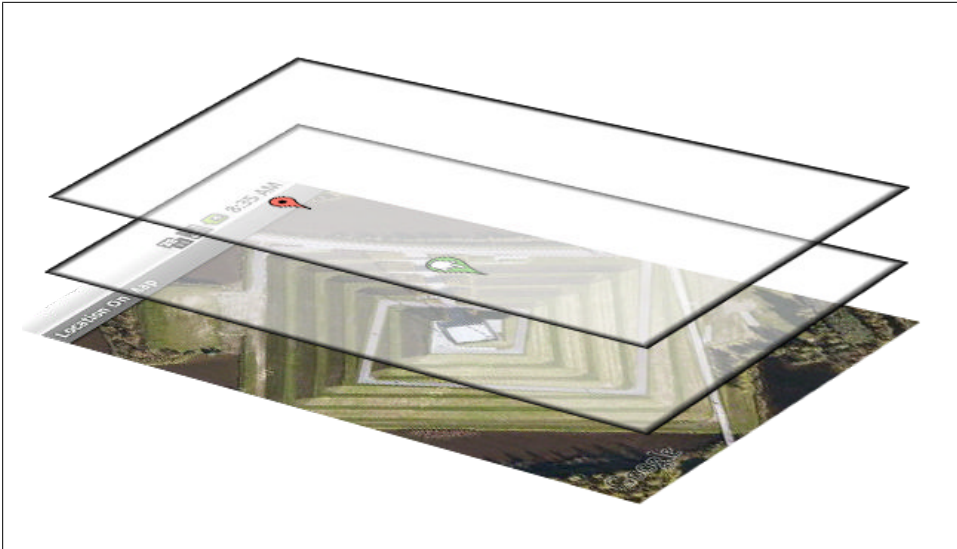


Figure 14-4.

The technique used to display way markers, your current location and other points of interest, on a map is done with *overlays*. You can think of an overlay as you've probably seen them in the old days, used in combination with an overhead projector. Overlays can be seen as those transparent plastic sheets, which sometimes had graphics or text on them. You can layer several overlays on a single `MapView`.

Create a private inner class which extends `Overlay` and override the `draw` method.

Example 14-26.

```
private class MyOverlay extends com.google.android.maps.Overlay {  
  
    @Override  
    public void draw(Canvas canvas, MapView mapView, boolean shadow) { // 1  
        super.draw(canvas, mapView, shadow);  
  
        if (!shadow) { // 2  
  
            Point point = new Point();  
            mapView.getProjection().toPixels(geoPoint, point); // 3  
  
            Bitmap bmp = BitmapFactory.decodeResource(getResources(), R.drawable.marker_default); // 4  
  
            int x = point.x - bmp.getWidth() / 2; // 5  
  
            int y = point.y - bmp.getHeight(); // 6  
  
            canvas.drawBitmap(bmp, x, y, null); // 7  
  
        }  
    }  
}
```

```
}  
}
```

A couple of things are done here.

1. The draw method has a couple of arguments. The first argument is a handle to an instance of `Canvas` which we will use to draw our marker on. The second is an instance of `MapView` on which this overlay is displayed. The third argument is a boolean which indicates whether we are drawing the actual image, or the shadow. In fact, this method is called twice. Once to draw the shadow and once to draw the actual thing you want to draw.
2. We don't want to draw a shadow
3. We translate the geo point to actual pixels and store this information in the `point` variable.
4. We use the resource identifier to decode it to an actual instance of `Bitmap` so we can draw it on the canvas
5. We calculate the x-coordinate of where to draw the marker. We shift it to the left so the *center* of the image is aligned with the x-coordinate of the geo point
6. We calculate the y-coordinate of where to draw the marker. We shift it upward so the *bottom* of the image is aligned with the y-coordinate of the geo point
7. We draw the bitmap at the calculated x and y locations.

You can use this image as the `marker_default.png`. Drop it in your `./res/drawable` directory

You can manipulate the overlays by calling `getOverlays()` on the `MapView` instance.

Example 14-27.

```
List<Overlay> overlays = mapView.getOverlays();  
overlays.clear();  
overlays.add(new MyOverlay());  
  
mapView.invalidate();
```

To force a view to draw, call the `invalidate()` method, which is implemented in the `View` class.

That's it. Fire it up and you should see something like this!

See Also

Recipe [Recipe 14.6](#)



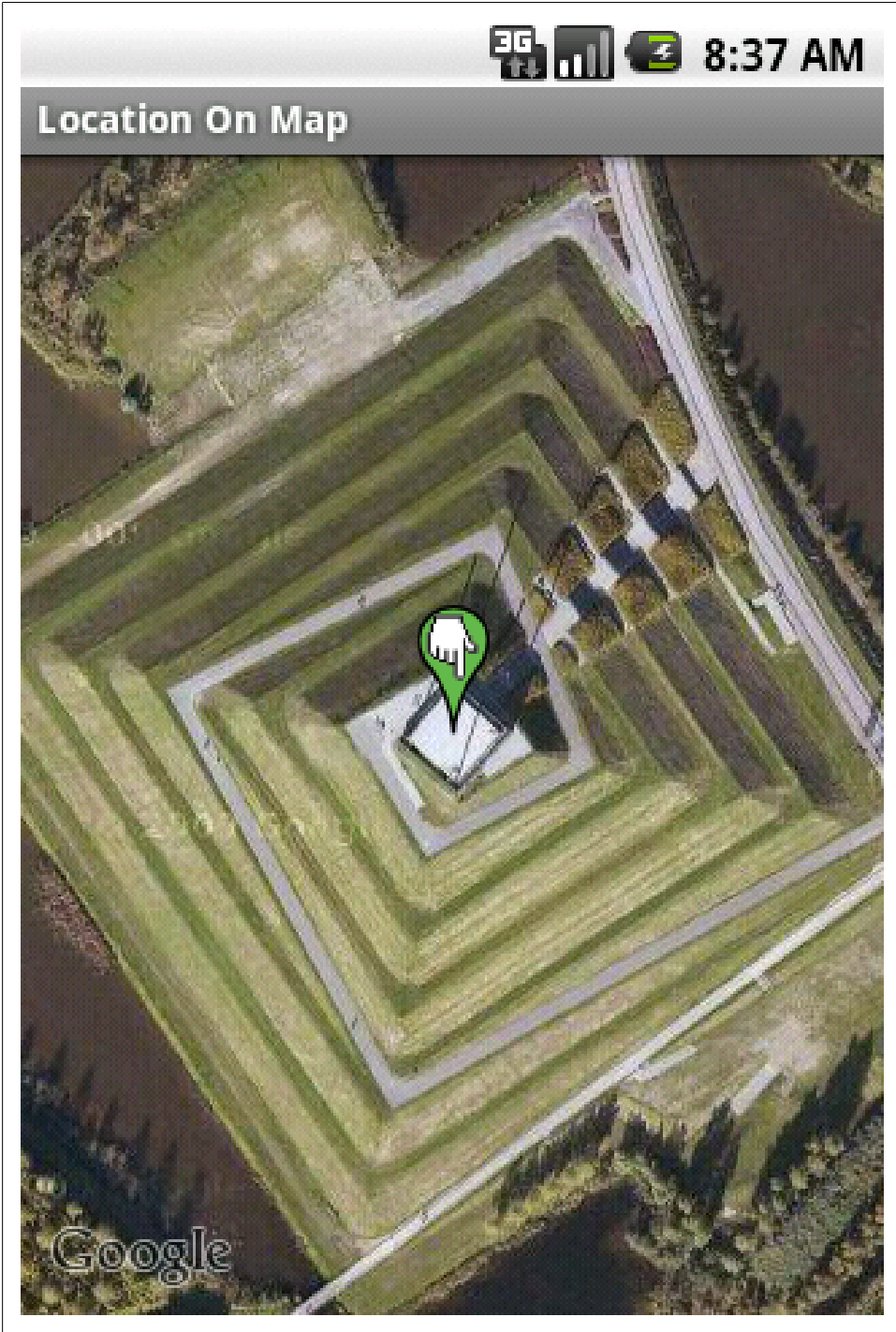


Figure 14-6.

Source Download URL

The source code for this example may be downloaded from this URL: <https://github.com/downloads/jpelgrim/androidcookbook/LocationOnMap.zip>

14.11 Drawing multiple location markers on a MapView

Johan Pelgrim

Problem

You have several GeoPoints which you want to display on a Google MapView.

Solution

Implement the `ItemizedOverlay` *abstract* class and add various `OverlayItems` to it.

Discussion

Introduction

If you want to draw multiple location markers in your `MapView` you can of course take the approach of implementing the `Overlay` interface and do all the resource gathering and drawing in an overridden `draw()` method, as was done in the [Recipe 14.10](#) recipe. This can become cumbersome and hard to maintain. If you want to do *core* drawing of lines and shapes you cannot avoid overriding the `draw()` method, but when it comes down to drawing several simple location markers and handling user clicks on those marker (to name something) the Google Maps API has introduced the `ItemizedOverlay`. This *abstract* class is meant to maintain a list of `Overlay` items and display it as an aggregated `Overlay` on the `MapView`. `ItemizedOverlay` itself implements the `Overlay` interface. Besides that it implements sorting *north-to-south* for drawing, creating span bounds, drawing a marker for each point, and maintaining a focused item. It also matches screen-taps to items, and dispatches focus-change events to an optional listener. This looks like the right candidate to display a couple of location markers on our `MapView`.

Adding the `ItemizedOverlay` to your `MapView`

Let's begin with the skeleton Google Maps project which is described in [Recipe 14.6](#), or create your own and check this recipe's check-list at the end of the recipe to make sure you are good-to-go.

Add an inner class to your `MapActivity` which extends `ItemizedOverlay` and implement the abstract methods and the default constructor. The `ItemizedOverlay` uses your implementations of the `createItem` and `size()` methods to get hold of all the overlay items in your implementation and do the aggregation.

Example 14-28.

```
private class MyItemizedOverlay extends ItemizedOverlay<OverlayItem> {  
  
    public MyItemizedOverlay(Drawable defaultMarker) {  
        super(defaultMarker);  
    }  
  
    @Override  
    protected OverlayItem createItem(int i) {  
        return null;  
    }  
  
    @Override  
    public int size() {  
        return 0;  
    }  
}
```

The `defaultMarker` is a `Drawable` which is drawn on every `OverlayItem` we add to our `ItemizedOverlay`. Whenever you add a `Drawable` to an `OverlayItem` you must set its bounding rectangle via the `setBounds` method. Or you can use one of the two convenience methods `boundCenterBottom` or `boundCenter` which sets the bounding rect to the center-bottom respectively the center of the `Drawable`. Note: a call to `boundCenterBottom` basically results in this call to `setBounds` (given `marker` is an instance of `Drawable`: `marker.setBounds(-marker.getIntrinsicWidth()/2, -marker.getIntrinsicHeight(), marker.getIntrinsicWidth()/2, 0)`). Typically the constructor is rewritten like this:

Example 14-29.

```
public MyItemizedOverlay(Drawable defaultMarker) {  
    super(boundCenterBottom(defaultMarker));  
}
```

We want to add several `OverlayItem` instances so we add a `List` to this inner type and modify the `createItem(int i)` and `size()` methods to use our new list.

Example 14-30.

```
private List<OverlayItem> mOverlays = new ArrayList<OverlayItem>();  
  
@Override  
protected OverlayItem createItem(int i) {  
    return mOverlays.get(i);  
}  
  
@Override  
public int size() {  
    return mOverlays.size();  
}
```

So far so good. Now we add a convenience method to add `OverlayItems` to our internal list.

Example 14-31.

```
public void addOverlayItem(OverlayItem overlayItem) {
    mOverlays.add(overlayItem);
    populate();
}
```

The `populate()` method is a utility method which perform all processing on a new `ItemizedOverlay`. We provide `Items` through the `createItem(int)` method. Rule of thumb is to call this as soon as we have data in our `ItemizedOverlay`, before anything else gets called.

We're basically done with our inner class. Let's add some statements to our `onCreate` method of the surrounding `MapActivity` to add some `OverlayItems` to our implementation of `ItemizedOverlay`

Using `MyItemizedOverlay` in `onCreate`

Let's expand our `onCreate` method and create an instance of our `MyItemizedOverlay` inner type.

Example 14-32.

```
Drawable makerDefault = this.getResources().getDrawable(R.drawable.marker_default);
MyItemizedOverlay itemizedOverlay = new MyItemizedOverlay(makerDefault);
```

Now let's add some overlay items. When creating an `OverlayItem` we must provide three things to the constructor. A `GeoPoint` and two `Strings`, one for the title and one for an additional snippet of text. Let's add an `OverlayItem` for the city of Amsterdam.

Example 14-33.

```
GeoPoint point = new GeoPoint(52372991, 4892655);
OverlayItem overlayItem = new OverlayItem(point, "Amsterdam", null);
itemizedOverlay.addOverlayItem(overlayItem);
```

Let's add another convenience method to our `MyItemizedOverlay` inner type which basically takes two `int` values for *latitude* and *longitude* and a `String` for a title.

Example 14-34.

```
public void addOverlayItem(int lat, int lon, String title) {
    GeoPoint point = new GeoPoint(lat, lon);
    OverlayItem overlayItem = new OverlayItem(point, title, null);
    addOverlayItem(overlayItem);
}
```

We can now rewrite our addition of the *Amsterdam* `OverlayItem` and add two more, one for London and one for Paris.

Example 14-35.

```
itemizedOverlay.addOverlayItem(52372991, 4892655, "Amsterdam");
itemizedOverlay.addOverlayItem(51501851, -140623, "London");
itemizedOverlay.addOverlayItem(48857522, 2294496, "Paris");
```

The next step is to add our itemized overlay to the `MapView` overlays. We get a handle to the list over overlays with a call to `getOverlays()`.

Example 14-36.

```
mapView.getOverlays().add(itemizedOverlay);
```

Finally we manipulate the `MapView` `MapController` to show the right area and zoom-level on our `MapView`. We set the center to a `GeoPoint` of Dunkerque, which appears to be a nice center. There is no `getCenter()` convenience method in the `ItemizedOverlay` class, but this is something you can easily implement yourself if you want to. We can set the zoom-level to a fixed level, but the `ItemizedOverlay` class does have some nice methods to calculate the span which covers all its overlay items. We use this to call `zoomToSpan` on the `MapController` instance.

Example 14-37.

```
MapController mc = mapView.getController();
mc.setCenter(new GeoPoint(51035349, 2370987)); // Dunkerque, Belgium
mc.zoomToSpan(itemizedOverlay.getLatSpanE6(), itemizedOverlay.getLonSpanE6());
```

We're done! When you fire up your app you should see something like this.

(BROKEN XREF TO RECIPE -1 'image:multiple-locations-on-map-1.png')

Extra exercise: Draw an alternate marker Search Google for some nice 100 by 100 pixel markers and place them in your `./res/drawable` directory. Add these drawables as an extra argument to your `addOverlayItem` convenience method. When you create your `OverlayItem` instance use the `setMarker(Drawable drawable)` method to assign a different marker drawable. Remember to set the bounds by calling the `boundCenterBottom` or `boundCenter` convenience methods or do the math yourself and call `setBounds`. Good luck! (The accompanying source code has the solution if these hints are not sufficient).

Do something when the user clicks your marker Finally the `ItemizedOverlay` class has some nice features to handle taps and focus changes on your overlay items. In this final section we will implement the `onTap(int index)` method to show a `Toast` message which displays our overlay item's title. Of course you can do whatever you want when a user taps your marker, show a dialog or another activity, draw a view on the map with `addView`, etc. As you will see this could not be simpler!

```
@Override protected boolean onTap(int index) { Toast.makeText(MainActivity.this,
getItem(index).getTitle(), Toast.LENGTH_LONG).show(); return true; }
```

We return `true` to indicate we have handled the tap-event. If we return `false` the `onTap` is executed for all the overlay items in our `ItemizedOverlay`

Again, when taking your app for a spin, you should see something like this when you tap near your Paris location marker.

(BROKEN XREF TO RECIPE -1 'image:multiple-locations-on-map-2.png')

See Also

[Recipe 14.6](#)

Source Download URL

The source code for this example may be downloaded from this URL: <https://github.com/downloads/jpelgrim/androidcookbook/MultipleLocationsOnMap.zip>

14.12 Creating Overlays for a Google MapView

Rachee Singh

Problem

Demarcating a point on a Google map using an image.

Solution

Use the concept of Map Overlays.

Discussion

Creating one's own map overlay is a 2 step process:

1. Extend the `Overlay` class and implement the required functionality (the type and characteristics of the overlay) in that class.
2. Another class which controls that Google map on the screen then instantiates the class that extends `Overlay`.

Example 14-38.

```
public class AddressOverlay extends Overlay
```

Constructor Initialization in the `AddressOverlay` class:

Example 14-39.

```
public AddressOverlay(Context context, Address address, int drawable) {  
    super();  
    this.context=context;  
    this.drawable=drawable;  
}
```

```

        assert(null != address);
        this.setAddress(address);
        Double convertedLongitude = address.getLongitude() * 1E6;
        Double convertedLatitude = address.getLatitude() * 1E6;

        setGeopoint(new GeoPoint(
            convertedLatitude.intValue(),
            convertedLongitude.intValue()));
    }

```

The draw() method of the Overlay class has to be overridden.

Example 14-40.

```

@Override
public boolean draw(Canvas canvas, MapView mapView, boolean shadow, long when) {
    super.draw(canvas, mapView, shadow);
    Point locationPoint = new Point();
    Projection projection = mapView.getProjection();
    projection.toPixels(getGeopoint(), locationPoint);

    // Reading the image
    Bitmap markerImage = BitmapFactory.decodeResource(context.getResources(), drawable);

    // Drawing the image, keeping the center of the image at the address's location
    canvas.drawBitmap(markerImage, locationPoint.x - markerImage.getWidth() / 2, locationPoint.y - markerImage.getHeight() / 2, null);
    return true;
}

```

In the java class that is implementing the map View's function, the following lines of code are added to add an Overlay on the map:

Example 14-41.

```

List<Overlay> mapOverlays = mapView.getOverlays();
//Instantiating the AddressOverlay class we just defined
//'androidmarker' is the name of the image that you wish to place on the map
AddressOverlay addressOverlay = new AddressOverlay(this, address, R.drawable.androidmarker);
//adding the overlay to the map
mapOverlays.add(addressOverlay);
mapView.invalidate();

```

14.13 Changing Views of a MapView.

Rachee Singh

Problem

Showing an appropriate view of a MapView based on the context in the application.

Solution

Changing the View of a map from the default map mode to satellite or street mode.

Discussion

If the application needs to display distance information between 2 locations on the map, then keeping the map in Street mode is more suitable. Similarly, some applications might need to use the Satellite view of Google maps. This can be done programmatically by:

Example 14-42.

```
//For street view  
mapView.setStreetView(true);
```

```
//For satellite view  
mapView.setSatellite(true);
```

14.14 Draw overlay icon without using Drawable

Keith Mendoza

Problem

How can you display an map overlay in MapView without using Drawable objects?

Solution

Override the `ItemizedOverlay::draw()` function.

Discussion

This assumes that you have at least done the "Hello, MapView" tutorial, so I will not cover what abstract functions that you need to implement from `ItemizedOverlay`. The complete source code for `Nearby Metars 01.01.0.2` is available for download so some the complete code for the classes mentioned will not be shown in full.

Overview

`Nearby Metars` displays the cloud condition icon and the direction part of a wind barb as an overlay on a `MapView`. This icon is drawn in a way where the cloud condition covers the scale equivalent of about 1 mile around the airport. For anyone curious here is the description of METAR taken from the [METARs help page](#) provided by NOAA's Aviation Weather Services:

Weather stations all over the world report weather conditions every hour using a data format referred to as METAR (this is a French acronym with a loose English translation to "routine aviation weather observation"). These data are collected centrally by the U.S. National Weather Service (and other country's equivalents) and distributed.

[Page 4](#) of the help page shows the cloud coverage icons. These are the icons that needs to be drawn as an overlay over the airport to depict the cloud coverage. The wind barb points the wind direction (it's actually the direction the wind is coming from).

Overriding the `ItemizedOverlay::draw()` function

`ItemizedOverlay::draw()` is called whenever the `MapView` needs to be redrawn for whatever reason. Here is the function signature of the `draw()` function:

Example 14-43.

```
public void draw(android.graphics.Canvas canvas,
                 MapView mapView,
                 boolean shadow)
```

Here are the parameter description taken directly from the API document:

- `canvas` - The `Canvas` upon which to draw. Note that this may already have a transformation applied, so be sure to leave it the way you found it.
- `mapView` - the `MapView` that requested the draw. Use `MapView.getProjection()` to convert between on-screen pixels and latitude/longitude pairs.
- `shadow` - If true, draw the shadow layer. If false, draw the overlay contents.

For each time that the screen is being redrawn the `draw()` function will be called twice: Once when `shadow` is true, and again when `shadow` is false. For `Nearby Metars` there is no need to draw shadows in overlay items.

For `Nearby Metars`, `MetarList` is the `MetarItem` specific (note to editor: not sure if this is the correct term) implementation of `ItemizedOverlay`. This class overrides the abstract functions, and the `draw()` function. This is the code for `MetarList::draw()`:

Example 14-44.

```
public void draw(android.graphics.Canvas canvas, MapView mapView, boolean shadow) {
    if(!shadow) {
        Log.v("NearbyMetars", "Drawing items");
        MetarItem item;
        for(int i=0; i<mOverlays.size(); i++) {
            item = mOverlays.get(i);
            item.draw(canvas, mapView);
        }
    }
}
```

`mOverlays` is an instance of `ArrayList<MetarItem>`. Whenever `draw()` is called, we iterate through `mOverlays` and call `MetarItem::draw()`. This implementation makes `MetarList` and `MetarItem` tightly coupled for the sake of performance.

Overview of MetarItem class

This class is a subclass of `OverlayItem`. The `mTitle` and `mSnippet` fields inherited from `OverlayItem` are used for the ICAO code and the raw metar string respectively. There are two fields added in `MetarItem`:

- `skyCond` - This is an instance of the `SkyConds` enumerated type defined inside `MetarItem`
- `windDir` - This is a float value to store the wind direction

`MetarItem::draw()` function

This is where the real work of drawing the icon onto the canvas really happens. In the METAR charts from ADDS, the cloud condition icons are drawn using the colors to depict the flight category in effect for that airport; however, as of version 01.01.0.2 `Nearby Metars` doesn't depict the flight category so the icons are all black. To make thing short the code is broken into sections and the explanation follows after each code snippet.

Example 14-45.

```
public void draw(Canvas canvas, MapView mapView) {
```

This function takes two parameters: `canvas` and `mapView`. These two parameters have the same types as the first 2 parameters of `ItemizedOverlay::draw()`.

Example 14-46.

```
    //Get the bounds of the icon
    Point point = new Point();
    Projection projection = mapView.getProjection();
    projection.toPixels(mPoint, point);
```

First we convert the lat,long coordinates of the airport to x,y coordinate. `Projection::toPixels()` takes a `GeoPoint` object that stores the lat,long of the location that will be marked by the overlay as the first parameter; and a `Point` instance to store the x,y coordinate of that location in the `MapView` canvas.

Example 14-47.

```
    final float project = (float)((projection.metersToEquatorPixels((float)1609.344) > 10) ? projection.mete
    Log.d("NearbyMetars", "Value of project: " + Float.toString(project));
    final RectF drawPos = new RectF(point.x-project, point.y-project, point.x+project, point.y+project);
```

We then calculate how many pixels 1 mile would be given the map's current zoom level. Then we calculate the bounding coordinates of the icon to be drawn in as a `RectF` instance.

Example 14-48.

```
//Get the paint to use for drawing the icons
Paint paint = new Paint();
paint.setStyle(Paint.Style.STROKE);
paint.setARGB(179, 0, 0, 0);
paint.setStrokeWidth(2.0f);
paint.setStrokeCap(Paint.Cap.BUTT);
```

A *Paint* object is instantiated and set to draw a 2 pixel thick black line at about 70% transparency. The reason to not make the cloud condition icons not drawn completely opaque is to allow the user to be able to read the labels on the map. Remember, the cloud icons are drawn on top of the map in a layered fashion.

Example 14-49.

```
switch(skyCond) {
    case CLR:
        canvas.drawRect(drawPos, paint);
        break;
    case SKC:
        canvas.drawCircle(point.x, point.y, project, paint);
        break;
    case FEW:
        canvas.drawCircle(point.x, point.y, project, paint);
        canvas.drawLine(point.x, drawPos.top, point.x, drawPos.bottom, paint);
        break;
    case SCT:
        canvas.drawArc(drawPos, 0, 270, false, paint);
        paint.setStyle(Paint.Style.FILL_AND_STROKE);
        canvas.drawArc(drawPos, 270, 90, true, paint);
        break;
    case BKN:
        canvas.drawArc(drawPos, 180, 90, false, paint);
        paint.setStyle(Paint.Style.FILL_AND_STROKE);
        canvas.drawArc(drawPos, 270, 270, true, paint);
        break;
    case OVC:
        paint.setStyle(Paint.Style.FILL_AND_STROKE);
        canvas.drawCircle(point.x, point.y, project, paint);
        break;
    case OVX:
        canvas.drawArc(drawPos, 45, 180, true, paint);
        canvas.drawArc(drawPos, 135, 180, true, paint);
        canvas.drawArc(drawPos, 315, 90, true, paint);
        break;
}
```

This section of code renders the cloud condition icons based on the value of `skyCond`. Please see the [Canvas](#) reference for the description of the `draw*()` functions. Drawing the icons for `CLR` and `SKC` are straight forward, call the appropriate `draw*()` function. `FEW` calls a `drawCircle()` to draw the circular outline, and then calls the `drawLine()` to draw the vertical line. In the case of this icon, it won't matter if `drawLine()` was called

first instead of `drawCircle()`. However, it would be good to remember that successive calls to the `draw*()` function over the same area will draw shapes on top of each other.

Conditions like `SKT`, `BKN`, and `OVC` first calls `drawArc()` to draw the unfilled portion of the icon, and then switches the pen style to `FILL_AND_STROKE` then calls `drawArc()` again to complete the circle with the filled portion of the icon. The use of `drawArc()` on these icons are actually an optimization. `Canvas::drawCircle()` actually calls `Canvas::drawArc()` under the hood. Why render a graphic that will simply be covered by another graphic drawn in the same location.

Example 14-50.

```
//Draw the wind bar if wind is NOT variable
if(windDir > 0)
{
    final float barLen = project * 3;

    //This has been modified to go the opposite direction of
    //standard polar to Cartesian plotting
    canvas.drawLine(point.x, point.y, (float)(point.x + barLen * Math.sin(windDir)), (float)(point.y - b
}
}
```

This last portion of code draws the wind barb without the wind speed lines. As the comment states, this function calculates the cartesian coordinate with the angle going in a clockwise direction since that's how compass directions go. The standard mathematic polar coordinates have angles going in a counter-clockwise direction. Another thing to note is that the value of `project` is actually the radian equivalent of the wind compass direction.

Final Thoughts

Using the `Canvas::draw*()` functions is not necessarily the best method for drawing the overlay icons. Android can render [Drawable Resources](#) in a more optimized manner than calling the `Canvas::draw*()` functions; and it's easier to create great looking images using an image editor. If the overlays for Nearby Metars were done using `Drawable Resources`, editing the XML files would be cumbersome; using bitmaps will just be a resource hog. Whether to use `Drawable` or programmatically draw the overlay icon will depend largely on the project's requirements.

See Also

[Hello, MapView Tutorial Canvas class reference ItemizedOverlay class reference OverlayItem class reference Google Add-On API Reference](#)

Source Download URL

The source code for this example may be downloaded from this URL: <https://github.com/keithmendozasr/NearbyMetars/zipball/01.01.0.2>

Binary Download URL

The executable code for this example may be downloaded from this URL: <https://github.com/downloads/keithmendozasr/NearbyMetars/NearbyMetars-01.01.0.2.apk>

14.15 Location search on Google maps

Rachee Singh

Problem

Selecting text from an editText and looking for that location on Google Maps. n finding, making a list of all the results. Displaying the most appropriate location result.

Solution

The text entered into the edit Text by the user is extracted. It is searched and the search results are extracted. The best out of the location search results is displayed as a toast (This is a sample, many other activities can be done using the location search result).

Discussion

This method obtains text from a textView named: addressText. Then this text is searched for using the getFromLocationName() method of the Geocoder class. From the search results obtained the first result is extracted and displayed as a Toast. If the string returned is of size=0, an appropriate message is displayed.

Example 14-51.

```
protected void mapCurrentAddress() {
    String addressString = addressText.getText().toString();
    Geocoder g = new Geocoder(this);
    List<Address> addresses;
    try {
        addresses = g.getFromLocationName(addressString, 1);
        String add = "";
        if (addresses.size() > 0) {

            address = addresses.get(0);
            for (int i=0; i<address.getMaxAddressLineIndex();i++) {
                add += address.getAddressLine(i) + "\n";
            }
            Toast.makeText(getBaseContext(), add, Toast.LENGTH_SHORT).show();

        } else {
            Toast.makeText(getBaseContext(),"We failed to locate this address.", Toast.LENGTH_SHORT).show();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

14.16 MapView inside TabView

Vladimir Kroz

Problem

You want to place a MapView object within TabView.

Target application outline

- Draw two empty tabs
- Show map on the first tab

Solution

1. Create MapView and corresponding XML layout, make sure it runs standalone
2. Create TabView and corresponding XML layout
3. Attach MapView activity to one of the tabs using TabSpec.setContent() That's it!

Discussion

Setup - obtain maps api key

First step - get from google api key to enable your MapView. Just follow google Maps Add-On documentation and eventually you should get something like that: Your key is: 01234567890abvgd9876543eprst333223 (just example) This key is good for all apps signed with your certificate whose fingerprint is: 00:00:11:AA:BB:CC:AB:01:02:22:33:44:55:FF:EE:DD Check out the API documentation for more information.

Layouts

TabLayout (main.xml). The structure of typical TabLayout includes TabHost as a container, TabWidget to draw tabs and FrameLayout with predefined id "@android:id/tabcontent" to contain interchangeable content.

Example 14-52.

```
<?xml version="1.0" encoding="utf-8"?>
<TabHost xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/tabhost"
    android:layout_width="fill_parent" android:layout_height="fill_parent">
    <LinearLayout android:orientation="vertical"
        android:layout_width="fill_parent" android:layout_height="fill_parent">
        <TabWidget android:id="@android:id/tabs"
            android:layout_width="fill_parent" android:layout_height="wrap_content"/>
        <FrameLayout android:id="@android:id/tabcontent"
            android:layout_width="fill_parent" android:layout_height="fill_parent">
            <RelativeLayout android:id="@+id/emptylayout1" android:orientation="vertical"
                android:layout_width="fill_parent" android:layout_height="fill_parent"/>
        </FrameLayout>
    </LinearLayout>
</TabHost>
```



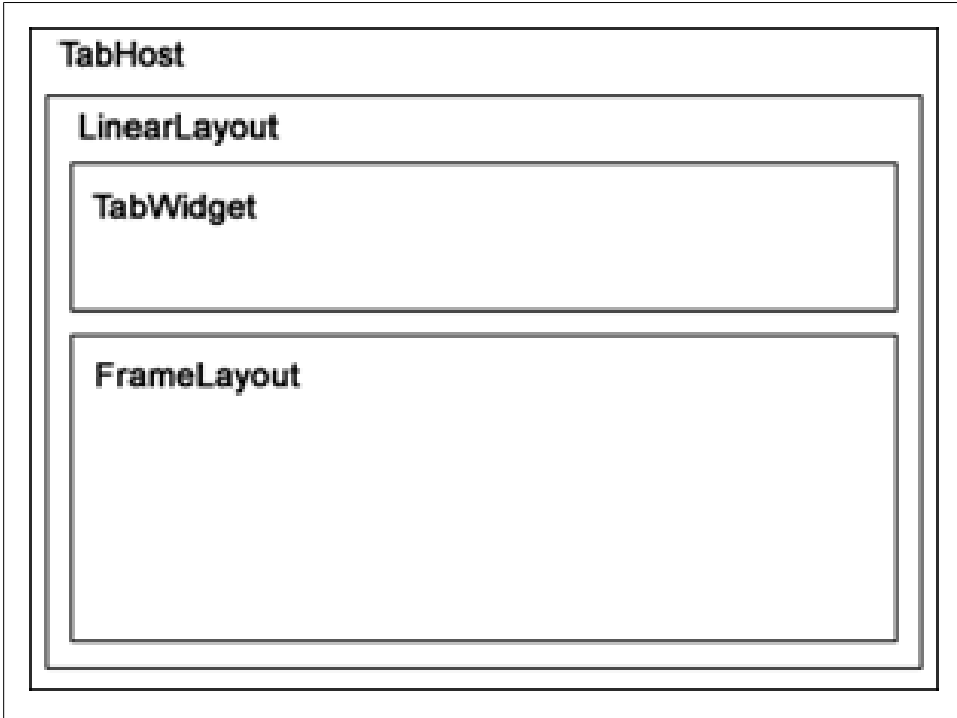


Figure 14-8.

```
<TextView android:id="@+id/textview2"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:text="Details Details Details Details"/>
</FrameLayout>
</LinearLayout>
</TabHost>
```

Layout for MapView (Maptabview.xml). Code for the MapView layout follows.

Example 14-53.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/maptablayout" android:orientation="vertical"
android:layout_width="fill_parent" android:layout_height="fill_parent">
<com.google.android.maps.MapView android:id="@+id/mapview"
android:layout_width="fill_parent" android:layout_height="fill_parent"
android:clickable="true"
android:apiKey="OpFtdSwta8EMTfArj32yc0w2kZg0LSEqa4fUGFA"/>
</RelativeLayout>
```

Application code

AppMain.java - Entry point. This would be application entry point; later we'll enhance this code with more details

Example 14-54.

```
package org.kroztech.cookbook;

import android.app.TabActivity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.widget.FrameLayout;
import android.widget.TabHost;
import android.widget.TabHost.TabSpec;

public class AppMain extends TabActivity {
    TabHost mTabHost;
    FrameLayout mFrameLayout;

    /** Called when the activity is first created.*/
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mTabHost = getTabHost();
        TabSpec tabSpec = mTabHost.newTabSpec("tab_test1");
        tabSpec.setIndicator("Map");
        Context ctx = this.getApplicationContext();
        Intent i = new Intent(ctx, MapTabView.class);
        tabSpec.setContent(i);
        mTabHost.addTab(tabSpec);
        mTabHost.addTab(mTabHost.newTabSpec("tab_test2").setIndicator("Details").setContent(R.id.textview2));
        mTabHost.setCurrentTab(0);
    }
}
```

MapTabView.java - Map Activity. The Map Activity follows.

Example 14-55.

```
package org.kroztech.cookbook;

import android.os.Bundle;
import com.google.android.maps.MapActivity;

public class MapTabView extends MapActivity {
    @Override
    protected void onCreate(Bundle icle) {
        super.onCreate(icle);
        setContentView(R.layout.maptabview);
    }
    @Override
    protected boolean isRouteDisplayed() {
```

```

    return false;
}
}

```

Manifest (AndroidManifest.xml). And the manifest file.

Example 14-56.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.kroz.tag" android:versionCode="1" android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <uses-library android:name="com.google.android.maps"/>
        <activity android:name=".AppMain" android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:name="MapView" android:label="@string/mapview_name">
            <intent-filter>
                <category android:name="android.intent.category.EMBED"></category>
                <action android:name="android.intent.action.MAIN"></action>
            </intent-filter>
        </activity>
    </application>
    <uses-sdk android:minSdkVersion="3"/>
    <uses-permission android:name="android.permission.INTERNET"></uses-permission>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"></uses-permission>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"></uses-permission>
    <uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS"></uses-permission>
</manifest>

```

Source Download URL

The source code for this example may be downloaded from this URL: http://www.kroztech.com/res/android_cookbook/src/MapViewDemo.zip

14.17 Handling longpress in a map

Roger Kind Kristiansen

Problem

Some map applications might want to trigger an action related to an arbitrary point on the map, for example through a context menu. You might think support for this would be built in, but this seems not to be the case.

Solution

Override `MapActivity.dispatchTouchEvent(MotionEvent event)` with code to listen for the longpress.

Discussion

The `dispatchTouchEvent()` method processes all touch events triggered on the `MapView`. When overriding this method, we need to take care of the following elements to handle longpresses:

1. If the finger has just touched the screen, we start checking how long it is held. We need to do this in a separate thread, or else the UI will lock up. If the checks in the following point do not kick in within a given threshold, we have a longpress and call the desired code, In my case this was to show the context menu.
2. If some motion event that can not be part of a longpress has been performed, cancel the check in pt.2. These events are:
3. * Finger has been removed from screen
4. * Finger has moved more than a given threshold since the previous event

Here is the complete `MapActivity`, handling all this:

Example 14-57.

```
import android.os.Bundle;
import android.os.Looper;
import android.view.MotionEvent;

import com.google.android.maps.MapActivity;

public class Map extends MapActivity {
    private boolean isPotentialLongPress;

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

    @Override
    public boolean dispatchTouchEvent(MotionEvent event) {
        handleLongPress(event);
        return super.dispatchTouchEvent(event);
    }

    private void handleLongPress(MotionEvent event) {
        if (event.getAction() == MotionEvent.ACTION_DOWN) {
            // A new touch has been detected

            new Thread(new Runnable() {
                public void run() {
                    Looper.prepare();
                    if (isLongPressDetected()) {
                        // We have a longpress! Perform your action here
                    }
                }
            }).start();
        }
    }
}
```

```

    } else if (event.getAction() == MotionEvent.ACTION_MOVE) {
        /*
         * Only MotionEvent.ACTION_MOVE could potentially be regarded as
         * part of a longpress, as this event is triggered by the finger
         * moving slightly on the device screen. Any other events causes us
         * to cancel this events status as a potential longpress.
         */
        if (event.getHistorySize() < 1)
            return; // First call, no history

        // Get difference in position since previous move event
        float diffX = event.getX()
            - event.getHistoricalX(event.getHistorySize() - 1);
        float diffY = event.getY()
            - event.getHistoricalY(event.getHistorySize() - 1);

        /* If position has moved substantially, this is not a long press but
         probably a drag action */
        if (Math.abs(diffX) > 0.5f || Math.abs(diffY) > 0.5f) {
            isPotentialLongPress = false;
        }
    } else {
        // This motion is something else, and thus not part of a longpress
        isPotentialLongPress = false;
    }
}

/**
 * Loops for an amount of time while checking if the state of the
 * isPotentialLongPress variable has changed. If it has, this is regarded as
 * if the longpress has been canceled. Else it is regarded as a longpress.
 */
public boolean isLongPressDetected() {
    isPotentialLongPress = true;
    try {
        for (int i = 0; i < (50); i++) {
            Thread.sleep(10);
            if (!isPotentialLongPress) {
                return false;
            }
        }
        return true;
    } catch (InterruptedException e) {
        return false;
    } finally {
        isPotentialLongPress = false;
    }
}

@Override
protected boolean isRouteDisplayed() {
    return false;
}
}

```

This will trigger your action if the user keeps a finger pressed on the map for more than half a second. Modify the integer values in `isLongPressDetected()` to change this.

Now, open your context menu or perform any other action you can think of.

14.18 Using OpenStreetMap

Rachee Singh

Problem

You want to use OpenStreetMap (OSM) map data in your application in place of the standard Google Maps.

Solution

Use the third-party `osmdroid` library to interact with OpenStreetMap-Data

Discussion

OpenStreetMap is a free, editable map of the world. The `OpenStreetMapView` is a (almost) full/free replacement for Android's `MapView` class. See the `osmdroid` [google-code](#) page for more details.

To use OSM in your android app, your project must be Android API level 3 (version 1.5) or higher. You need to include 2 jars in the Android project namely, `osmdroid-android-x.xx.jar` and `slf4j-android-1.5.8.jar`. OSMDroid is a set of tools for OpenStreetMap data; SLF4J is (yet another) simplified logging facade. These can be downloaded from the links below:

- [osmdroid](#) jar file.
- [slf4j](#) jar file.

See [Recipe 1.15](#) to learn how to use external libraries in your Android project.

After adding the JARs to the project we can start coding.

- You need to add a OSM `MapView` to your XML layout like this:

Example 14-58.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <org.osmdroid.views.MapView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/mapview">
    </org.osmdroid.views.MapView>
</LinearLayout>
```

- Also include the INTERNET permission in the AndroidManifest.

Example 14-59.

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

- Now we have to use this MapView in the Activity code. This is done exactly as you would do in the case of Google maps.

Example 14-60.

```
private MapView mapView;  
private MapController mapController;  
mapView = (MapView) this.findViewById(R.id.mapview);  
mapView.setBuiltInZoomControls(true);  
mapView.setMultiTouchControls(true);  
mapController = this.mapView.getController();  
mapController.setZoom(2);
```

Once you are done with the code, go ahead and run it! This is how the application would look:

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LYjIwYTM1NTctZTU3OS00NTE5LTg1NmItZTU4MGRkYTMzODJl&hl=en_US

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LY2U5MzVlMGYtOWY1Ni00NThhLTg0MmItMzI2MDgyYzRjNzI5&hl=en_US

14.19 Creating overlays in OpenStreetMaps

Rachee Singh

Problem

You need to create overlays for your OpenStreetMap

Solution

Instantiate an Overlay class and add the overlay to the point you wish to demarcate on the map.



Figure 14-10. Location and Map Applications

Discussion

To use OpenStreetMaps, see [Recipe 14.18](#). To begin with, we get a handle on to the `MapView`, enable zoom controls on the map using `setBuiltInZoomControls(true)` and set the zoom level to a reasonable value. Then we create two `GeoPoints`, first (`mapCenter`) is to center the OSM around the point when the application starts and the second (`overlayPoint`) is a little away from the `mapCenter` point. The overlay is placed at the `overlayPoint`. To create the overlay item, we need to instantiate the `OverlayItem` class (along with appropriate arguments specifying the overlay description and the point at which the overlay has to be placed). We can add multiple overlays to an `ArrayList` and then add all of them in one go (using the `add` method). After adding the overlay, we also call the `invalidate` method update the `MapView`.

This is the code:

Example 14-61.

```
mapView = (MapView) this.findViewById(R.id.mapview);
mapView.setBuiltInZoomControls(true);
mapController = this.mapView.getController();
mapController.setZoom(12);
GeoPoint mapCenter = new GeoPoint(53554070, -2959520);
GeoPoint overlayPoint = new GeoPoint(53554070 + 1000, -2959520 + 1000);
mapController.setCenter(mapCenter);
ArrayList<OverlayItem> overlays = new ArrayList<OverlayItem>();
overlays.add(new OverlayItem("New Overlay", "Overlay Description", overlayPoint));

resourceProxy = new DefaultResourceProxyImpl(getApplicationContext());
this.myLocationOverlay = new ItemizedIconOverlay<OverlayItem>(overlays, null, resourceProxy);
this.mapView.getOverlays().add(this.myLocationOverlay);
mapView.invalidate();
```

This is how the overlay looks:

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LMTlhYmI3ZjctMGU4ZS00ZDhjLWJjMGMtYWYwMTBmNzcxNzJl&hl=en_US

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LNDIwN2U3OTItYjgwMy00OWIwLTg1ZmUtYzM5ZjZiMTIjNjRk&hl=en_US

14.20 Using a scale on an OpenStreetMap

Rachee Singh

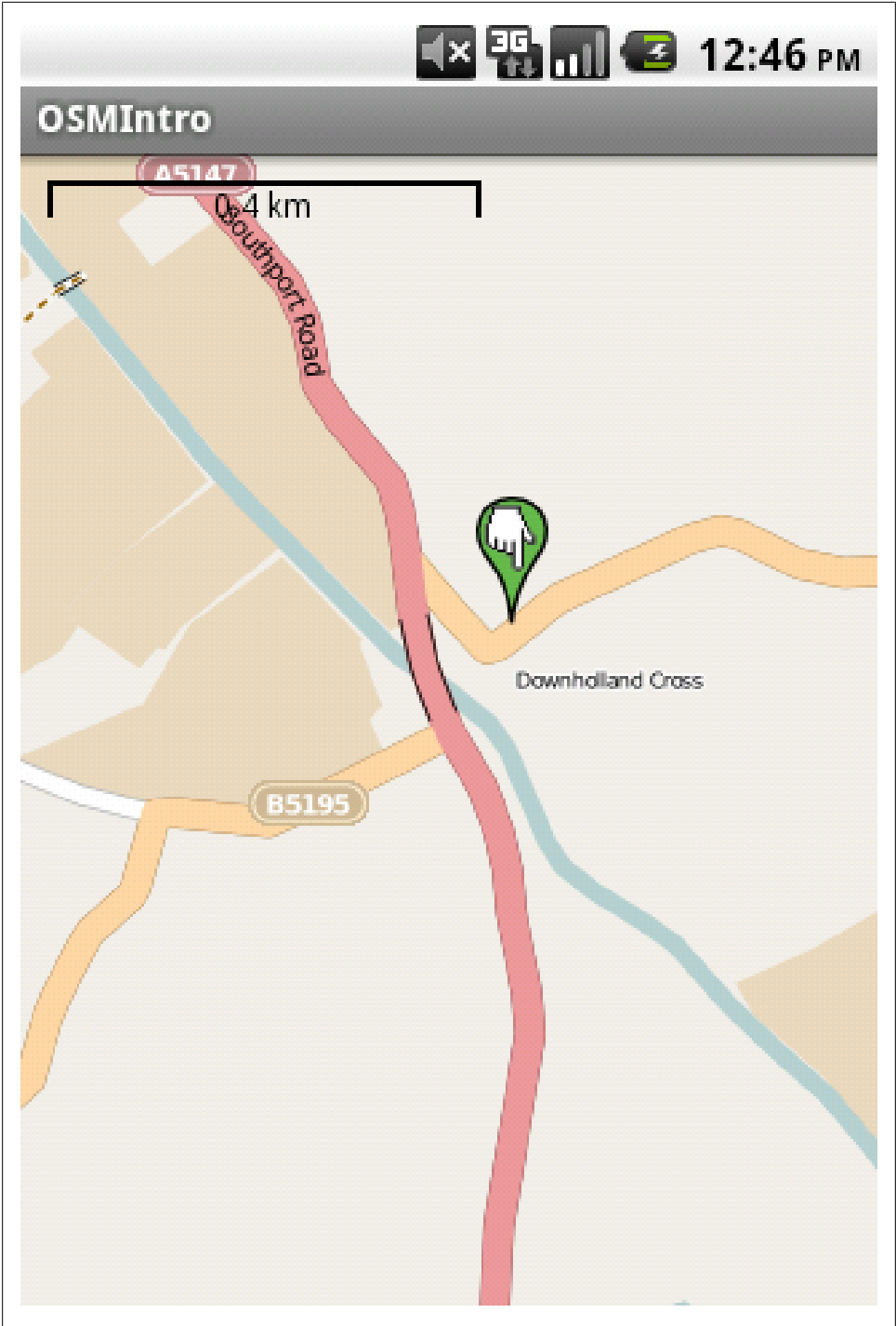


Figure 14-11.

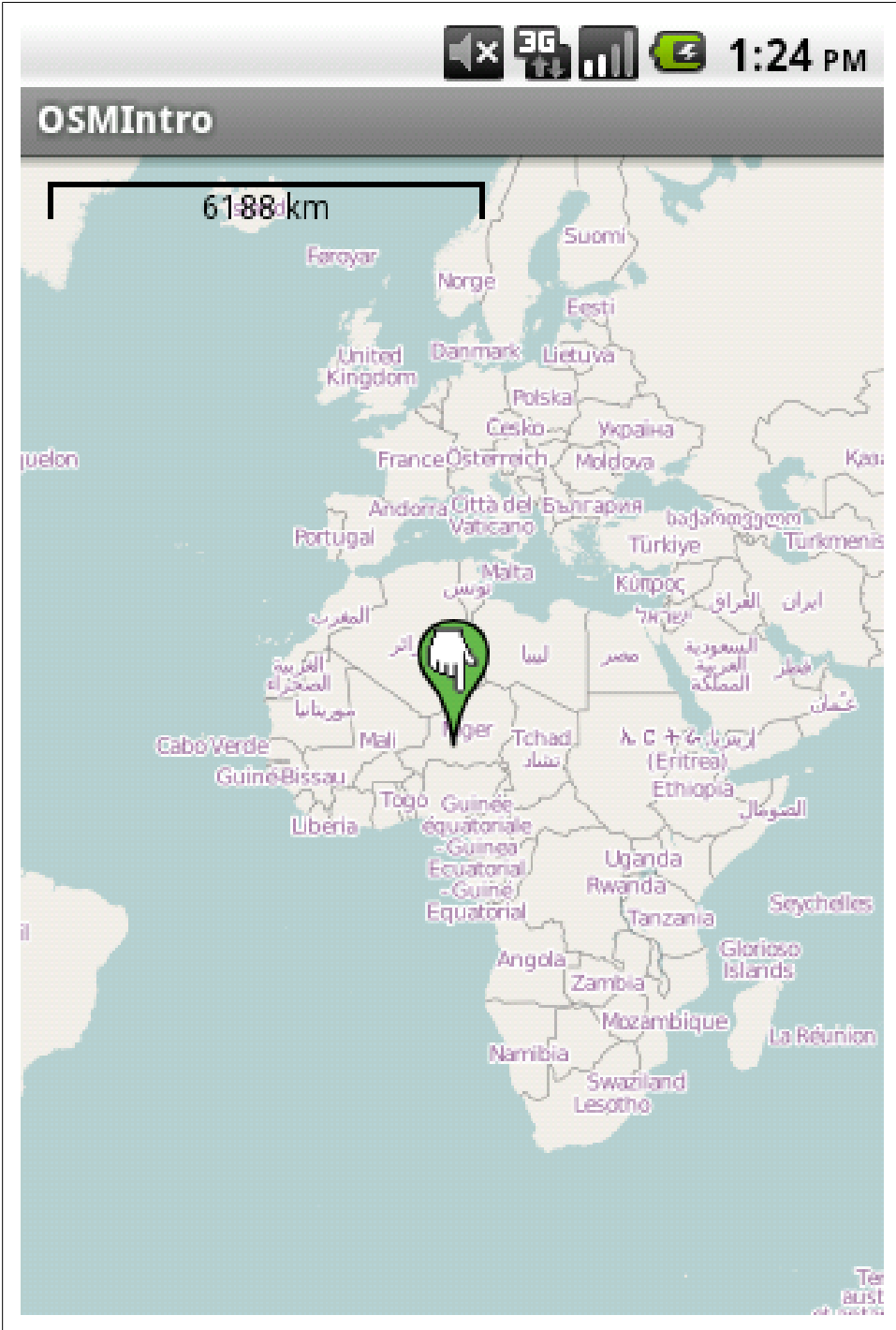


Figure 14-12.

Problem

You need to show a map scale on your OSM to indicate the level of zoom on the MapView

Solution

A scale can be added on the OSM as an overlay using the ScaleBarOverlay class

Discussion

Putting a scale on your MapView helps the user keep track of the zoom level at which he is (also helping him/her of the approximate distances on the map). To overlay a scale on your OSM MapView, instantiate the ScaleBarOverlay and add it to your MapView using the add method.

Here is how the code would look:

Example 14-62.

```
ScaleBarOverlay myScaleBarOverlay = new ScaleBarOverlay(this);  
this.mapView.getOverlays().add(this.myScaleBarOverlay);
```

The scale bar overlay looks like this:

14.21 Handling touch events on an OpenStreetMap Overlay

Rachee Singh

Problem

You need to perform actions when the overlay on an OpenStreetMap is tapped

Solution

Overriding the methods of OnItemGestureListener method for single tap events and long press events.

Discussion

To address touch events on the map overlay, we modify the way we instantiate an overlay item (To see more details about using overlays in OSM, check out: [Recipe 14.19](#)). While instantiating the OverlayItem, we make use of an anonymous object of OnItemGestureListener class as an argument and provide our own implementation of onItemSingleTapUp and onItemLongPress methods. In these methods, we simply display a toast depicting which action took place: single tap or long press and also the title and description of the overlay touched.

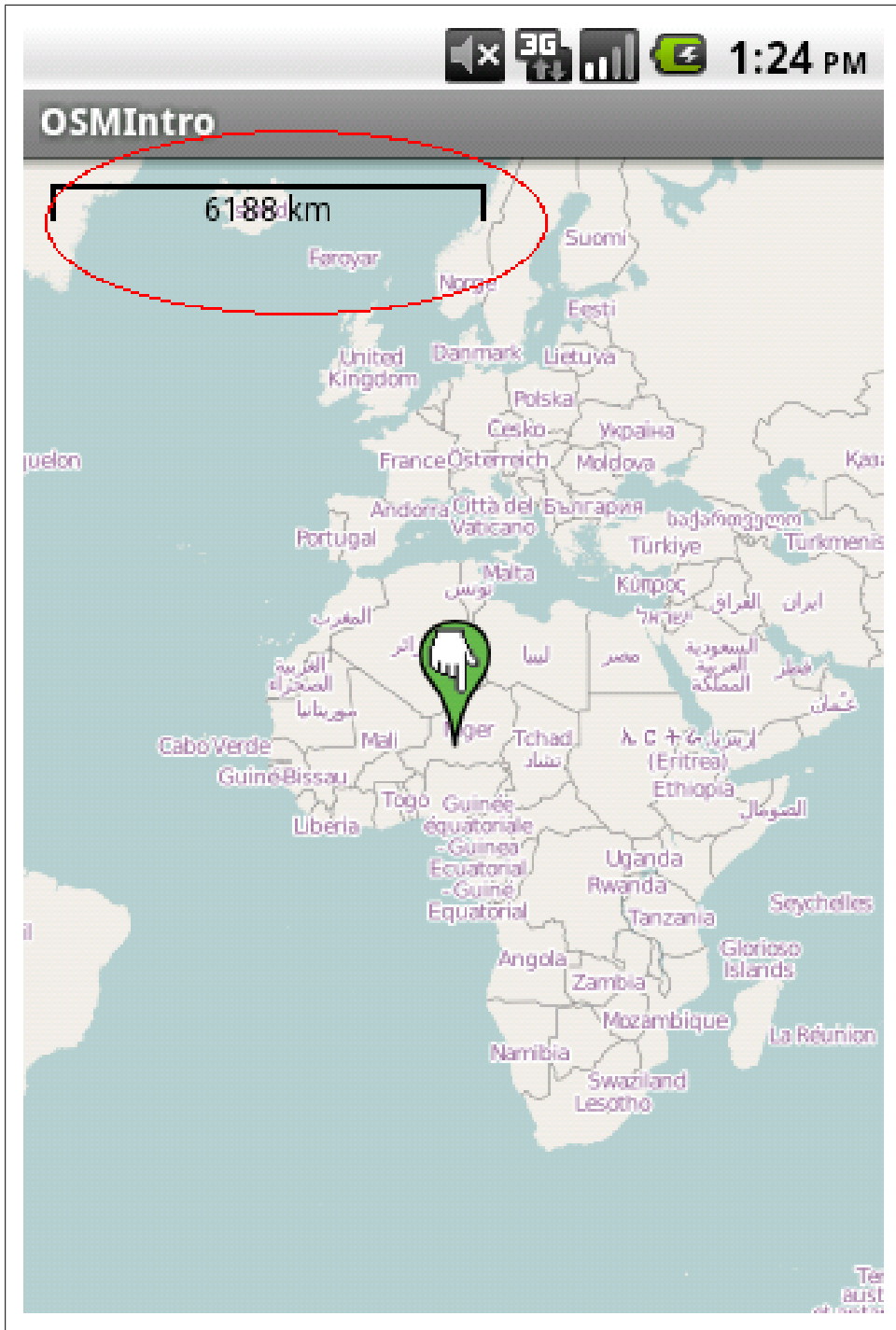


Figure 14-13.

Here is how the code looks:

Example 14-63.

```
ArrayList<OverlayItem> items = new ArrayList<OverlayItem>();
items.add(new OverlayItem("New Overlay", "Overlay Sample Description", overlayPoint));

resourceProxy = new DefaultResourceProxyImpl(getApplicationContext());

this.myLocationOverlay = new ItemizedIconOverlay<OverlayItem>(items,
    new ItemizedIconOverlay.OnItemGestureListener<OverlayItem>() {
        @Override
        public boolean onItemSingleTapUp(final int index, final OverlayItem item) {
            Toast.makeText( getApplicationContext(), "Overlay Titled: " +
item.mTitle + " Single Tapped" + "\n" + "Description: " + item.mDescription, Toast.LENGTH_LONG).show();
            return true;
        }
        @Override
        public boolean onItemLongPress(final int index, final OverlayItem item) {
            Toast.makeText( getApplicationContext(), "Overlay Titled: " +
item.mTitle + " Long pressed" + "\n" + "Description: " + item.mDescription , Toast.LENGTH_LONG).show();
            return false;
        }
    }, resourceProxy);
this.mapView.getOverlays().add(this.myLocationOverlay);
mapView.invalidate();
```

On tapping the overlay once, this is how the application looks:

On long pressing the overlay, this is how the application looks:

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LMzZmMjJkZjYtN2M1OC00MGEzLWI2ZTQtNTUxMzFhZjEzMGIx&hl=en_US

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LZjk2MTAxYWMtMWRiOC00ZGVlThmZTlTmWJhNjU1MGRmYmI4&hl=en_US

14.22 Getting location updates with OpenStreetMaps

Rachee Singh

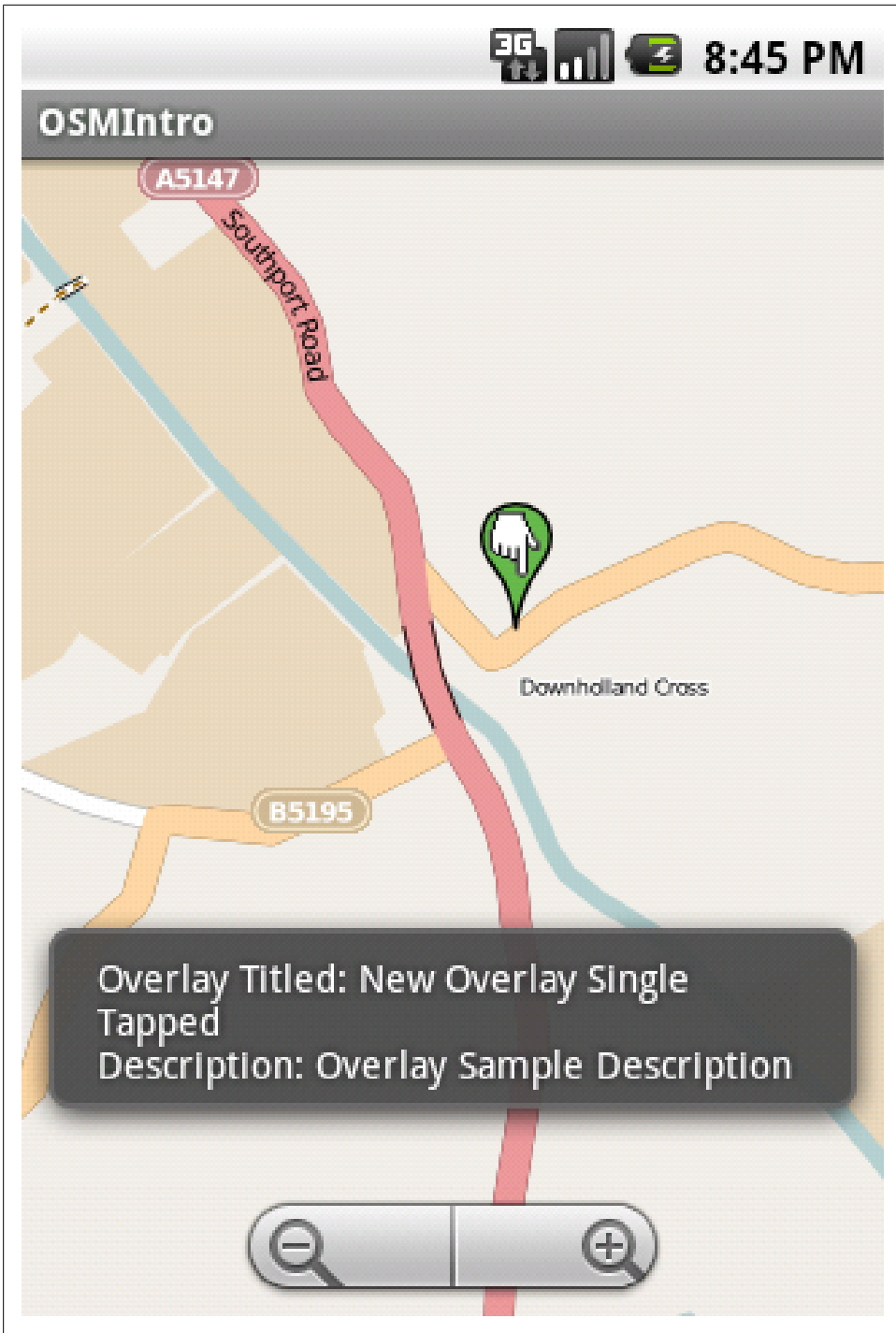


Figure 14-14.

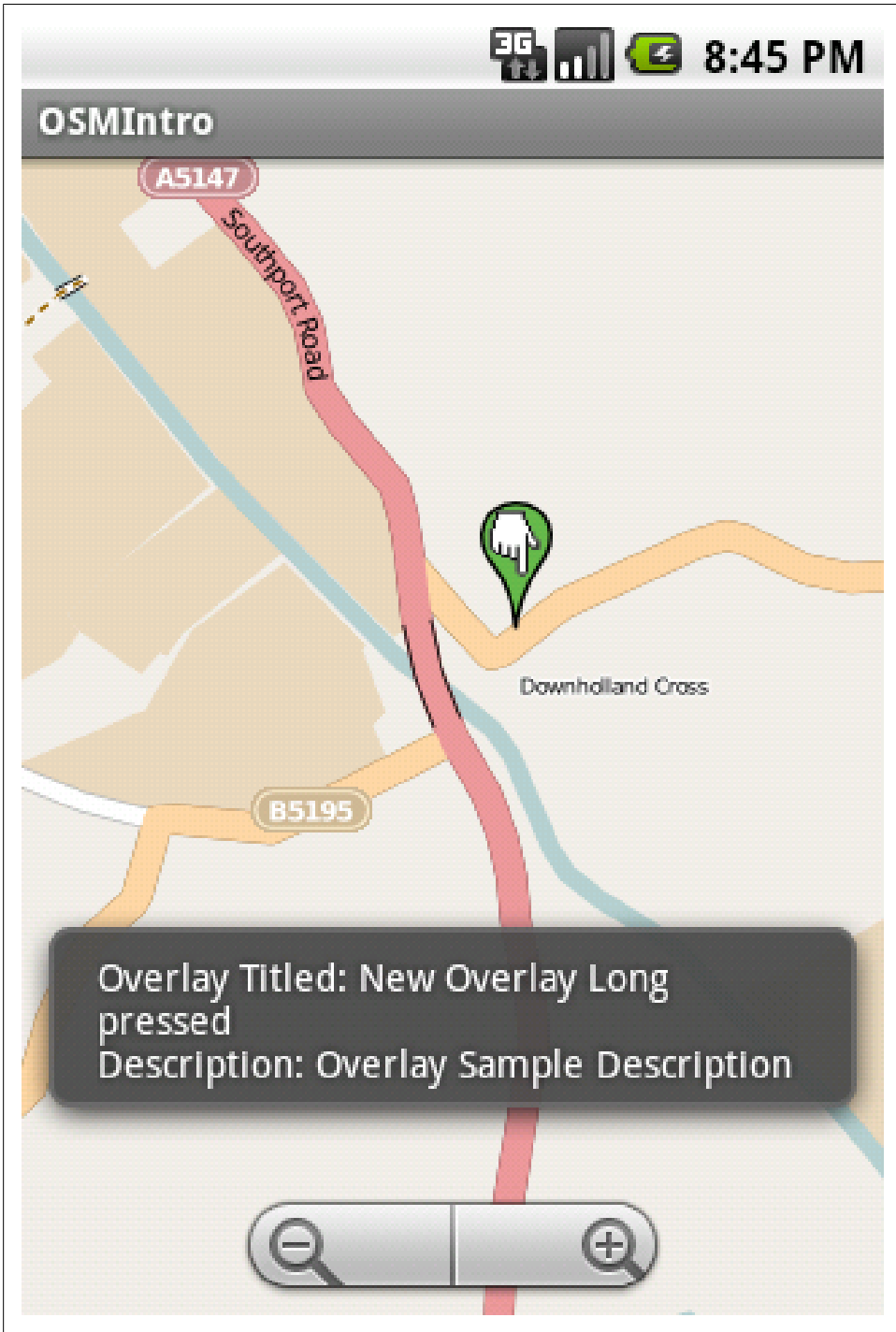


Figure 14-15. 586 | Chapter 14: Location and Map Applications

Problem

You need to react to the changes in the device's location and move the map focus to changed location

Solution

Using `LocationListener`, an application can request for location updates and then react to these changes in location

Discussion

- The activity that includes the OSM `MapView` needs to implement "`LocationListener`" to be able to request for changes in the device's location. An activity implementing `LocationListener` will also need to add the unimplemented (abstract) methods from `LocationListener` interface (Eclipse will do this for you). We set the center of the map to the `GeoPoint` named `mapCenter` so that the application starts with map focussed around that point.
- Now we need to get an instance of "`LocationManager`" and use to to request for location updates using the `requestLocationUpdates` method.
- One of the overridden methods (which were abstract in the `LocationListener` interface) named '`onLocationChanged`' we can write the code that we want to be executed when the location of the device changes.
- In the `onLocationChanged` method we obtain the latitude and longitude of the new location and set the map's center to the new `GeoPoint`.

Example 14-64.

```
public class LocationChange extends Activity implements LocationListener {
    private LocationManager myLocationManager;
    private MapView mapView;
    private MapController mapController;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mapView = (MapView)findViewById(R.id.mapview);
        mapController = this.mapView.getController();
        mapController.setZoom(15);
        GeoPoint mapCenter = new GeoPoint(53554070, -2959520);
        mapController.setCenter(mapCenter);
        myLocationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
        myLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 1000, 100, this);
    }

    @Override
    public void onLocationChanged(Location location) {
        int latitude = (int) (location.getLatitude() * 1E6);
```

```

        int longitude = (int) (location.getLongitude() * 1E6);
        GeoPoint geopoint = new GeoPoint(latitude, longitude);
        mapController.setCenter(geopoint);
        mapView.invalidate();
    }

    @Override
    public void onProviderDisabled(String arg0) {
    }

    @Override
    public void onProviderEnabled(String arg0) {
    }

    @Override
    public void onStatusChanged(String arg0, int arg1, Bundle arg2) {
    }
}

```

When the application starts, the map is centered around the mapCenter GeoPoint. Since the application is listening to location changes, the icon in the top bar of the phone is visible:

Now using the Emulator controls new GPS coordinates (-122.094095, 37.422006) are sent to the emulator. The application reacts to it and centers the map around then new coordinates:

Similarly, different GPS coordinates are given from the Emulator controls and the application centers the map around the new location:

Also, to allow the application to listen for location changes, include these permissions in the AndroidManifest.

Example 14-65.

```

<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />

```

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LNGViMzhmM2ItZGFjZC00NGVhLWJmNjctNTRjNTA0M2QzMjdh&hl=en_US



Figure 14-16.

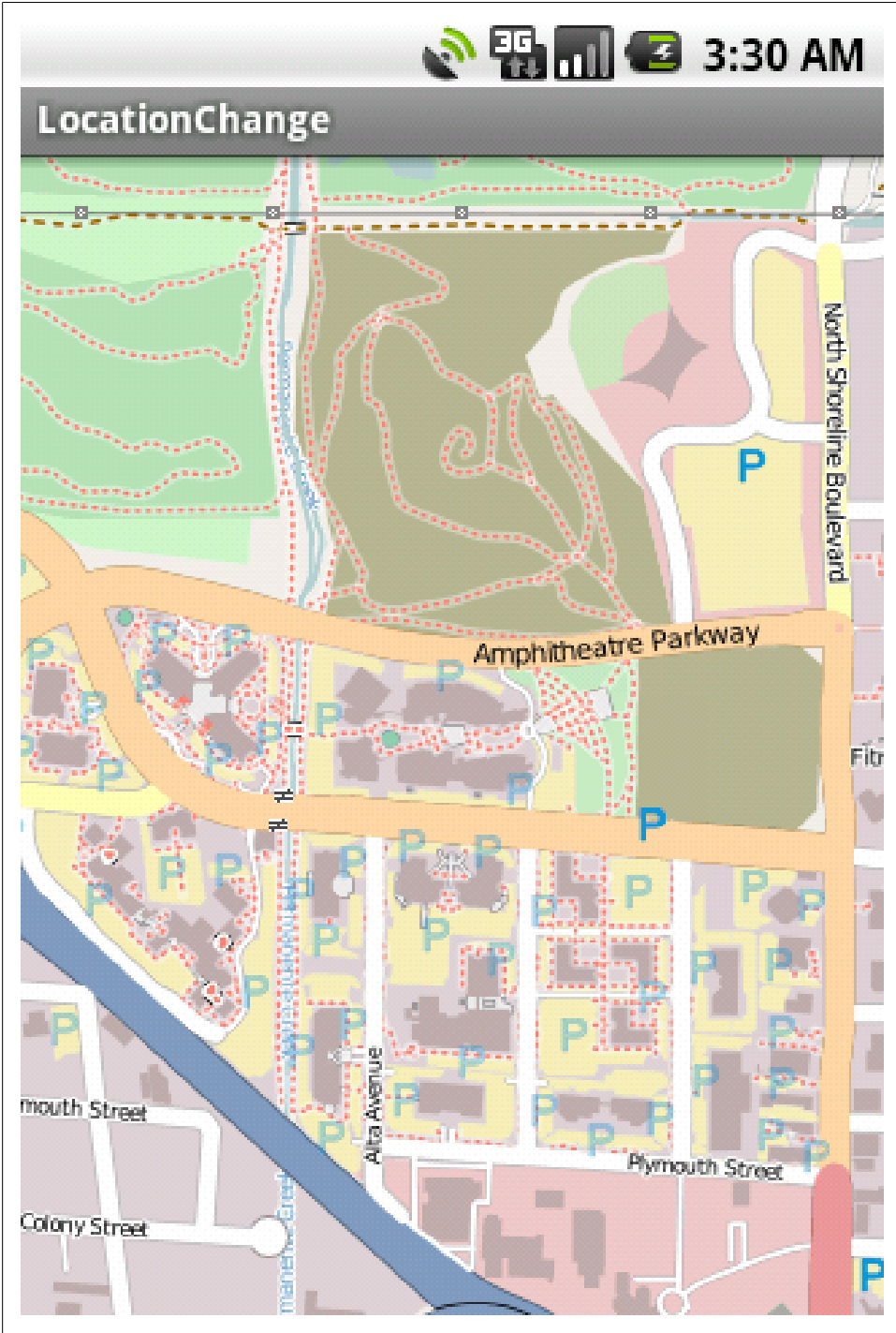


Figure 14-17: Location and Map Applications
590 | Chapter 14: Location and Map Applications



Figure 14-18.

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LNDM0NWVIYTUtNjAzNy00ODJjLTkwMmItNTFhOTFiZjk0ODdk&hl=en_US

Accelerometer

15.1 Using the accelerometer to detect shaking of the device

Thomas Manthey

Problem

Sometimes it makes sense to evaluate not only on-screen input, but also gestures like tilting or shaking the telephone. But how can you use the accelerometer to detect whether the phone has been shaken?

Solution

The solution is to register with the accelerometer and to compare the current acceleration values on all three axes to the previous ones. If the values have repeatedly changed on at least two axes and those changes exceed a high enough threshold, you can clearly determine shaking.

Discussion

Let us first define shaking as a fairly rapid movement of the device in one direction followed by further one in another direction, mostly but not necessarily the opposite. If we want to detect such a shake motion in an activity, we need a connection to the hardware sensors, those are exposed by the class `SensorManager`. Furthermore we need to define a `SensorEventListener` and register it with the `SensorManager`.

So the source of our activity starts like this:

Example 15-1.

```
public class ShakeActivity extends Activity {
    /* The connection to the hardware */
    private SensorManager mySensorManager;

    /* The SensorEventListener lets us wire up to the real hardware events */
    private final SensorEventListener mySensorEventListener = new SensorEventListener() {
```



```

    public void onSensorChanged(SensorEvent se) {
        /* we will fill this one later */
    }

    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        /* can be ignored in this example */
    }
};

....

```

To implement `SensorEventListener`, we have to implement two methods - `onSensorChanged(SensorEvent se)` and `onAccuracyChanged(Sensor sensor, int accuracy)`. The first one gets called whenever new sensor data is available, the second one whenever the accuracy of measurement changes, e.g. when the location service switches from GPS to network-based. In our example we just need to cover `onSensorChanged`.

Before we continue, let us define some more variables, which will store the information about values of acceleration and some state.

Example 15-2.

```

    /* Here we store the current values of acceleration, one for each axis */
    private float xAccel;
    private float yAccel;
    private float zAccel;

    /* And here the previous ones */
    private float xPreviousAccel;
    private float yPreviousAccel;
    private float zPreviousAccel;

    /* Used to suppress the first shaking */
    private boolean firstUpdate = true;

    /*What acceleration difference would we assume as a rapid movement? */
    private final float shakeThreshold = 1.5f;

    /* Has a shaking motion been started (one direction) */
    private boolean shakeInitiated = false;

```

I hope that the names and comments do explain enough about what is stored in these variables, if not, it will become clearer in the next steps. Now let us connect to the hardware sensors and wire up for their events, `onCreate` is the perfect place to do so.

Example 15-3.

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mySensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE); // (1)
    mySensorManager.registerListener(mySensorEventListener, mySensorManager

```

```

        .getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
        SensorManager.SENSOR_DELAY_NORMAL); // (2)
    }

```

At (1) we get a reference to Android's sensor service, at (2) we register the previously defined `SensorEventListener` with the service. More precisely, we register only for events of the accelerometer and for a normal update rate - this could be changed, if we needed to be more precise.

Now let us define what we want to do when new sensor data arrives. We have yet defined a stub for `SensorEventListener`'s method `onSensorChanged`, now we will fill it with some life.

Example 15-4.

```

    public void onSensorChanged(SensorEvent se) {
        updateAccelParameters(se.values[0], se.values[1], se.values[2]); // (1)
        if ((!shakeInitiated) && isAccelerationChanged()) { // (2)
            shakeInitiated = true;
        } else if ((shakeInitiated) && isAccelerationChanged()) { // (3)
            executeShakeAction();
        } else if ((shakeInitiated) && (!isAccelerationChanged())) { // (4)
            shakeInitiated = false;
        }
    }

```

Once again into the details:

At (1) we copy the values of acceleration which we received from the `SensorEvent` into our state variables. The corresponding method is declared like this:

Example 15-5.

```

    /* Store the acceleration values given by the sensor */
    private void updateAccelParameters(float xNewAccel, float yNewAccel,
        float zNewAccel) {
        /* we have to suppress the first change of acceleration, it results from first values being
        if (firstUpdate) {
            xPreviousAccel = xNewAccel;
            yPreviousAccel = yNewAccel;
            zPreviousAccel = zNewAccel;
            firstUpdate = false;
        } else {
            xPreviousAccel = xAccel;
            yPreviousAccel = yAccel;
            zPreviousAccel = zAccel;
        }
        xAccel = xNewAccel;
        yAccel = yNewAccel;
        zAccel = zNewAccel;
    }

```

At (2) we test for a rapid change of acceleration and whether any has happened before; if not, we store the information that now has happened.

At (3) we test again for a rapid change of acceleration, this time with another on before. If this is true, we can assume a shaking movement according to our definition and commence action.

At last at (4) we reset if we detected shaking before but do not get a rapid change of acceleration any more.

To complete the code, we add the last two methods, at first `isAccelerationChanged()`.

Example 15-6.

```
/* If the values of acceleration have changed on at least two axes, we are probably in a shake motion
private boolean isAccelerationChanged() {
    float deltaX = Math.abs(xPreviousAccel - xAccel);
    float deltaY = Math.abs(yPreviousAccel - yAccel);
    float deltaZ = Math.abs(zPreviousAccel - zAccel);
    return (deltaX > shakeThreshold && deltaY > shakeThreshold)
        || (deltaX > shakeThreshold && deltaZ > shakeThreshold)
        || (deltaY > shakeThreshold && deltaZ > shakeThreshold);
}
```

Here we compare the current values of acceleration with the previous ones, if at least two of them have changed above our threshold, we return true.

The last method is `executeShakeAction()` which does whatever we wish to do when being shaken.

Example 15-7.

```
private void executeShakeAction() {
    /* Save the cheerleader, save the world
       or do something more sensible... */
}
```

15.2 Introduction: Sensors

Ian Darwin

Discussion

Accelerometers are one of the more interesting bits of hardware in smartphones. Earlier devices like the OpenMoko "Neo" smart phone and the Apple iPhone included them. Before Android was released I was advocating for OpenMoko at open source conferences. One of my favorite imaginary applications was private key generation. Adhering to the theory that "When privacy is outlawed, only outlaws will have privacy," several people were talking about this as early as 2008 (when I presented the idea, to great applause, at the Ontario Linux Fest). The idea is: if you can't or don't want to exchange private keys over a public channel, you meet on a street corner and shake hands - with each hand having a cell phone concealed in the palm. The devices are touching each other, thus their sensors should record exactly the same somewhat random motions.

With a bit of mathematics to filter out the leading and trailing motion of the hands moving together, both devices should have quite a few bits' worth of identical, random data that nobody else has - just what you need for crypto key exchange. I've yet to see anybody implement this, and I must admit I still hope one of our contributors will come through.

Meanwhile we have many other recipes on accelerometers and other sensors in this chapter...

15.3 Checking for device facing up or facing down based on screen orientation using Accelerometer.

Rachee Singh

Problem

Check for the orientation (Facing up/Facing Down) of the Android device.

Solution

Checking for appropriate Accelerometer values.

Discussion

We require to listen for changes in the accelerometer values, for this purpose we need to implement a `SensorEventListener`. The `onSensorChanged` method is called when sensor values change. Within this method we check if the values lie within a particular range for the device to be facing down or facing up.

To obtain the sensor object for an accelerometer:

Example 15-8.

```
List<android.hardware.Sensor> sensorList = deviceSensorManager.getSensorList(Sensor.TYPE_ACCELEROMETER);  
sensor = sensorList.get(0);
```

Creating the listener:

Example 15-9.

```
private SensorEventListener accelerometerListener = new SensorEventListener() {  
    @Override  
    public void onSensorChanged(SensorEvent event) {  
        float z = event.values[2];  
        if (z > 9 && z < 10)  
            face.setText("FACE UP");  
        else if (z > -10 && z < -9)  
            face.setText("FACE DOWN");  
    }  
}
```

```

@Override
public void onAccuracyChanged(Sensor arg0, int arg1) {

}

};

```

After implementing the listener along with the methods required, we need to register the listener for a particular sensor (which in our case is the accelerometer). 'sensor' is an object of Sensor class, it represents the sensor being used in the application (Accelerometer).

Example 15-10.

```
deviceSensorManager.registerListener(accelerometerListener, sensor, 0, null);
```

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LZjQwYWI3MGEtYmI4NS00YjZjLWE0OGYtOTcxODY0OGZjYTI5&hl=en_US&authkey=CNq2oK4H

15.4 Finding the orientation of an Android device using Orientation sensor.

Rachee Singh

Problem

Being able to detect which side of the android device is higher up compared to the rest (Top/Bottom/Right/Left side)

Solution

By checking if the pitch and roll values of the Orientation sensor of an android device lie within certain intervals, it can be determined that which side is higher.

Discussion

Like in case of every other sensor supported by Android, we first require to instantiate the SensorManager class.

Example 15-11.

```
SensorManager sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
```

Using the object of SensorManager class we can get a handle on the sensors available on the device. The getSensorList() method returns a list of all sensors of a particular type (in this case Orientation). We need to check if Orientation sensor is supported by

the device, if it is, we get the first sensor from the list of sensors. In case the sensor is not supported, an appropriate message is displayed.

Example 15-12.

```
List<android.hardware.Sensor> sensorList = sensorManager.getSensorList(Sensor.TYPE_ORIENTATION);
if (sensorList.size() > 0) {
    sensor = sensorList.get(0);
}
else {
    orient.setText("Orientation sensor not present");
}
```

To register a `SensorEventListener` with this sensor:

Example 15-13.

```
sensorManager.registerListener(orientationListener, sensor, 0, null);
```

Now, we define the `SensorEventListener`. This would require implementing 2 methods: `onAccuracyChanged()` and `onSensorChanged()`. `onSensorChanged` is called when the sensor values change. In this case its the orientation sensor values that change on moving the device. Orientation sensor returns 3 values: azimuth, pitch and roll angles. Now we check the returned values, if they lie within particular range and depending upon the range they lie in, appropriate text is displayed.

Example 15-14.

```
private SensorEventListener orientationListener = new SensorEventListener() {

    @Override
    public void onAccuracyChanged(Sensor arg0, int arg1) {
    }

    @Override
    public void onSensorChanged(SensorEvent sensorEvent) {
        if (sensorEvent.sensor.getType() == Sensor.TYPE_ORIENTATION) {
            float azimuth = sensorEvent.values[0];
            float pitch = sensorEvent.values[1];
            float roll = sensorEvent.values[2];
            if (pitch < -45 && pitch > -135) {
                orient.setText("Top side of the phone is Up!");
            } else if (pitch > 45 && pitch < 135) {
                orient.setText("Bottom side of the phone is Up!");
            } else if (roll > 45) {
                orient.setText("Right side of the phone is Up!");
            } else if (roll < -45) {
                orient.setText("Left side of the phone is Up!");
            }
        }
    }
}
```

```
    }  
  }  
};
```

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LNzZiODY5YmMtNDAXMi00OGQwLWl3NmQtMGY1ZTdlN2E5MmI5&hl=en_US&authkey=COHZxYkE

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LMGJmYWFhNjktZjY5Ni00NTVvklWFlNmMtMTZjNmE4ZTFhNzU0&hl=en_US&authkey=CMG8z6QL

15.5 Checking for the Presence or Absence of a Sensor

Rachee Singh

Problem

Before using an Android device for a sensor-based application, the programmer needs to ensure that the required sensor is supported by the device.

Solution

Check for the availability of the sensor on the Android device.

Discussion

SensorManager class is used to manage the sensors available on an Android device. So we require an object of this class:

Example 15-15.

```
SensorManager deviceSensorManager = (SensorManager) getSystemService(SOME_SENSOR_SERVICE);
```

Then using the `getSensorList()` method we check for the presence of sensors of any type (accelerometer, gyroscope, pressure etc). If the list returned has `size>0`, implies that the sensor is present. A `TextView` is used to show the result: Sensor Present/Sensor Absent.

Example 15-16.

```
List<android.hardware.Sensor> sensorList = deviceSensorManager.getSensorList(Sensor.TYPE_ACCELEROMETE
```

```

if (sensorList.size() > 0) {
    sensorPresent = true;
    sensor = sensorList.get(0);
}
else
    sensorPresent = false;

/* Set the face TextView to display sensor presence */
face = (TextView) findViewById(R.id.face);

if (sensorPresent)
    face.setText("Sensor present!");
else
    face.setText("Sensor absent.");

```

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LZWJhODM4YzktYjc0YS00YmRmLTkzZTYtZDI0ZGRkZDYyNmM3&hl=en_US&authkey=COa-3YoC

15.6 Reading the Temperature Sensor

Rachee Singh

Problem

You need to get temperature values using the temperature sensor.

Solution

Using `SensorManager` and `SensorEventListener` to track changes in temperature values detected by the temperature sensor.

Discussion

We need to create an object of `SensorManager` to use sensors in an application. Then we register a listener with the type of sensor we require. To register the listener we provide the name of the listener, a `Sensor` object and the type of delay (in this case it is `SENSOR_DELAY_FASTEST`) to the `registerListener` method. In this listener, within the overridden `onSensorChanged` method, we can print the temperature value into a `textView` named `tempVal`.

Example 15-17.

```

SensorManager sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
sensorManager.registerListener(temperatureListener, sensorManager.getDefaultSensor(Sensor.TYPE_TEMPERATURE),

```


Example 15-18.

```
private final SensorEventListener temperatureListener = new SensorEventListener(){
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {}
    @Override
    public void onSensorChanged(SensorEvent event) {

        tempVal.setText("Temperature is:"+event.values[0]);

    }
};
```

See Also

[Recipe 15.5](#)

16.1 Introduction: Bluetooth

Ian Darwin

Discussion

Bluetooth technology allows connection of a variety of peripherals to a computer, tablet or phone. Headsets, speakers, keyboards, printers; medical devices such as glucometers, ECG; these and many more are available. Some such as headsets will be supported automatically by Android; the more esoteric ones will need some programming. Some of these other devices use SPP, which is basically an unstructured protocol that requires you to write code to format data yourself. We dont have any recipes on these yet, but we do have coverage of how to locate and discover the devices that try to connect.

16.2 Connecting to Bluetooth enabled device

Ashwini Shahapurkar

Problem

You need to connect to other bluetooth enabled device and want to communicate with it.

Solution

You will use the android bluetooth api to connect to device using sockets. The communication will be over the socket streams.

Discussion

For any bluetooth application you need to add these two permissions to Android-Manifest.xml

Example 16-1.

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

You will create the socket connection to the other bluetooth device. Then you shall continuously listen for the data from the socket stream in a thread. You can write to connected stream outside the thread. The connection is a blocking call and bluetooth device discovery being a heavy process, may slow down the connection. So it is a good practice to cancel the device discovery before trying to connect to other device.

Note: The bluetooth socket connection is a blocking call and returns only if a connection is successful or if an exception occurs while connecting to device.

The `BluetoothConnection` shall create the socket connection to other device, once instantiated and start listening to the data from connected device.

Example 16-2.

```
private class BluetoothConnection extends Thread {
    private final BluetoothSocket mmSocket;
    private final InputStream mmInStream;
    private final OutputStream mmOutStream;
    byte[] buffer;

    // Unique UUID for this application, you may use different
    private static final UUID MY_UUID = UUID
        .fromString("fa87c0d0-afac-11de-8a39-0800200c9a66");

    public BluetoothConnection(BluetoothDevice device) {

        BluetoothSocket tmp = null;

        // Get a BluetoothSocket for a connection with the given BluetoothDevice
        try {
            tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
        } catch (IOException e) {
            e.printStackTrace();
        }
        mmSocket = tmp;

        //now make the socket connection in separate thread to avoid FC
        Thread connectionThread = new Thread(new Runnable() {

            @Override
            public void run() {
                // Always cancel discovery because it will slow down a connection
                mAdapter.cancelDiscovery();

                // Make a connection to the BluetoothSocket
                try {
                    // This is a blocking call and will only return on a
                    // successful connection or an exception
                    mmSocket.connect();
                }
            }
        });
    }
}
```

```

        } catch (IOException e) {
            //connection to device failed so close the socket
            try {
                mmSocket.close();
            } catch (IOException e2) {
                e2.printStackTrace();
            }
        }
    }
});

connectionThread.start();

InputStream tmpIn = null;
OutputStream tmpOut = null;

// Get the BluetoothSocket input and output streams
try {
    tmpIn = socket.getInputStream();
    tmpOut = socket.getOutputStream();
    buffer = new byte[1024];
} catch (IOException e) {
    e.printStackTrace();
}

mmInStream = tmpIn;
mmOutStream = tmpOut;
}

public void run() {

    // Keep listening to the InputStream while connected
    while (true) {
        try {
            //read the data from socket stream
            mmInStream.read(buffer);
            // Send the obtained bytes to the UI Activity
        } catch (IOException e) {
            //an exception here marks connection loss
            //send message to UI Activity
            break;
        }
    }
}

public void write(byte[] buffer) {
    try {
        //write the data to socket stream
        mmOutStream.write(buffer);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void cancel() {

```

```
        try {
            mmSocket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

See Also

[Recipe 16.5](#)

16.3 Enabling Bluetooth and making the device Discoverable.

Rachee Singh

Problem

If the application requires the Bluetooth adapter to be switched On, the programmer needs to check if its enabled. If it is not, the use should be prompted to enable Bluetooth. For allowing remote devices to detect the host device, the host device should be made discoverable.

Solution

Use of Intents to prompt the user to enable Bluetooth and make the device discoverable.

Discussion

Before performing any action with an instance of BluetoothAdapter class, it should be checked if the device had enabled the Bluetooth adapter using the isEnabled() method. If it returns false then the user should be prompted to enable Bluetooth.

Example 16-3.

```
BluetoothAdapter BT = BluetoothAdapter.getDefaultAdapter();
if (!BT.isEnabled()) {
    //Taking user's permission to switch the bluetooth adapter On.
    Intent enableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
    startActivityForResult(enableIntent, REQUEST_ENABLE_BT);
}
```

This piece of code would show an AlertDialog to the user prompting her to enable Bluetooth.

On returning to the activity that started the intent, onActivityResult() is called in which, the name of the host device and its MAC address can be extracted.

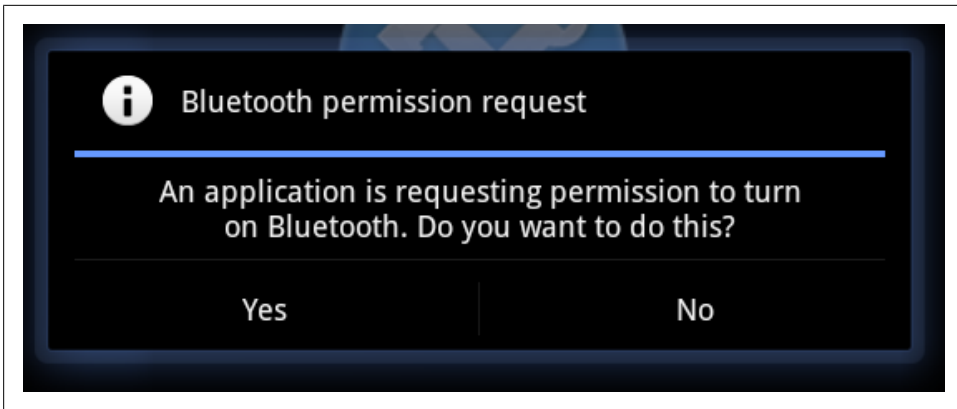


Figure 16-1.

Example 16-4.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if(requestCode==REQUEST_ENABLE_BT && resultCode==Activity.RESULT_OK) {
        BluetoothAdapter BT = BluetoothAdapter.getDefaultAdapter();
        String address = BT.getAddress();
        String name = BT.getName();
        String toastText = name + " : " + address;
        Toast.makeText(this, toastText, Toast.LENGTH_LONG).show();
    }
}
```

In order to make the device discoverable to other Bluetooth-enabled devices in the vicinity, these lines of code can be used to prompt take the user's permission.

Example 16-5.

```
//Taking user's permission to make the device discoverable for 120 secs.
Intent discoverableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
startActivity(discoverableIntent);
```

16.4 Listening for Bluetooth Connection Requests.

Rachee Singh

Problem

Creating a listening server for Bluetooth Connections.

Solution

For Bluetooth devices to interact, prior to the establishment of a connection, one of the communicating devices acts like a server. It obtains an instance BluetoothServerSocket

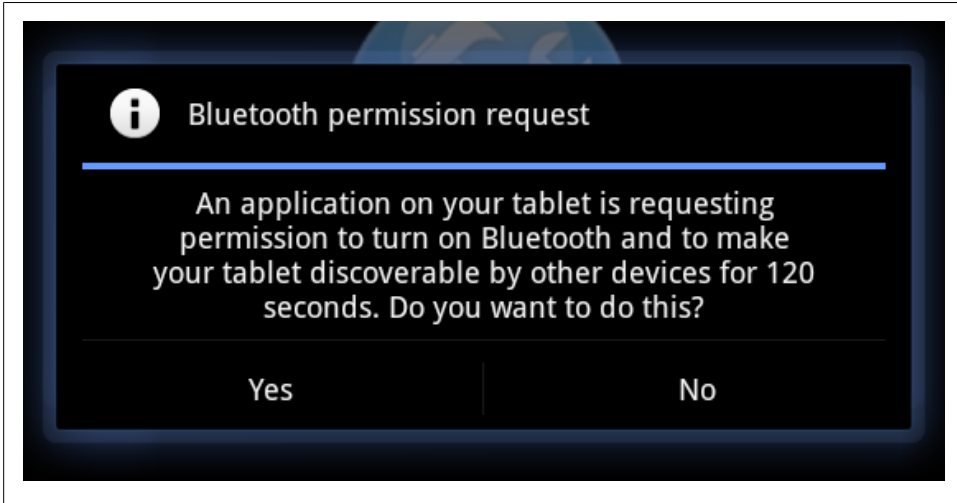


Figure 16-2.

and listens for incoming requests. This instance is obtained by calling the `listenUsingRfcommWithServiceRecord` method on the Bluetooth adapter

Discussion

With the instance of `BluetoothServerSocket`, we can start listening for incoming requests from remote devices through the `start()` method. Listening is a blocking process so we have to make a new thread and call it within the thread otherwise the UI of the application becomes unresponsive.

Example 16-6.

```
//Making the host device discoverable
startActivityForResult(new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE),DISCOVERY_REQUEST_BLUETOOTH)
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == DISCOVERY_REQUEST_BLUETOOTH) {
        boolean isDiscoverable = resultCode > 0;
        if (isDiscoverable) {
            UUID uuid = UUID.fromString("a60f35f0-b93a-11de-8a39-08002009c666");
            String serverName = "BTserver";
            final BluetoothServerSocket bluetoothServer = bluetoothAdapter.listenUsingRfcommWithServiceRecord(
                serverName, uuid);

            Thread listenThread = new Thread(new Runnable() {

                public void run() {
                    try {

                        BluetoothSocket serverSocket = bluetoothServer.accept();

                    } catch (IOException e) {
```

```
                Log.d("BLUETOOTH", e.getMessage());
            }
        }
    });
    listenThread.start();
}
}
```

16.5 Bluetooth Device discovery

Shraddha Shravagi

Problem

You want to display a list of Bluetooth devices in the vicinity.

Solution

Simple steps

1. Create XML file to display the list
2. Create a class file to load list
3. Edit manifest file

It's that simple.

Discussion

1 Create XML file to display the list

Example 16-7.

```
<ListView
    android:id="@+id/paired_devices"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
/>
```

2 Create a class file to load list

Example 16-8.

```
//IntentFilter will match the action specified
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
//broadcast receiver for any matching filter
this.registerReceiver(mReceiver, filter);

//attach the adapter
ListView newDevicesListView = (ListView)findViewById(R.id.new_devices);
```



```

newDevicesListView.setAdapter(mNewDevicesArrayAdapter);

filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
this.registerReceiver(mReceiver, filter);

//Create a receiver for the Intent
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {

    @Override
    public void onReceive(Context context, Intent intent) {
        // TODO Auto-generated method stub

        String action = intent.getAction();

        if(BluetoothDevice.ACTION_FOUND.equals(action)){
            BluetoothDevice btDevice = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);

            if(btDevice.getBondState() != BluetoothDevice.BOND_BONDED){
                mNewDevicesArrayAdapter.add(btDevice.getName()+"\n"+btDevice.getAddress());
            }
        }
        else
            if(BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)){
                setProgressBarIndeterminateVisibility(false);
                setTitle(R.string.select_device);
                if(mNewDevicesArrayAdapter.getCount() == 0){
                    String noDevice = getResources().getText(R.string.none_paired).toString();
                    mNewDevicesArrayAdapter.add(noDevice);
                }
            }
    }
};

```

3 Edit manifest file

Example 16-9.

...

Source Download URL

The source code for this example may be downloaded from this URL: <http://projects.darwinsys.com/AndroidBluetoothDemo.zip>

System and Device Control

17.1 Phone network/connectivity information

Amir Alagic

Problem

You want to find information about phones network connectivity

Solution

You can find out whether your phone is connected to the network, its type of connection, and whether your phone is in roaming territory, using the Connectivity Manager and a NetworkInfo object.

Discussion

Often you need to know whether your phone can connect to the internet and since roaming can be expensive it is also very useful if we can tell to the app user if he is in roaming (the user who is truly worried about this will disable data roaming using the Settings application). To find this and more we can use the NetworkInfo class in the android.net package.

Example 17-1.

```
ConnectivityManager connManager = (ConnectivityManager)this.getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo ni = connManager.getActiveNetworkInfo();
/*Indicates whether network connectivity is possible.
A network is unavailable when a persistent or semi-persistent
condition prevents the possibility of connecting to
that network.*/
boolean available = ni.isAvailable();
/*Indicates whether network connectivity is possible.
A network is unavailable when a persistent
or semi-persistent condition prevents the possibility
of connecting to that network. Examples include*/
```

```
boolean connected = ni.isConnected();
boolean roaming = ni.isRoaming();
/* Reports the type of network (currently mobile or Wi-Fi) to which the info in this object pertains
int networkType = ni.getType();
```

17.2 Changing incoming call notification to Silent, Vibrate, or normal

Rachee Singh

Problem

You need to put the Android device to silent, vibrate or normal mode

Solution

Use Android's AudioManager system service to turn the phone to Normal, Silent and Vibrate modes

Discussion

Here is a sample application that has 3 buttons to change the phone mode to Silent, Vibrate and Normal:

We instantiate the AudioManager class to be able to use the setRingerMode method. For each of these buttons (silentButton, normalButton and vibrateButton) we have OnClickListener defined in which we used the AudioManager object to set the ringer mode. We also display a Toast notifying the mode change.

Example 17-2.

```
am= (AudioManager) getBaseContext().getSystemService(Context.AUDIO_SERVICE);
silentButton = (Button)findViewById(R.id.silent);
normalButton = (Button)findViewById(R.id.normal);
vibrateButton = (Button)findViewById(R.id.vibrate);

//For Silent mode
silentButton.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View arg0) {
        am.setRingerMode(AudioManager.RINGER_MODE_SILENT);
        Toast.makeText(getApplicationContext(), "Silent Mode Activated.", Toast.LENGTH_LONG).show();
    }
});

//For Normal mode
normalButton.setOnClickListener(new View.OnClickListener() {
```

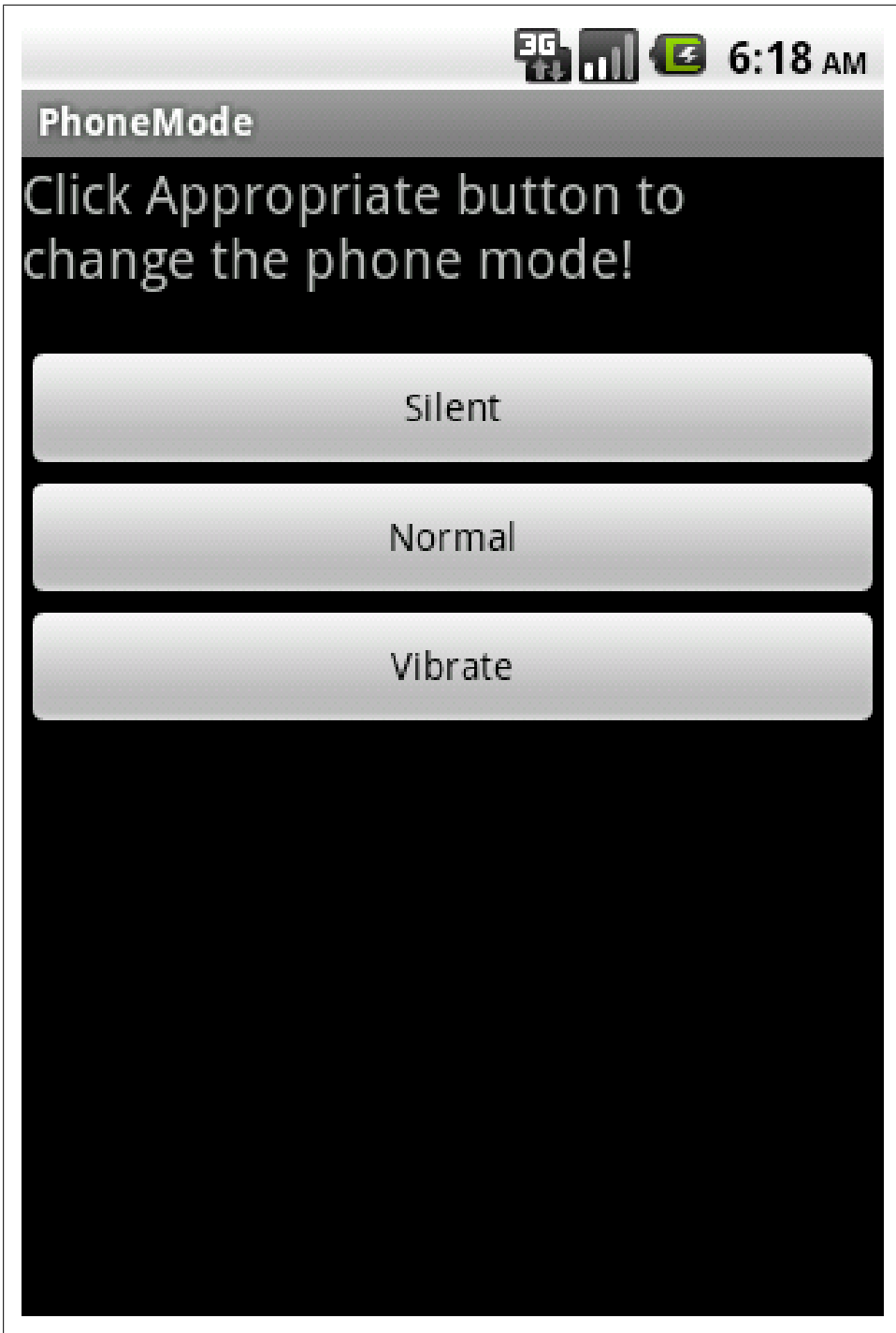


Figure 17-1.

```

        @Override
        public void onClick(View arg0) {
            am.setRingerMode(AudioManager.RINGER_MODE_NORMAL);
            Toast.makeText(getApplicationContext(), "Normal Mode Activated", Toast.LENGTH_LONG).show();
        }
    });

    //For Vibrate mode
    vibrateButton.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View arg0) {
            am.setRingerMode(AudioManager.RINGER_MODE_VIBRATE);
            Toast.makeText(getApplicationContext(), "Vibrate Mode Activated", Toast.LENGTH_LONG).show();
        }
    });

```

This is how the application looks when the Silent button is clicked. (Notice the silent icon in the status bar of the phone.)

17.3 Rebooting the Device

Ian Darwin

Problem

For some strange reason, you want the user's Android device to reboot.

Solution

- Ask for REBOOT permission in your AndroidManifest.
- Get a PowerManager instance from getSystemService(), and call its reboot method.

Discussion

Since rebooting the operating system of the mobile device (phone or tablet) is so drastic, you must ask for it in your AndroidManifest.xml:

Example 17-3.

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

Even that will probably not be enough, however. You should be aware that the REBOOT permission has its `protectionLevel` parameter set to `signatureOrSystem`, meaning that it is only intended for use by (e.g, the application must be signed by) the organization that is creating the Android System Image. So unless you are using a custom-built system image, you will not be able to use this function!

Then, if you're really sure, go ahead and reboot:

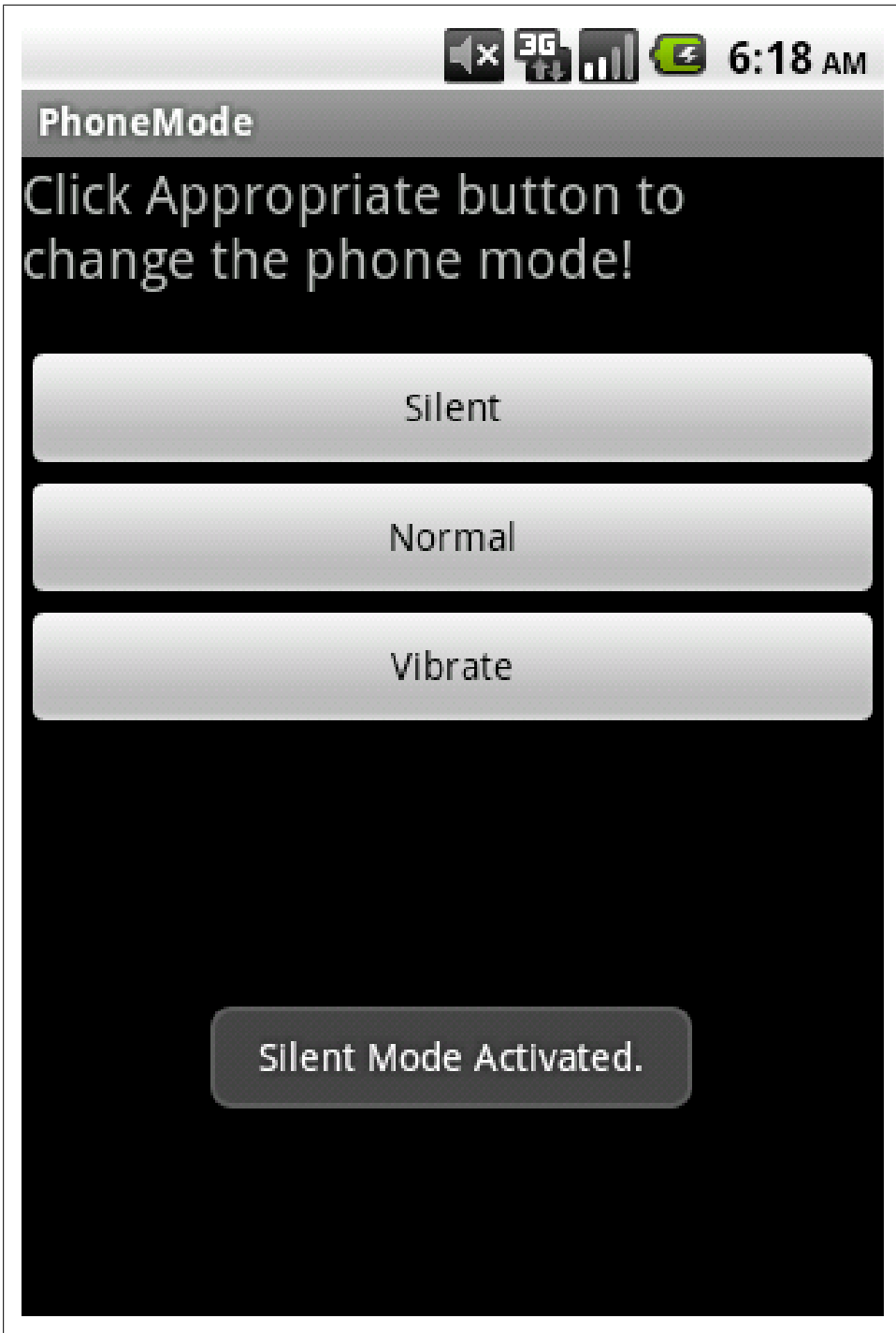


Figure 17-2.

Example 17-4.

```
PowerManager pwrMgr = (PowerManager)mContext.getSystemService(Context.POWER_SERVICE);  
pwrMgr.reboot(null);
```

Again, if the application attempting the reboot is not signed with the same certificate as the system image, the permission in the manifest will be silently ignored, and you will get an Application Stopped Unexpectedly dialog.

The argument to the `reboot` call is named "reason", but the values are a few that are specially known to the kernel; this is not intended for logging the shutdown reason. Use the normal logging API for that.

Remember, just because you *can* do something does not mean that you should! There are very few circumstances where this is likely to be "socially acceptable". Hopefully most users will not capriciously install apps with this permission without a good reason...

See Also

<http://developer.android.com/reference/android/os/PowerManager.html>

17.4 Running shell commands from your application

Rachee Singh

Problem

You need to run a shell command from your application (for instance `pwd`, `ls` etc).

Solution

Using the `exec()` method of the `Runtime` class we can provide the shell command we wish to run as an argument.

Discussion

Applications can not create an instance of `Runtime` class, but they can get an instance by invoking `getRuntime()` method. Using this instance we call the `exec` method which executes the specified program in a separate native process. It takes the name of the program to execute as an argument. The `exec` method returns the new `Process` object that represents the native process.

As an example, we run the `ps` command that lists all the process running on the system. The full location of the command is specified (`/system/bin/ps`) as an argument to `exec()`.

We get the output of the command and return the `String`. `process.waitFor()` is used to wait for the command to finish executing.

Example 17-5.

```
try {
    Process process = Runtime.getRuntime().exec("/system/bin/ps");
    InputStreamReader reader = new InputStreamReader(process.getInputStream());
    BufferedReader bufferedReader = new BufferedReader(reader);
    int numRead;
    char[] buffer = new char[5000];
    StringBuffer commandOutput = new StringBuffer();
    while ((numRead = bufferedReader.read(buffer)) > 0) {
        commandOutput.append(buffer, 0, numRead);
    }
    bufferedReader.close();
    process.waitFor();

    return commandOutput.toString();
} catch (IOException e) {
    throw new RuntimeException(e);
} catch (InterruptedException e) {
    throw new RuntimeException(e);
}
```

Here is how the output of the ps command looks:

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LNTkxMDIlyYTgtMzlmMS00ZDVlLThkOTUtYWY4MjQ5NGY1NzFk&hl=en_US

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LNzE1MjhhkODUtOTZkMS00YjhlLTg2YTgtM2U2MmUzZW44NDMw&hl=en_US

17.5 Copying text and getting text from the Clipboard

Rachee Singh

Problem

You need to copy text to the clipboard and access the text stored on the clipboard

Solution

With the help of the ClipboardManager class, we can access the items stored on the clipboard of an Android device

ShellCommand

USER	PID	PPID	VSIZE	RSS	WCHAN	PC
root	1	0	312	220	c009b74c	0000ca4c
S	/init					
root	2	0	0	0	c004e72c	00000000 S
kthreadd						
root	3	2	0	0	c003fdc8	00000000 S
ksoftirqd/0						
root	4	2	0	0	c004b2c4	00000000 S
events/0						
root	5	2	0	0	c004b2c4	00000000 S
khelper						
root	6	2	0	0	c004b2c4	00000000 S
suspend						
root	7	2	0	0	c004b2c4	00000000 S
kblockd/0						
root	8	2	0	0	c004b2c4	00000000 S
cqueue						
root	9	2	0	0	c018179c	00000000 S
kseriod						
root	10	2	0	0	c004b2c4	00000000 S
kmmcd						
root	11	2	0	0	c006fc74	00000000 S
pdflush						
root	12	2	0	0	c006fc74	00000000 S
pdflush						
root	13	2	0	0	c00744e4	00000000 S

Figure 17-3.

Discussion

ClipboardManager class allows us to copy text to the clipboard using the setText method and also allows us to get the text stored on the clipboard using the getText method. getText returns a CharSequence which is converted to a String by the toString() method.

Here is a sample code, demonstrating how to obtain an instance of the ClipboardManager class and using it to copy text to the clipboard. Then the getText method is used to get the text on the clipboard and the text is set to a TextView.

Example 17-6.

```
ClipboardManager clipboard = (ClipboardManager) getSystemService(CLIPBOARD_SERVICE);
clipboard.setText("Using the clipboard for the first time!");
String clip = clipboard.getText().toString();
clipTextView = (TextView) findViewById(R.id.clipText);
clipTextView.setText(clip);
```

17.6 Making LED based notifications

Rachee Singh

Problem

Some Android phones are equipped with LEDs. An application can flash different color lights using the LED.

Solution

Using the NotificationManager and Notification class we can make notifications using the LED on the device.

Discussion

As is in case of all notifications, we instantiate the NotificationManager class. Then we create a Notification class's object. Using the method ledARGB() we can specify the color of the LED light. The constant ledOnMS is used to specify the time in milliseconds for which the LED will be on and ledOffMS specifies the time in milliseconds for which the LED is off. The notify method starts the notification process. Here's the code corresponding to the actions just described:

Example 17-7.

```
NotificationManager notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
Notification notification = new Notification();
notification.ledARGB = 0xff0000ff;           // Blue color light flash
notification.ledOnMS = 1000;                // LED is on for 1 second
notification.ledOffMS = 1000;               // LED is off for 1 second
```

```
notification.flags = Notification.FLAG_SHOW_LIGHTS;
notificationManager.notify(0, notification);
```

17.7 Making the Device Vibrate.

Rachee Singh

Problem

Through your application, you wish to notify the user of an event by means of device's vibration.

Solution

Using Notifications to set a vibration pattern.

Discussion

To use device vibration, include this permission in the AndroidManifest file:

Example 17-8.

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

In the Java code, we need to get an instance of the NotificationManager class and of Notification class:

Example 17-9.

```
NotificationManager notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
Notification notification = new Notification();
```

To set a pattern for the vibration, assign a sequence of long values (time in millisecond) to Notification's vibrate property. This sequence represents the time for which the device will vibrate and the time for which it will pause vibration, alternatively. For instance: The pattern used in this example will cause the device to vibrate for 1 second then pause for 1 second then vibrate again for 1 second and so on.

Example 17-10.

```
notification.vibrate = new long[]{1000, 1000, 1000, 1000, 1000};
notificationManager.notify(0, notification);
```

Source Download URL

The source code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LZjjiMTU5MzEtYzk3NC00NTcxLWE0NDAtMDVjY2I3ZWVmMG13&hl=en_US&authkey=CJ2SjpAC

Binary Download URL

The executable code for this example may be downloaded from this URL: https://docs.google.com/leaf?id=0B_rESQKgad5LZTNiY2E3YTYtYjc3Yy00OTFiLTlkNTQtZGRlYzFhNzUzYzBh&hl=en_US&authkey=COzAjd0P

17.8 Determining Whether a Given Application is Running

Colin Wilcox

Problem

How do i know if mine or some other app is running?

Solution

The system activity manager maintains a list of all active tasks. This provides the name of all running tasks and can be interrogated for various system specific information

Discussion

This method takes the name of an application and returns true if the ActivityManager thinks it is currently running.

Example 17-11.

```
import android.app.ActivityManager;
import android.app.ActivityManager.RunningAppProcessInfo;

public boolean isAppRunning (String aApplicationPackageName)
{
    ActivityManager activityManager = (ActivityManager) this.getSystemService(ACTIVITY_SERVICE);
    if (activityManager == null)
    {
        return false; // should report: can't get Activity Manager
    }

    List<RunningAppProcessInfo> procInfos =
        activityManager.getRunningAppProcesses();
    for(int idx = 0; idx < procInfos.size(); idx++)
    {
        if(procInfos.get(i).processName.equals(aApplicationPackageName))
        {
            return true;
        }
    }

    return false;
}
```

Other Programming Languages

18.1 Run external/native Linux command

Amir Alagic

Problem

Sometimes it can be convenient to start one of the Linux commands that are available on the phone such as `rm`, `sync`, `top`, `uptime`...

Solution

To run Linux commands available on the Android OS you should use classes that are available in standard Java and are used to start external processes. First you have to know which command you want to run and then get/obtain `Runtime` object and then execute the native command in a separate native process. Often you will need to read results and to do that use streams to do that.

Discussion

Since Java is such a powerful language it is pretty simple to start external processes.

With `AndroZip File Manager` or other you can find Linux commands in `./system/bin` folder. One of the commands is `ls` which lists the files (and subfolders) in a folder. To run this command we will send its path to the `Runtime.exec` method.

You can not create a `Runtime` object directly since it is a singleton; to obtain its instance you call the static `getRuntime()` method and then pass the path to the Linux command you want to run.

Example 18-1.

```
Process process = Runtime.getRuntime().exec("/system/bin/ls");
```

The Process class is used above to create the process; it will also help us read from the process and we obtain an InputStream that is connected to the standard output stream (stdout) of the native process represented by this object.

Example 18-2.

```
DataInputStream osRes = new DataInputStream(process.getInputStream());
```

Then we create BufferedReader object which will help us to read results line by line.

Example 18-3.

```
BufferedReader reader = new BufferedReader(new InputStreamReader(osRes));  
  
String line;  
  
while ((line = reader.readLine()) != null || reader.read() != -1) {  
    Log.i("Reading command result", line);  
}
```

And as you see we will read all lines and show them on LogCat console. You can see the output for example in your Eclipse IDE.

You could, of course, capture the output of any system command back into your program and either parse it for display in e.g., a ListView, or display it as text in a TextView.

18.2 Running Adobe Air/Flex on Android

Wagied Davids

Problem

Developing AIR/Flex Apps for Android.

Solution

Necessary SDK downloads:

1. Download the Flex 4.1 SDK: <http://opensource.adobe.com/wiki/display/flexsdk/Download+Flex+4>
2. Download the Adobe AIR 2.5 SDK: <http://www.adobe.com/products/air/sdk/>
3. Download the Android SDK: http://dl.google.com/android/android-sdk_r08-windows.zip

Steps involved:

1. Create a directory : C:\Library\AIR4Android
2. First, extract/unzip the Android SDK 2.2 - think of this as the top/base layer

3. Second, extract/unzip the Flex 4.1 SDK
4. Third, copy the the AIR 2.5 sdk ZIP file into the extracted Flex SDK folder
5. Fourth, extract/unzip the AIR2.5 SDK OVER-WRITING the existing FLEX SDK files/folders
6. Push the AIR2.5 runtime onto the Android Emulator using the command: `adb install .\Runtime.apk`

Discussion

The technique involves overlaying the Adobe AIR sdk over the the Flex sdk. Once this is done the necessary Flex development environment is set-up. For FLASH and AIR applications to run on Android, an AIR runtime is required. To install the runtime on the emulator or device the adb tool is used. After the runtime is installed, AIR apps can be created using Flex Builder

Note: This works only with Android 2.2 upwards which have Flash player support.

18.3 Getting Started with "Scripting Layer for Android" (formerly Android Scripting Environment)

Ian Darwin

Problem

You want to write your application in one of several popular scripting languages, or, you want to program interactively on your phone.

Solution

Here's how to get started:

- Download the Scripting Layer for Android (formerly Android Scripting Environment) from <http://code.google.com/p/android-scripting/>
- Add the interpreter(s) you want to use
- Type in your program
- Run it immediately - no compilation or packaging steps needed!

Discussion

The SL4A application is not (yet) in the Android Market, so you have to visit the web site and download it (there is a QRCode for downloading, so start in your laptop or desktop browser). And since it's not in the market, before you can download it you'll have to go into Settings->Applications->Unknown Sources and enable unknown-

sourced applications. Also note that since this is not downloaded via the Market, you will not be notified when the Google project releases a new binary.

Once you have the SL4A binary installed, you must start it and download the particular interpreter you want to use. The following are available as of this writing:

- Python
- Perl
- JRuby
- Lua
- BeanShell
- JavaScript
- Tcl
- Unix shell

Some of the interpreters (like JRuby) run in the Dalvik VM, while others (like Python) run the "native" versions of the language under Linux on your phone. Communication happens via a little server that is started automatically when needed or can be started from the Interpreters menu bar.

The technique for downloading new interpreters is a bit sub-obvious. When you start the SL4A application it shows a list of scripts, if you have any. Press the menu button, then go to the View menu, and select Interpreters (while here notice that you can also view the Logcat, the system exception log file). From the Interpreters list, pressing Menu *again* will get you a menu bar with an Add button, and this lets you add another interpreter.

Pick a language (Python)

Suppose you think Python is a great language (which it is).

Once your interpreter is installed, go back to the SL4A main page and click the Menu button, then Add (in this context, Add creates a new file, not another interpreter). Select the installed interpreter and you'll be in Edit Mode. We're trying Python, so type in this canonical "hello world" example:

Example 18-4.

```
import android
droid = android.Android()
droid.makeToast("Hello, Android")
```

Press the Menu button, and "Save and Run" if enabled, or "Save and Exit" otherwise. The former will run your new app; the latter will return you to the list of scripts, in which case you want to tap your script's name. In the resulting pop-up, the choices are (left to right):

- Run ("DOS box" icon)
- disabled
- Edit ("pencil" icon)
- Save ("1980 floppy disk icon")
- Delete (trash can icon).

If you long-press a file name, a pop-up gives you the choice of Rename or Delete.

When you run this trivial application, you will see the Toast near the bottom of your screen.

Source Editing

If you want to keep your scripts in a source repository, and/or if you prefer to edit them on a laptop or desktop with a traditional keyboard, just copy the files back and forth (if your phone is rooted, you can probably run your repository directly on the phone). Scripts are stored in "sl4a/scripts" on the SD card. If you have your phone mounted on your laptop's /mnt folder, for example, you might see this (On Windows it might be E: or F: instead of /mnt):

Example 18-5.

```
laptop$ ls /mnt/sl4a/
Shell.log          hello_world.py.log    phonepicker.py.log    say_weather.py.log    speak.py.log
demo.sh.log        ifconfig.py.log       say_chat.py.log       scripts                take_picture.py.
dialer.py.log      notify_weather.py.log say_time.py.log       sms.py.log            test.py.log
laptop$ ls /mnt/sl4a/scripts
bluetooth_chat.py dialer.py              hello_world.py        notify_weather.py     say_chat.py          say_weather.py
demo.sh            foo.sh                ifconfig.py           phonepicker.py        say_time.py          sms.py
laptop$
```

See Also

The official web site is <http://code.google.com/p/android-scripting/>; there is a QRCode there to download the latest binary.

Binary Download URL

The executable code for this example may be downloaded from this URL: <http://code.google.com/p/android-scripting/>

18.4 Running Native Code with JNI on the NDK

Ian Darwin

Problem

You need to run parts of your application natively in order to use existing C/C++ code or, possibly, to improve performance of CPU-intensive code

Solution

Use JNI (Java Native Interface) via the [Android Native Development Kit](#).

Discussion

Standard Java has always allowed you to load *native* or compiled code into your Java program, and Android's Dalvik runtime supports this in a way that is pretty much identical with the original. Why would you as a developer want to do such a thing? One reason might be to access OS-dependent functionality. Another is speed: native code will likely run faster than Java, at least at present, although there is some contention as to how much difference this really makes. Search the web for conflicting answers.

The native code language bindings are defined for code that has been written in C or C++. If you need to access a language other than C/C++, you could write a bit of C/C++ and have it pass control to other functions or applications, but you should also consider using the (BROKEN XREF TO RECIPE -1 'Android Scripting Environment').

For this example I use a simple numeric calculation, computing the square root of a `double` using the simple Newton-Raphson iterative method. The code provides both a Java and a C version, to compare the speeds.

Ian's Basic Steps: Java Calling Native Code

To call native code from Java:

1. Install the Android Native Development Kit (NDK) in addition to the ADK.
2. Write Java code that declares and calls a native method.
3. Compile this Java code.
4. Create a `.h` header file using `javah`.
5. Write a C function that includes this header file and implements the native method to do the work.
6. Prepare the `Android.mk` (and optionally `Application.mk`) configuration files.
7. Compile the C code into a loadable object using `$NDK/ndk-build`.
8. Package and deploy your application, and test it.

The preliminary step is to download the NDK as a tar or zip file, extract it someplace convenient, and set the environment variable such as `NDK` to where you've installed it, for referring back to the NDK install. You'll want this to read documentation as well as to run the tools.

The first step is to write Java code that declares and calls a native method. To declare the method, use the keyword `native` to indicate that the method is native. To use the native method, no special syntax is used, but your application - typically in your main Activity - must provide a static code block that loads your native method using `System.loadLibrary()`. (The dynamically loadable module will be created in Step 6.) Static blocks are executed when the class containing them is loaded; loading the native code here ensures it is in memory when needed!

Object variables that your native code may modify should carry the `volatile` modifier. In my example, `SqrtDemo.java` contains the native method declaration (as well as a Java implementation of the algorithm).

Example 18-6.

```
public class SqrtDemo {

    public static final double EPSILON = 0.05d;

    public static native double sqrtC(double d);

    public static double sqrtJava(double d) {
        double x0 = 10.0, x1 = d, diff;
        do {
            x1 = x0 - (((x0 * x0) - d) / (x0 * 2));
            diff = x1 - x0;
            x0 = x1;
        } while (Math.abs(diff) > EPSILON);
        return x1;
    }
}
```

The Activity class `Main.java` uses the native code:

Example 18-7.

```
// In the Activity class, outside any methods:
static {
    System.loadLibrary("sqrt-demo");
}

// In a method of the Activity class where you need to use it:
double d = SqrtDemo.sqrtC(123456789.0);
```

The next step is simple; just build the project normally, using the ADK Eclipse Plugin or Ant.

Next, you need to create a C-language `.h` header file that provides the interface between the JVM and your native code. Use `javah` to produce this file. `javah` needs to read the class that declares one or more native methods, and will generate a `.h` file specific to the package and class name.

Example 18-8.

```
mkdir jni // keep everything JNI-related here
javah -d jni -classpath bin foo.ndkdemo.SqrtDemo // produces foo_ndkdemo_SqrtDemo.h
```

The `.h` file produced is a "glue" file, not really meant for human consumption and particularly not for editing. But by inspecting the resulting `.h` file, you'll see that the C method's name is composed of the name Java, the package name, the class name, and the method name:

Example 18-9.

```
JNIEXPORT jdouble JNICALL Java_foo_ndkdemo_SqrtDemo_sqrtC
(JNIEnv *, jclass, jdouble);
```

Then create a C function that does the work. You must import the `.h` file and use the same function signature as is used in the `.h` file.

This function can do whatever it wishes. Note that it is passed two arguments before any declared arguments: a JVM environment variable and a "this" handle for the invocation context object. The table shows the correspondence between Java types and the C types (JNI types) used in the C code.

Table 18-1. Java and JNI types

"Java type"	"JNI"	"Java array type"	"JNI"
byte	jbyte	byte[]	jbyteArray
short	jshort	short[]	jshortArray
int	jint	int[]	jintArray
long	jlong	long[]	jlongArray
float	jfloat	float[]	jfloatArray
double	jdouble	double[]	jdoubleArray
char	jchar	char[]	jcharArray
boolean	jboolean	boolean[]	jbooleanArray
void	jvoid		
Object	jobject	Object[]	jobjectArray
Class	jclass		
String	jstring		
array	jarray		
Throwable	jthrowable		

The following is the complete C native implementation. It simply computes the square root of the input number, and returns that. The method is static, so there is no use made of the "this" pointer.

Example 18-10.

```
// jni/sqrt-demo.c

#include <stdlib.h>

#include "foo_ndkdemo_SqrtDemo.h"

JNIEXPORT jdouble JNICALL Java_foo_ndkdemo_SqrtDemo_sqrtC(
    JNIEnv *env, jclass clazz, jdouble d) {

    jdouble x0 = 10.0, x1 = d, diff;
    do {
        x1 = x0 - (((x0 * x0) - d) / (x0 * 2));
        diff = x1 - x0;
        x0 = x1;
    } while (labs(diff) > foo_ndkdemo_SqrtDemo_EPSILON);
    return x1;
}
```

The implementation is basically the same as the Java version. Note that *javah* even maps the final double EPSILON from the Java class SqrtDemo into a *#define* for use within the C version.

The next step is to prepare the file *Android.mk*, also in the *jni* folder. For a simple shared library, this example will suffice:

Example 18-11.

```
# Android.mk

LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

LOCAL_MODULE := sqrt-demo
LOCAL_SRC_FILES := sqrt-demo.c

include $(BUILD_SHARED_LIBRARY)
```

Finally, you compile the C code into a loadable object. Under desktop Java, the details depend on platform, compiler, etc. However, the NDK provides a build script to automate this. Assuming you have set the NDK variable to the install root of the NDK download from Step 1, you only need to type

Example 18-12.

```
$ $NDK/ndk-build # for Linux, Unix, OS-X?
> %NDK%/ndk-build # for MS-Windows

Compile thumb : sqrt-demo <= sqrt-demo.c
SharedLibrary : libsqrtdemo.so
Install       : libsqrtdemo.so => libs/armeabi/libsqrtdemo.so
```

And you're done! Just package and run the application normally. The full download example for this chapter includes buttons to run the `sqrt` function a number of times in either Java or C and compare the times. Note that at present it does this work on the event thread, so large numbers of repetitions will result in "Application Not Responding" errors, which will mess up the timing.

Congratulations! You've called a native method. Your code may run slightly faster. However, you will require extra work for portability; as Android begins to run on more hardware platforms, you will have to (at least) add them to the `Application.mk` file. If you have used any assembler code, the problem is much worse.

Beware that problems with your native code can and will crash the runtime process right out from underneath the Java Virtual Machine. The JVM can do nothing to protect itself from poorly written C/C++ code. Memory must be managed by the programmer; there is no automatic garbage collection of memory obtained by the system runtime allocator. You're dealing directly with the operating system and sometimes even the hardware, so, 'Be careful. Be very careful.'

See Also

There is a recipe in Chapter 26 of the Java Cookbook which shows variables from the Java class being accessed from within the native code. The official documentation for Android's NDK is at [The Android Native SDK information page](#). Considerable documentation is included in the `docs` folder of the NDK download. If you need more information on Java Native Methods, you might be interested in the comprehensive treatment found in *Essential JNI: Java Native Interface* by Rob Gordon (Prentice Hall), originally written for Desktop Java.

Source Download URL

The source code for this example may be downloaded from this URL: <http://projects.darwinsys.com/ndkdemo-src.zip>

18.5 Introduction: Other Programming Languages

Ian Darwin

Discussion

Developing new programming languages is a constant process in this industry. Several new (or not-so-new) languages have become popular recently: Erlang, Scala, Clojure, Groovy, C#, F#, and more. While the Apple approach on the iPhone has been to mandate use of Objective C and to ban (at least officially, it seems to be honored more in the breach) use of other languages, particularly JVM-style translated languages, Android positively encourages use of many languages. You can write your app in pure

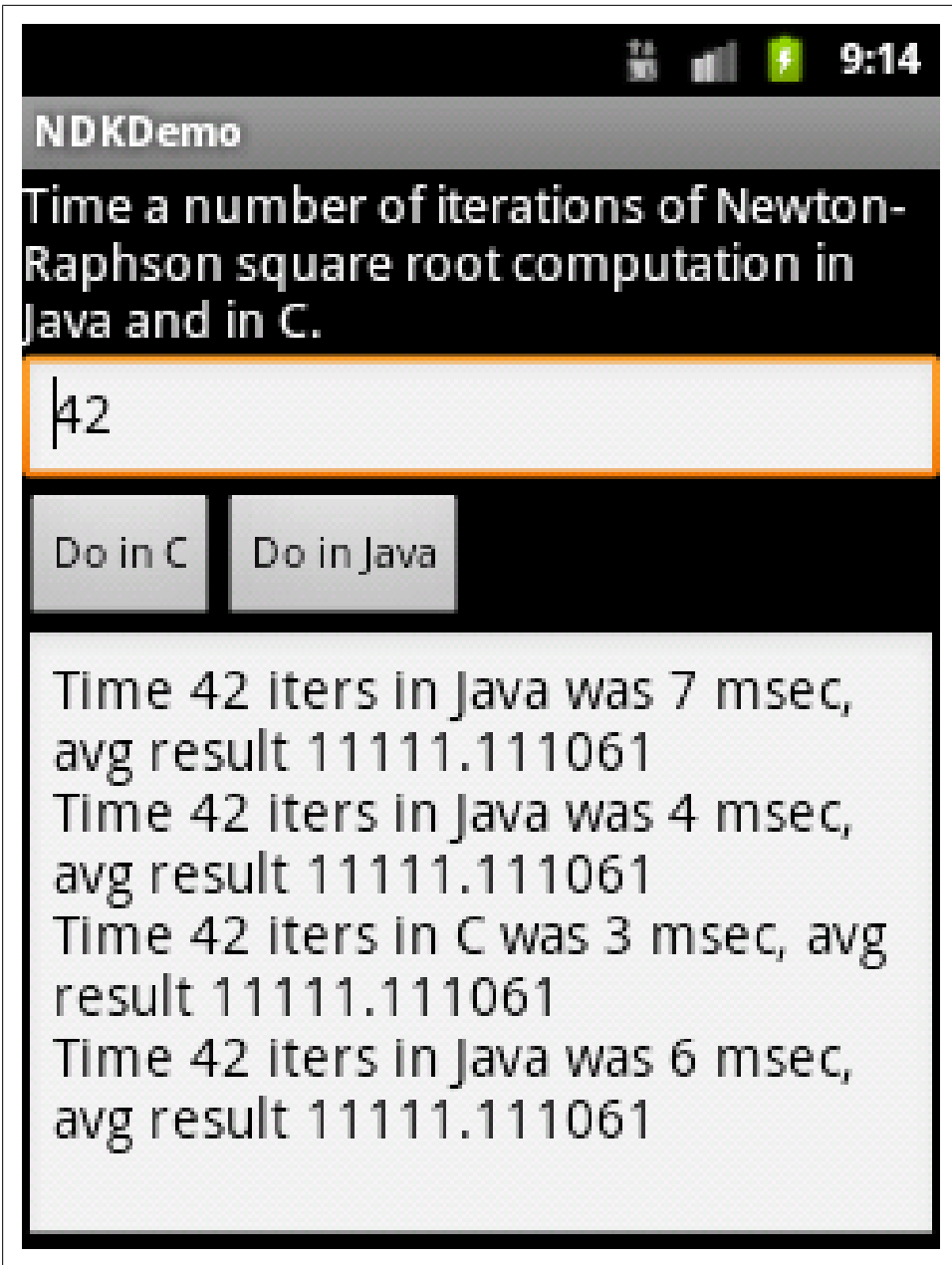


Figure 18-1.

Java using the SDK, of course - that's the subject of most of the rest of the book. You can mix in some C/C++ code into Java using (BROKEN XREF TO RECIPE -1 'Running native code|native code'), using Android's NDK. People have made most of the major

compiled languages work, especially (but not exclusively) the JVM-based ones. You can write using a variety of scripting languages like Perl, Python, Ruby, see (BROKEN XREF TO RECIPE -1 'Getting Started with Scripting Layer for Android'). And there's more...

If you want a very high-level, drag-and-drop development process, look at (BROKEN XREF TO RECIPE -1 'Android Application Inventor'), Google's own toolkit for building applications easily. At present these can't easily be packaged as apps, but this will probably come fairly soon.

If you are a web developer used to working your magic in JavaScript and CSS, there is a route for you to become an Android Developer using the tools you already know. There are, in fact, five or six technologies that go this route, such as AppCelerator Titanium, (list others here...). These mostly use CSS to build a style, JavaScript to provide actions, and W3 standards to provide device access such as GPS. Most of these work by packaging up a JavaScript interpreter along with your HTML and CSS into an APK file. Many of these have the further advantage that they can be packaged to run on iPhone, Blackberry, and other mobile platforms. The risk I see with these is that, since they're not using native toolkit items, they may easily provide strange-looking user interfaces that don't conform either to the Guidelines or to users' expectations of how apps should behave on the Android platform. That is certainly something to be aware of if you are using one of these toolkits.

Whether to use the standard SDK or to go the HTML/CSS route is a continuing debate. To view the pros and cons on both side, check out [this DEVOXX presentation](#).

One of the key ideas in Android was to keep it as an open platform. The wide range of languages that you can use to develop Android apps testifies that this openness has been maintained.

18.6 Intro to Flex 4.5 Android Programming

Wagied Davids

Problem

Creating an Android-application using Flex 4.5. The application uses a company stock symbol as query, and an HTTP service to Google's Finance API, retrieving stock data.

Solution

Flex Builder Burrito and Flex 4.5 - Detailed Screenshots

Discussion

File: GoogleStockApp.mxml

Example 18-13.

```
<s:TabbedViewNavigatorApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
                                xmlns:s="library://ns.adobe.com/flex/spark">
  <s:ViewNavigator label="Search" width="100%" height="100%" firstView="views.SearchView"/>
  <fx:Declarations>
    <!-- Place non-visual elements (e.g., services, value objects) here -->
  </fx:Declarations>
</s:TabbedViewNavigatorApplication>
```

File: File: views.SearchView.mxml

Example 18-14.

```
<s:View xmlns:fx="http://ns.adobe.com/mxml/2009"
        xmlns:s="library://ns.adobe.com/flex/spark"
        xmlns:googlestockservicelookup="services.googlestockservicelookup.*"
        xmlns:googlestockservice="services.googlestockservice.*"
        title="Search">

  <fx:Script>
    XXXCDATA[
      import valueObjects.StockResult;
      protected function getStockQuote(stock:String):void
      {
        getStockQuoteResult.token = googleStockService.getStockQuote(stock);
      }

    XXX
  </fx:Script>

  <fx:Declarations>
    <s:CallResponder id="getStockQuoteResult"/>
    <googlestockservice:GoogleStockService id="googleStockService"/>
  </fx:Declarations>

  <s:HGroup x="10" y="32" width="460" height="71">
    <s:TextInput id="input" width="362" height="68" prompt="Stock Symbol"/>
    <s:Button height="69" label="Enter" click="getStockQuote(input.text);"/>
  </s:HGroup>
  <s:Scroller x="10" y="121" width="460" height="488">
    <s:VGroup width="100%" height="100%">
      <s:VGroup width="458" height="487">
        <s:HGroup width="456" height="49">
          <s:Label text="Time: "/>
          <s:Label text="{getStockQuoteResult.lastResult.finance.trade_timestamp.data.toString()}">
          <s:Label text="{getStockQuoteResult.lastResult.finance.trade_time_utc.data.toString()}">
        </s:HGroup>
        <s:HGroup width="456" height="49">
          <s:Label text="Company: "/>
          <s:Label text="{getStockQuoteResult.lastResult.finance.company.data.toString()}">
        </s:HGroup>
        <s:HGroup width="456" height="49">
          <s:Label text="Exchange"/>
          <s:Label text="{getStockQuoteResult.lastResult.finance.exchange.data.toString()}">
        </s:HGroup>
      </s:VGroup>
    </s:VGroup>
  </s:Scroller>
</s:View>
```

```

<s:HGroup width="456" height="49">
  <s:Label text="Currency: "/>
  <s:Label text="{getStockQuoteResult.lastResult.finance.currency.data.toString()}" />
</s:HGroup>
<s:HGroup width="456" height="49">
  <s:Label text="Open: "/>
  <s:Label text="{getStockQuoteResult.lastResult.finance.open.data.toString()}" />
</s:HGroup>
<s:HGroup width="456" height="49">
  <s:Label text="Close: "/>
  <s:Label text="{getStockQuoteResult.lastResult.finance.y_close.data.toString()}" />
</s:HGroup>
<s:HGroup width="456" height="49">
  <s:Label text="Change: "/>
  <s:Label text="{getStockQuoteResult.lastResult.finance.change.data.toString()}" />
  <s:Label text=" {getStockQuoteResult.lastResult.finance.perc_change.data.toString()} %" />
</s:HGroup>
<s:HGroup width="456" height="49">
  <s:Label text="Last: "/>
  <s:Label text="{getStockQuoteResult.lastResult.finance.last.data.toString()}" />
  <s:Label text="High: "/>
  <s:Label text="{getStockQuoteResult.lastResult.finance.high.data.toString()}" />
  <s:Label text="Low: "/>
  <s:Label text=" {getStockQuoteResult.lastResult.finance.low.data.toString()}" />
</s:HGroup>
<s:HGroup width="456" height="49">
  <s:Label text="Volume: "/>
  <s:Label text="{getStockQuoteResult.lastResult.finance.volume.data.toString()}" />
  <s:Label text="Ave. Volume: "/>
  <s:Label text="{getStockQuoteResult.lastResult.finance.avg_volume.data.toString()}" />
</s:HGroup>
</s:VGroup>
</s:VGroup>
</s:Scroller>

</s:View>

```

Note that the parts marked XXXCDATA and XXX in the above must be changed to be valid XML CDATA wrapper; we can't show that here as the Android Cookbook production software uses CDATA around source attachments, and XML doesn't let you nest CDATA elements...

Binary Download URL

The executable code for this example may be downloaded from this URL: <http://www.filefactory.com/file/cc04e7b/n/GoogleStockApp.zip>

18.7 Sharing your scripts (ASE) using QR codes

Rachee Singh

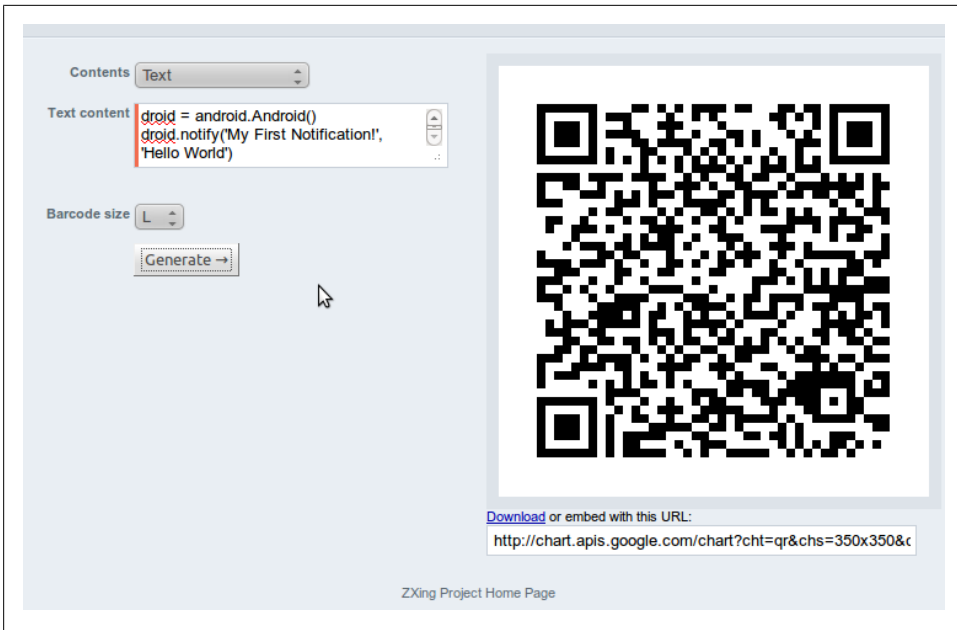


Figure 18-2.

Problem

You need to distribute your ASE scripts using QR (Quick Response) Codes

Solution

Use <http://zxing.appspot.com/generator/> or many other QR code generators to generate a QR code for your script.

Discussion

QR codes are a great way to share your scripts if they are short (since QR codes can only encode 4,296 characters of content). Follow these simple steps to generate a QR code for your script:

- Visit (BROKEN XREF TO RECIPE -1 '<http://zxing.appspot.com/generator/>').
- Select 'text' from the drop down menu.
- In text content, put the script's name in the first line.
- From the next line onwards, enter the script.
- Choose 'L' in barcode size and click generate.
- This is how it looks:

18.8 Using native handset functionality from webview using Javascript

Colin Wilcox

Problem

The advent of HTML5 as standard in many browsers means that applications can exploit the features of the HTML5 standard to create applications much more quickly than if they were written in native Java. This sounds great for many applications but alas not all of the cool functionality on the device is accessible through HTML5 and Javascript. Webkits attempt to bridge the gap but may not provide all the functionality needed in all cases.

Solution

You can invoke Java code in response to Javascript events using a bridge between the JS and Java environments.

Discussion

The idea is to tie up events within the Javascript embedded in an HTML5 webpage and handle the event on the Java side by calling native code.

The example below creates a button in HTML5 embedded in a webview which when clicked causes the contacts application to be invoked on the device through the Intent mechanism

Example 18-15.

```
import android.content.Context;
import android.content.Intent;
import android.util.Log;
```

Write some thin bridge code:

Example 18-16.

```
public class JavaScriptInterface
{
    private static final String TAG = "JavaScriptInterface";
    Context mContext = null;

    /** Instantiate the interface and set the context */
    JavaScriptInterface(Context mContext)
    {
        // save the local context for later use
        mContext = mContext;
    }
}
```

```

public void launchContacts();
{
    mContext.startActivity(contactIntent);
    launchNativeContactsApp ();
}
}

```

The native code to actually launch contacts is below:

Example 18-17.

```

private void launchNativeContactsApp()
{
    String packageName = "com.android.contacts";
    String className = ".DialtactsContactsEntryActivity";
    String action = "android.intent.action.MAIN";
    String category1 = "android.intent.category.LAUNCHER";
    String category2 = "android.intent.category.DEFAULT";

    Intent intent = new Intent();
    intent.setComponent(new ComponentName(packageName, packageName + className));
    intent.setAction(action);
    intent.addCategory(category1);
    intent.addCategory(category2);
    startActivity(intent);
}

```

The Javascript which ties this all together is shown below. In this case the call is triggered by a click event

Example 18-18.

```

<input type="button" value="Say hello" onClick="showAndroidContacts()" />
<script type="text/javascript">
    function showAndroidContacts()
    {
        Android.launchContacts();
    }
</script>

```

The only preconditions is that the web browser has Javascript enabled and the interface is known. This is done by

Example 18-19.

```

WebView iWebView = (WebView) findViewById(R.id.webview);
iWebView.addJavascriptInterface(new JavaScriptInterface(this), "Android");

```

Internationalization

19.1 Introduction: Internationalization

Ian Darwin

Discussion

"All the world's a stage," wrote William Shakespeare. But not all the players upon that great and turbulent stage speak the great Bard's native tongue. To be usable on a global scale, your software needs to communicate in many different languages. The menu labels, button strings, dialog messages, titlebar titles, and even command-line error messages must be settable to the user's choice of language. This is the topic of internationalization and localization. Because these words take a long time to say and write, they are often abbreviated by their first and last letters and the count of omitted letters, that is, I18N and L10N.

If you've got your Strings in a separate XML file as we advised in Chapter 1, Getting Started, you have already done part of the work of Internationalizing your app. Aren't you glad you followed our advice?

Android provides a `Locale` class to discover/control the internationalization settings. A default `Locale` is inherited from the user's Language settings when your App starts up.

Note that if you know Internationalization from Desktop Java, it's pretty much the same. We'll explain as we go along, with examples, in this chapter.

Ian's Basic Steps: Internationalization

Internationalization and localization consist of:

- Sensitivity training (Internationalization or I18N): making your software sensitive to these issues;
- Language lessons (Localization or L10N): writing configuration files for each language;

- Culture lessons (optional): customizing the presentation of numbers, fractions, dates, and message-formatting. Images that mean different things in different cultures.

There are examples of doing these things in the Recipes of this chapter.

See Also

See also by [Java Internationalization](#) by Andy Deitsch and David Czarnecki (O'Reilly).

19.2 Internationalizing Application Text

Ian Darwin

Problem

You want the text of your buttons, labels and so on to appear in the user's chosen language.

Solution

Create a new `strings.xml` in the `res/values-XX/` subdirectory of your application. Translate the string values to the given language.

Discussion

Every Android Project created with the SDK has a file called `strings.xml` in the `res/values` directory. This is where you are recommended to place all your application's strings, from the application title through to the button text and even down to the contents of dialogs.

When you refer to a string by name, either by a reference like `android:text="@string/hello"` in a layout file or by a lookup like `getString(R.string.hello)`, you look up the string's value from this file.

To make all of these strings available in a different language, you need to know the correct ISO-3166 language code; a few common ones are shown in the table.

Table 19-1. Common Languages and Codes

Language	Code
Chinese (traditional)	cn-tw
Chinese (simplified)	cn-zh
English	en
French	fr
German	de
Italian	it

Language	Code
Spanish	es

With this information, you can create a new subdirectory `res/values-XX/` (where `XX` is replaced by the ISO language code). In this directory you create a copy of `strings.xml` and in it, translate the individual string values. For example, a simple application might have the following `strings.xml`:

Example 19-1.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">Hello Android</string>
  <string name="app_name">MyAndroid</string>
</resources>
```

You might create `res/values-es/strings.xml` containing the following:

Example 19-2.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">Hola Android</string>
  <string name="app_name">MiAndroid</string>
</resources>
```

And create the file `res/values-fr/strings.xml` containing the following:

Example 19-3.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello">Bonjour Android</string>
  <string name="app_name">MonAndroid</string>
</resources>
```

Now when you look up the string "hello" using either of the methods described earlier, you will get the version based on the user's language choice. If the user selects a language that you don't have a L10N file for, the app will still work, but will get the value from the default file - the one in the `values` directory with no language code. For most of us, that will contain the English values, but it's up to the developer.

Similarly, if there is a string that's not defined in a language-specific file, the app will find the version of it from the default `strings.xml` file.

Is it really that simple?

Yes. Just package your application and deploy it (if you're using Eclipse, just Run As Android Application). Go into the Settings app of your emulator or device, choose Language, and select French or Spanish and the program title and window contents should reflect the change.

You just have to remember to keep the versions of `strings.xml` in sync.

Regional Variants

OK, so it's not quite that simple. There are also regional variations within a language. In English there are, for example, UK English (a.k.a. "the real thing" by some), US English, Canadian, Australian, and so on. These, fortunately, have tended to use the same vocabulary for technical terms, so using the regional variations is not as important for English. French and Spanish, to name just two, are languages where there is significant variation from in vocabulary from one region to another. Parisian French and French Canadian have used different vocabularies for many words coined since the 1500's when their exodus began. The many Spanish colonies were likewise largely isolated from hearing each others' speech for hundreds of years - from their founding until the age of radio - and they have diverged even more than French. So you may want to create 'variant' files for these languages, as for any other that has significant regional variation.

Android's practice here diverges slightly from Java's, in that Android uses a letter 'r' to denote regional variations, e.g., `values-fr-rCA` for French Canadian. Note that as in Java, language codes are in lower case and variations (which are usually the two-letter ISO country code) are written in capital letters (except for the leading 'r'). So we might wind up with the set of files listed in the table below.

Table 19-2. L10N Directory Examples

Directory	
<code>values</code>	English; default.
<code>values-es</code>	Spanish ("Castilian", generic)
<code>values-es-rCU</code>	Spanish - Cuban
<code>values-es-rCL</code>	Spanish - Chilean

See Also

There is a bit more detail at the official [Android Localization documentation](#).

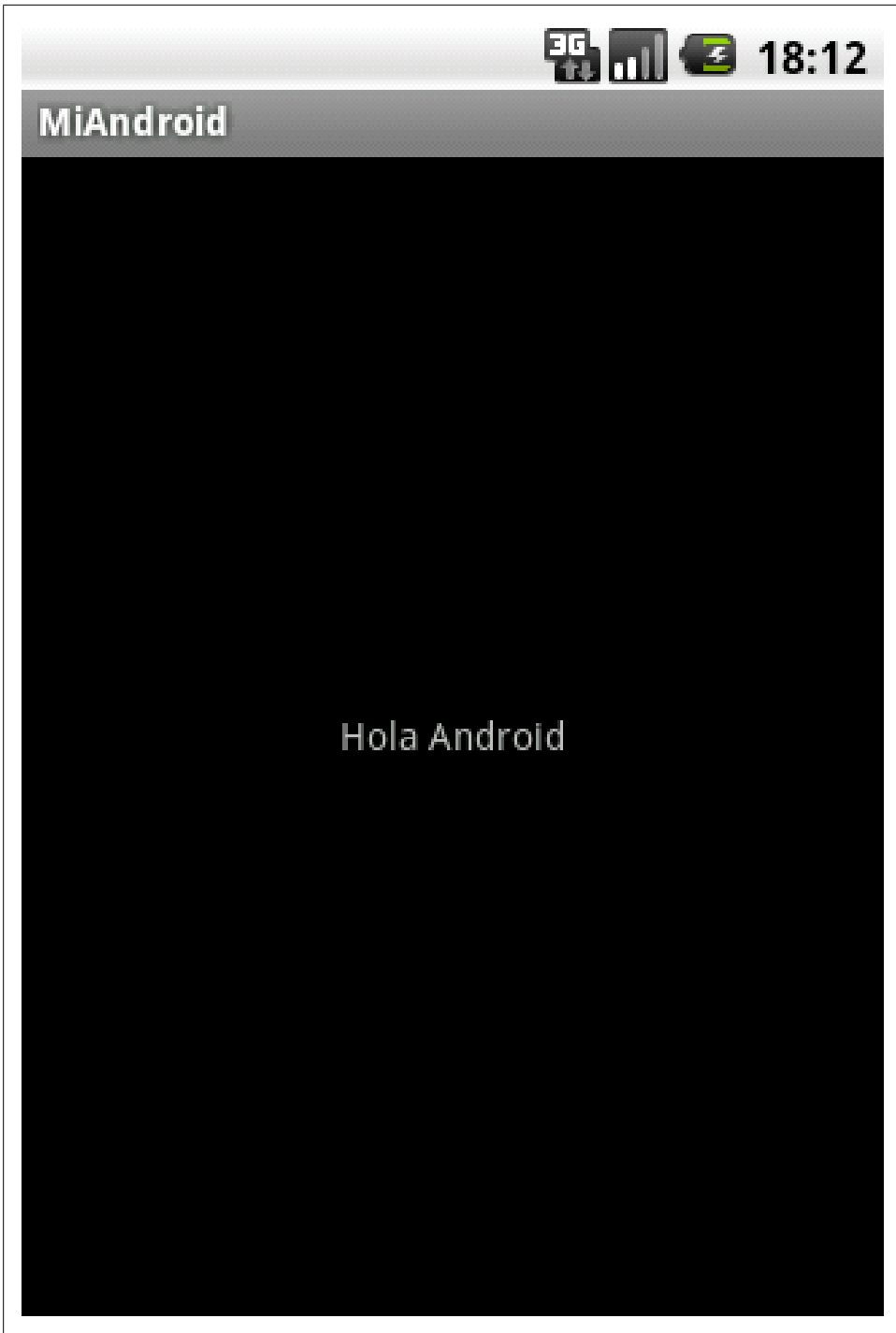


Figure 19-1.

Packaging, deploying and selling

20.1 Signing Your Application

Zigurd Mednieks

Problem

You want to sign your application prior to uploading it to the Android Market.

Solution

An APK file is a standard Java Archive (JAR) format, so you just use the standard JDK tool `jarsigner`.

Discussion

Having created a key, and a Map API key if needed, you are almost ready to sign your application, but first you need to create an unsigned version that you can sign with your digital certificate. To do that, in the Package Explorer window of Eclipse, right-click on your project name. You'll get a long pop-up menu; toward the bottom, click on Android Tools. You should see another menu that includes the item you want: "Export Unsigned Application Package...". This item takes you to a File Save dialog box, where you can pick the place to save the unsigned version of your apk file. It doesn't matter where you put it, just pick a place you can remember. Now that you have an unsigned version of your apk file, we can go ahead and sign it using `jarsigner`.

Open a terminal or command window in the directory where you stored the unsigned apk file. To sign `MyApp`, using the key (BROKEN XREF TO RECIPE -1 'Creating a Signing Certificate|generated earlier'):

Example 20-1.

```
$ jarsigner -verbose -keystore myapp.keystore MyApp.apk mykey
```

You should now have a signed version of your application that can be loaded and run on any Android device. But before you send it in to Android Market, there's one more intervening step.... You have re-built the application, so you must test it again, on real devices. If you don't have a real device get one. If you only have one, get more, or make friends with somebody that owns a device from a different manufacturer.

20.2 How to integrate Admob into your app

Enrique Diaz

Problem

You want to monetize your free app by showing ads within it.

Solution

Using Admob Libraries you can start using ads in your free app, getting money for each click.

Discussion

AdMob is one of the world's largest mobile advertising networks, offering solutions for discovery, branding and monetization on mobile phones.

The AdMob Android SDK contains the code necessary to install AdMob ads in your application.

Step 1

In your project's root directory create a subdirectory **libs**. This will already be done for you if you used Android's activitycreator tool. Copy the AdMob JAR (admob-sdk-android.jar) file into that libs directory.

For Eclipse projects:

1. Right-click on your project from the Package Explorer tab and select **Properties**
2. Select **Java Build Path** from left panel
3. Select **Libraries** tab from the main window
4. Click on **Add JARs...**
5. Select the JAR copied to the libs directory
6. Click **OK** to add the SDK to your Android project

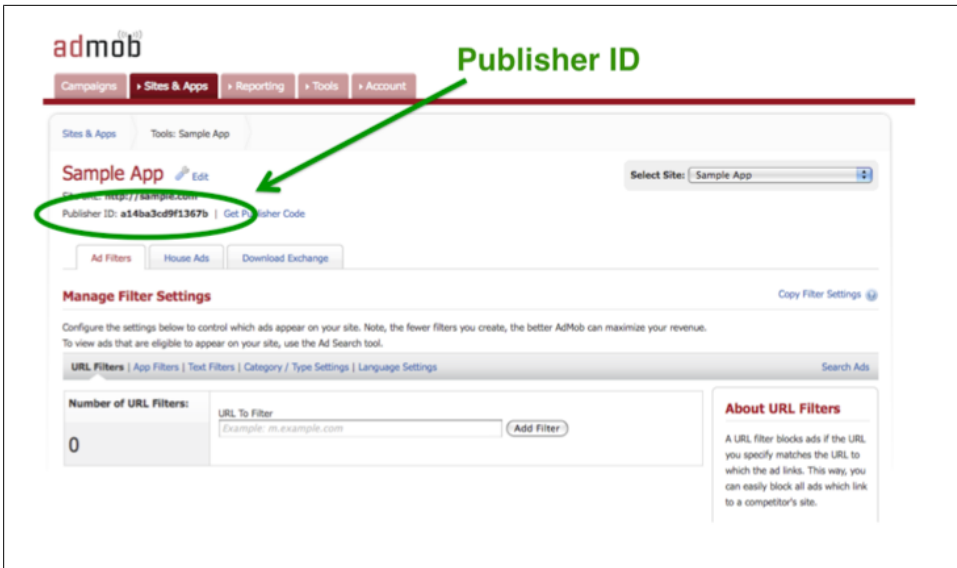


Figure 20-1.

Step 2

Add your publisher ID to your *AndroidManifest.xml*. Just before the closing `</application>` tag add a line to set your publisher ID. If your publisher ID were `149afxxxx`, the line would look like this:

Example 20-2.

```
<meta-data android:value="a149afxxxx" android:name="ADMOB_PUBLISHER_ID"/> </application>
```

To find your publisher ID, log into your AdMob account, select the **Sites and Apps** tab, and click on the **Manage Settings** link for your site. On this page, you can find your publisher ID as shows figure 1.

figure 1. Showing where you can find your Publisher ID

Step 3

Add the INTERNET permission to your *AndroidManifest.xml* just before the closing `</manifest>` tag:

Example 20-3.

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

Optionally, you can add the `ACCESS_COARSE_LOCATION` and/or `ACCESS_FINE_LOCATION` permissions to allow AdMob the ability to show geo- targeted ads.

Your final *AndroidManifest.xml* may look something like figure 2.

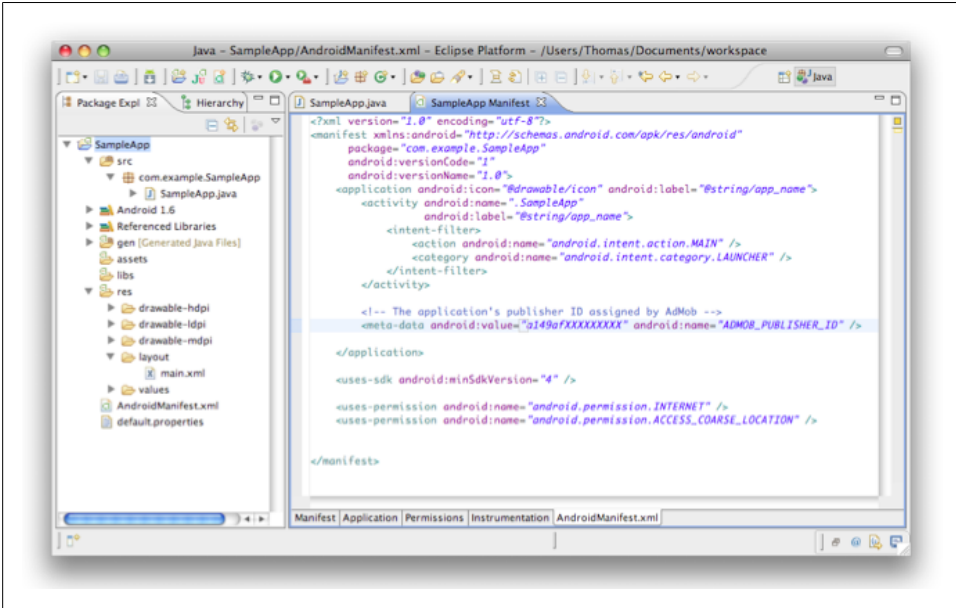


Figure 20-2.

figure 2. This is how it would look after pasting some code

Step 4

Paste the following into your `attrs.xml` file:

Example 20-4.

```

<declare-styleable name="com.admob.android.ads.AdView">
<attr name="backgroundColor" format="color" />
<attr name="primaryTextColor" format="color" />
<attr name="secondaryTextColor" format="color" />
<attr name="keywords" format="string" />
<attr name="refreshInterval" format="integer" />
</declare-styleable>

```

If your project does not already have an `attrs.xml` file, then create one in the `/res/values/` directory of your project, and paste the following:

Example 20-5.

```

<?xml version="1.0" encoding="utf-8"?> <resources>
<declare-styleable name="com.admob.android.ads.AdView">
<attr name="backgroundColor" format="color" />
<attr name="primaryTextColor" format="color" />
<attr name="secondaryTextColor" format="color" />
<attr name="keywords" format="string" />

```

```

<attr name="refreshInterval" format="integer" />
</declare-styleable>
</resources>

```

Step 5

Create a reference to the `attrs.xml` file in your layout element by adding `xmlns` line that includes your package name specified in `AndroidManifest.xml`. For example, if your package name were `com.example.sampleapp` you would include this line:

Example 20-6.

```
xmlns:myapp="http://schemas.android.com/apk/res/com.example.sampleapp"
```

So for a simple screen with only one ad, your layout element would look like this:

Example 20-7.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:myapp="http://schemas.android.com/apk/res/com.example.SampleApp"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">

<com.admob.android.ads.AdView
android:id="@+id/ad"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
myapp:backgroundColor="#000000"
myapp:primaryTextColor="#FFFFFF"
myapp:secondaryTextColor="#CCCCCC"
</LinearLayout>
/>

```

Step 6

When integrating AdMob ads into your application it is recommended to use test mode. In test mode test, ads are always returned. Test mode is enabled on a per-device basis. To enable test mode for a device, first request an ad, then look in LogCat for a line like the following:

To get test ads on the emulator use `AdManager.setTestDevices...`

Once you have the device ID you can enable test mode by calling in your main activity `AdManager.setTestDevices`:

Example 20-8.

```

AdManager.setTestDevices( new String[] { AdManager.TEST_EMULATOR, "E83D20734F72FB3108F104ABC0FFC738", //Phone
} );
}

```

Once you have successfully requested test ads, try clicking on each type of test ad to make sure it works properly from your application. The type of test ad returned is changed with `AdManager.setTestAction`. You can see the result in figure 3.

figure 3. The result of this snippet

See Also

<http://www.admob.com/> <http://androidtitlan.org/2010/09/como-agregar-publicidad-con-admob-a-tu-android-app/> <http://groups.google.com/group/admob-publisher-discuss>

20.3 Distributing Your Application via the Android Market

Zigurd Mednieks

Problem

You want to give away or sell your application via the Android Market.

Discussion

After you're satisfied that your application runs as expected on real Android devices, you're ready to upload to Android Market, Google's service for publishing and downloading Android applications. The procedure is pretty straightforward:

1. Sign up as an Android Developer (if you're not already signed up).
2. Upload your signed application.

Signing Up As an Android Developer

Go to Google's website at <http://market.android.com/publish>, and fill out the forms provided. You will be asked to:

- Use your Google account to log in (if you don't have a Google account, you can get one for free by following the Create Account link on the login page).
- Agree to the Android Market Terms of Service.
- Pay a one-time fee of \$25 (payable by credit card via Google Checkout; again, if you do't have an account set up, you can do so quickly).
- If the game is being charged for, specify your payment processor (again, you can easily sign up for a Google Payments account).

The forms ask for a minimal amount of information--your name, phone number, etc.--and you are signed up.

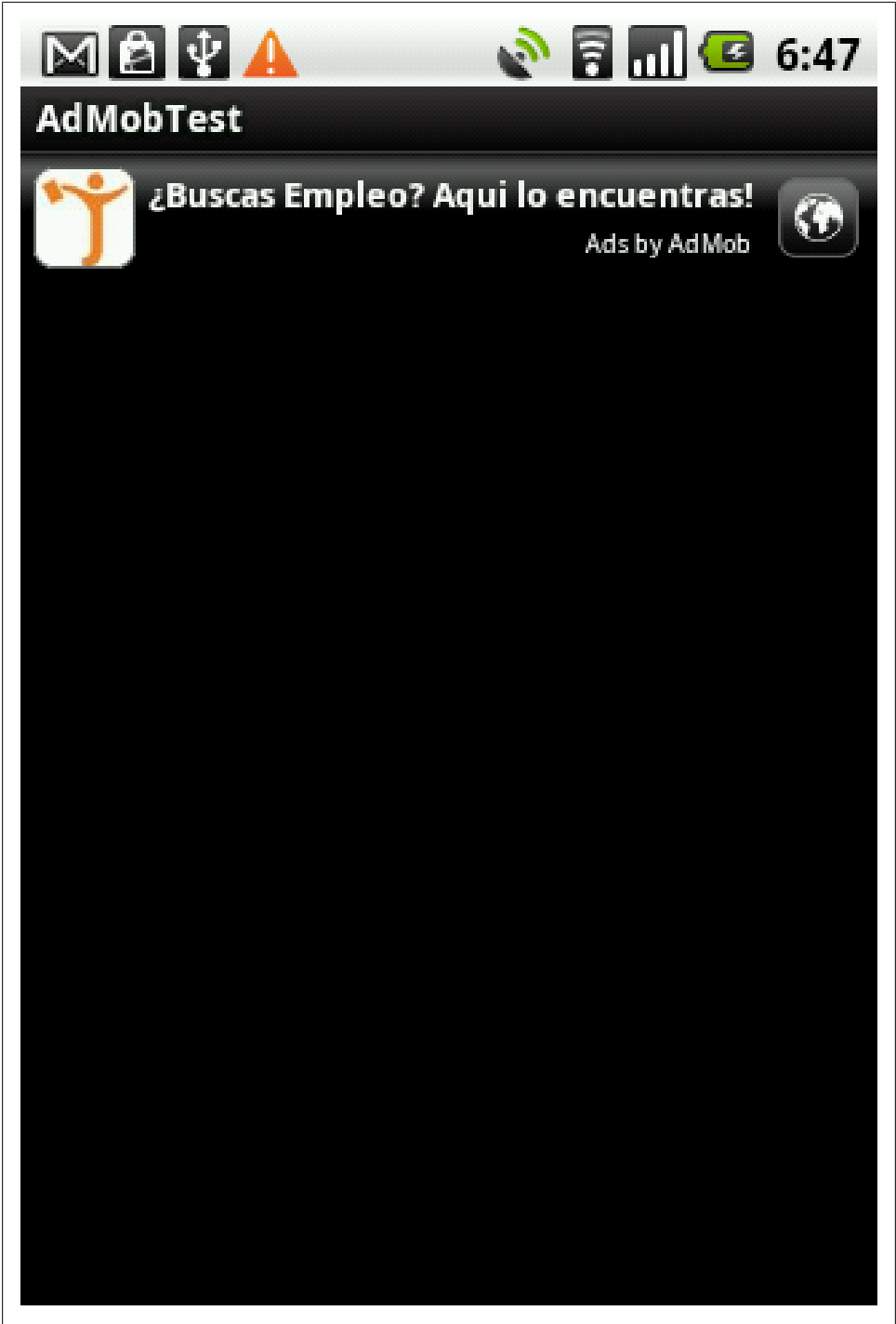


Figure 20-3.

Uploading Your Application

Now you can go to <http://market.android.com/publish/Home> to upload your application. To identify and categorize your application, you will be asked for the following:

- **Application apk file Name and Location:** The apk file of your application, signed with your private signature certificate.
- **Title and Description:** These are very important, because they are the core of your marketing message to potential users. Try to make the title descriptive and catchy at the same time, and describe the application in a way that will make your target market want to download it.
- **Application Type:** There are currently two choices: Applications or Games.
- **Category:** The allowable list of categories varies depending on Application Type. The currently available categories for Applications are: Communications, Demo, Entertainment, Finance, Lifestyle, Multimedia, News & Weather, Productivity, Reference, Shopping, Social, Software Libraries, Tools, and Travel. For Games, the currently available categories include: Arcade & Action, Brain & Puzzle, Cards & Casino, and Casual.
- **Price:** This may be "Free" or a fixed price. Refer to the agreement you agreed to above to see what percentage you actually get to keep.
- **Geography:** You can limit where your application is available, or choose to make it available everywhere.

Finally, you are asked to confirm that your application meets the Android Content Guidelines and that it does not knowingly violate any export laws. After that, you can upload your apk file, and within a few days your application will appear on the Android Market online catalog, accessible from any connected Android device. There is currently no way to access Android Market directly from your PC or Mac, so you'll have to use your Android phone to find out when your application is available for download. Use the search box in the market, or, load in the browser a file with a link of the form URL of `market://details?id=com.yourorg.yourprog`, but with your application's actual package name.

Then What?

Then, sit back and watch the fame or money - and the support emails - roll in. Be patient with end users, for they do not think as we do.

20.4 Creating a Signing Certificate

Zigurd Mednieks

Problem

You want to publish an application, and you need a "signing key" to complete the process.

Solution

Use the standard JDK tool *keytool* to generate a self-signed certificate.

Discussion

Google has stated that one of its intentions with Android was to minimize the hassle of getting applications signed. You don't have to go to a central signing authority to get a signing certificate; you can create the certificate yourself. Once you generate the certificate, you can sign your application using the *jarsigner* tool that comes with the Java JDK. Once again, you don't need to apply for or get anyone's approval. As you'll see, it's about as straightforward as signing can be.

To sign your application, you are going to create an encrypted signing certificate and use it to sign your application. You can sign every Android application you develop with the same signing certificate. You can create as many signing certificates as you want, but you really need only one for all your applications. And using one certificate for all your applications lets you do some things that you couldn't do otherwise:

- Simplify upgrades: Signing certificates are tied to the application package name, so if you change the signing certificate you use with subsequent versions of your application, you'll have to change the package name, too. Changing certificates is manageable, but messy.
- Multiple applications per process: When all your applications share the same signing certificate, they can run in the same Linux process. You can use this to separate your application into smaller modules (each one an Android application) that together make up the larger application. If you were to do that, you could update the modules separately and they could still communicate freely.
- Code/data sharing: Android lets you enable or restrict access to parts of your application based on the requester's signing certificate. If all your applications share the same certificate, it's easy for you to reuse parts of one application in another.

When you generate a key pair and certificate you'll be asked for the validity period you desire for the certificate. Although usual practice in web site development is to use one or two years, Google recommends that you set it for at least 25 years, and in fact, if you're going to use Android Market to distribute your application, it requires a validity date at least until October 22, 2033 (25 years to the day from when they opened Android Market) for your certificate.

Generating a key pair (public and private keys) and a signing certificate

To generate a pair of public/private keys, use a tool called `keytool`, which came with the Sun JDK when you installed it onto your development computer. `keytool` asks you for some information and uses that to generate the pair of keys:

- A private key that will be kept in a keystore on your computer, secured with passwords. You will use the private key to sign your application, and if you need a Map API Key for your application, you will use the MD5 fingerprint of the signing certificate to generate the Map API Key.
- A public key that Android can use to decrypt your signing certificate. You will send the public key along with your published application so that it can be made available in the runtime environment. Signing certificates are actually checked only at install time, so once installed, your application is good to run, even if the certificate or keys expire.

`keytool` is pretty straightforward. From your operating system's command line, enter something like:

Example 20-9.

```
$ keytool -genkey -v -keystore myapp.keystore -alias myapp -keyalg RSA
  -validity 10000
```

This asks `keytool` to generate a key pair and self-signed certificate (`-genkey`) in verbose mode (`-v`), so you get all the information, and put it in a keystore called `myapp.keystore` (`-keystore`). It also says that in the future you want to refer to that key by the name `myapp` (`-alias`), and that `keytool` should use the RSA algorithm for generating public/private key pairs (`-keyalg`). Finally, we say that we'd like the key to be valid for 10,000 days (`-validity`), or about 27 years.

`keytool` will prompt you for some things it uses to build the key pair and certificate:

- A password to be used in the future when you want to access the keystore
- Your first and last names
- Your organizational unit (the name for your division of your company, or something like "self" if you aren't developing for a company)
- Your organization name (the name of your company, or anything else you want to use)
- The name of your city or locality
- The name of your state or province
- The two-letter country code where you are located

`keytool` will then echo all this information back to you to make sure it's accurate, and if you confirm the information, will generate the key pair and certificate. It will then ask you for another password to use for the key itself (and give you the option of using

the same password you used for the keystore). Using that password, keytool will store the key pair and certificate in the keystore.

See Also

If you're not familiar with the algorithms used here such as RSA and MD5, well, you don't actually need to know much about them. Assuming you've a modicum of intellectual curiosity, you can find out all you need to know about them with any good web search engine.

You can get more information about security, key pairs, and the keytool utility on Sun's website at <http://java.sun.com/j2se/1.5.0/docs/tooldocs/#security>.

20.5 Obfuscating and Optimizing with ProGuard

Ian Darwin

Problem

You want to obfuscate your code, or optimize it (for speed or size), or all of the above.

Solution

The optimization and obfuscation tool ProGuard is supported by the Ant Script provided with the modern Android Project wizard in Eclipse, needing only to be enabled.

Discussion

Obfuscation of code is the process of trying to hide information (such as compile-time names visible in the binary) that would be useful in reverse-engineering your code. If your application contains commercial or trade secrets, you probably do want to obfuscate it. If your program is open source, there is probably no need to obfuscate the code. You decide.

Optimization of code is analogous to refactoring at the source level; but it usually aims to make the code either *faster*, *smaller*, or both.

The normal development cycle with Android and Eclipse involves compilation to standard Java ByteCode (done by the Eclipse Compiler) then conversion to the Android-specific DEX (Dalvik Executable) format. *ProGuard* is Eric Lafortune's open-source, free software program to optimize and obfuscate Java code. ProGuard is not Android-specific; it works with console-mode applications, Applets, Swing applications, JavaME Midlets, Android, or just about any type of Java program. ProGuard works on compiled Java, so it must be interposed in the development cycle before conversion to DEX. This is most readily achieved using the standard Java build tool Ant. The Eclipse Android Project Wizard, as of Gingerbread (2.3), includes support for

ProGuard in the generated build.xml file. You only need to edit the file `build.properties` to include the following line, which gives the name of the configuration file.

Example 20-10.

```
proguard.config=proguard.cfg
```

For older versions, please refer to the [ProGuard Reference Manual](#).

Configuration File

The ProGuard processing is controlled by the configuration file (normally called `proguard.cfg`), which has its own syntax. Basically keywords begin with a "-" character in the first character position, followed by a keyword, followed by optional parameters. Where the parameters reference Java classes or members, the syntax somewhat mimics Java syntax to make your life easier. Here is a minimal ProGuard configuration file for an Android application:

Example 20-11.

```
-injars      bin/classes
-outjars     bin/classes-processed.jar
-libraryjars /usr/local/java/android-sdk/platforms/android-9/android.jar

-dontpreverify
-repackageclasses ''
-allowaccessmodification
-optimizations !code/simplification/arithmetic

-keep public class com.example.MainActivity
```

The first section specifies the paths of your project, including a temporary directory for the optimized classes.

The next section lists various options. Preverification is only for full Java projects, so it's turned off. The optimizations shown are for a 1.5 Android and could probably be omitted today.

Finally, the class `com.example.MainActivity` has to be present in the output of the optimization and obfuscation process, since it is the main activity and is referred to by name in the `AndroidManifest`.

A full working `proguard.cfg` will normally be generated for you by the Eclipse New Android Project Wizard. Here, for example, is the configuration file generated for an Android 2.3.3 project:

Example 20-12.

```
-optimizationpasses 5
-dontusemixedcaseclassnames
-dontskipnonpubliclibraryclasses
-dontpreverify
```

```

-verbose
-optimizations !code/simplification/arithmetic,!field/*,!class/merging/*

-keep public class * extends android.app.Activity
-keep public class * extends android.app.Application
-keep public class * extends android.app.Service
-keep public class * extends android.content.BroadcastReceiver
-keep public class * extends android.content.ContentProvider
-keep public class * extends android.app.backup.BackupAgentHelper
-keep public class * extends android.preference.Preference
-keep public class com.android.vending.licensing.ILicensingService

-keepclasseswithmembernames class * {
    native <methods>;
}

-keepclasseswithmembernames class * {
    public <init>(android.content.Context, android.util.AttributeSet);
}

-keepclasseswithmembernames class * {
    public <init>(android.content.Context, android.util.AttributeSet, int);
}

-keepclassmembers enum * {
    public static **[] values();
    public static ** valueOf(java.lang.String);
}

-keep class * implements android.os.Parcelable {
    public static final android.os.Parcelable$Creator *;
}

```

The prolog is mostly similar to the earlier example. The `keep`, `keepclasseswithmembernames`, and `keepclassmembers` specify particular classes that must be retained. These are mostly obvious, but the `enum` entries may not be: the Java 5 `enum` methods `values()` and `valueOf()` are sometimes used with the Reflection API, so they must remain visible, as must any classes that you access via the Reflection API.

The `ILicensingService` entry is only needed if you are using Android's License Validation Tool (LVT):

Example 20-13.

```
-keep class com.android.vending.licensing.ILicensingService
```

See Also

The *ProGuard Reference Manual* has many more details. There is also information at [Google's Developer Site](#). Finally, Matt Quigley has an article at [AndroidEngineering](#) entitled *Optimizing, Obfuscating, and Shrinking your Android Applications with ProGuard*.

20.6 Provide a Link to other Published Apps in the Market

Daniel Fowler

Problem

Your developed App is running on a device; you want a link to your other Apps on the Android Market to encourage users to try them.

Solution

Use an Intent and URI that contains your Publisher Name or Package Name.

Discussion

Android's `Intent` system is a great way for your application to leverage functionality that has already been written by other developers. The **Android Market** application, which is used to browse and install Apps, can be called from an application by using an Intent. This allows an existing App to have a link to other Apps on the Android Market, thus allowing an App developer or publisher to encourage users to try their other Apps.

To search via the Android Market App the standard Intent mechanism is used, as described in the recipe [Recipe 1.8](#). The Uniform Resource Identifier (URI) used is **market://search?q=search term** where *search term* is replaced by the appropriate text, such as the program name or keyword. The Intent Action is `ACTION_VIEW`.

The URI can also point directly to the Android Market details page for a package by using **market://details?id=package name** where *package name* is replaced by the unique Package Name for the App.

The program shown here will allow a text search of the Android Market or show the details page for a given App. Here is the layout:

Example 20-14.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <EditText android:id="@+id/etSearch"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:singleLine="true"/>
    <RadioGroup android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <RadioButton android:id="@+id/rdSearch"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:checked="true"
```

```

        android:text="search"
        android:textSize="20sp"/>
<RadioButton android:id="@+id/rdDetails"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="details"
    android:textSize="20sp"/>
</RadioGroup>
<Button android:id="@+id/butSearch"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:text="Search Android Market"/>
</LinearLayout>

```

An `EditText` allows entry of the search term, a `RadioButton` can be used to do a straight search or show an Apps details page (provided the full Package Name is known). The `Button` starts the search.

The important point to notice in the code is that the search term is encoded.

Example 20-15.

```

public class main extends Activity {
    RadioButton publisherOption;    //Option for straight search or details
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //Search button press processed by inner class HandleClick
        findViewById(R.id.butSearch).setOnClickListener(new OnClickListener(){
            public void onClick(View arg0) {
                String searchText;
                //Reference search input
                EditText searchFor=(EditText)findViewById(R.id.etSearch);
                try {
                    //URL encoding handles spaces and punctuation in search term
                    searchText = URLEncoder.encode(searchFor.getText().toString(),"UTF-8");
                } catch (UnsupportedEncodingException e) {
                    searchText = searchFor.getText().toString();
                }
                Uri uri; //Stores intent URI
                //Get search option
                RadioButton searchOption=(RadioButton)findViewById(R.id.rdSearch);
                if(searchOption.isChecked()) {
                    uri=Uri.parse("market://search?q=" + searchText);
                } else {
                    uri=Uri.parse("market://details?id=" + searchText);
                }
                Intent intent = new Intent(Intent.ACTION_VIEW, uri);
                try {
                    main.this.startActivity(intent);
                } catch (ActivityNotFoundException anfe) {
                    Toast.makeText(main.this, "Please install the Android Market App", Toast.LENGTH_SHORT);
                }
            }
        });
    }
}

```

```
    }  
  });  
}  
}
```

A straight text search is simply the text appended to the URI `market://search?q=`. To search by publisher name use the **pub:** qualifier, i.e. append the publishers name to `market://search?q=pub:`. **However**, at the time of writing a bug exists in some versions of Android Market that causes publisher names of more than one word to return no results. So whilst `market://search?q=pub:IMDb` works, `market://search?q=pub:O'Reilly+Media` does not. The work around is to use the straight text search for publisher names of two words or more. For example `market://search?q=oreilly+media`.

The `pub:` search qualifier is also case sensitive, thus `market://search?q=pub:IMDb` returns a result but `market://search?q=pub:imdb` does not.

It is also possible to search for a specific application if the Package Name is known by using the **id** qualifier. So if a App has Package Name of **com.example.myapp** the search term will be `market://search?q=id:com.example.myapp`. Even better go straight to the Apps details page with `market://details?q=id:com.example.myapp`. For example O'Reilly has a free App the details of which can be show using `market://details?id=com.aldiko.android.oreilly.isbn9781449388294`.

Using the above techniques it is very easy to put a button or menu option on a screen to allow users to go directly to other Apps that you have published.

See Also

<http://developer.android.com/guide/publishing/publishing.html#marketintent>

21.1 Introduction: Everything Else

Ian Darwin

Discussion

This chapter is a catch-all for Recipes that don't fit neatly into the twenty pigeonholes called "Chapters" that make up the rest of the book.

If you have read this far and still seek more examples of complete programs, here are some examples.

- [AndNav2 - OpenStreetmap-based navigation](#)
- [Replica Island](#) game

21.2 Sending messages between threads using activity thread queue and Handler class

Vladimir Kroz

Problem

Notify activity with information posted by another thread.

Solution

- Implement Handler class, override method `handleMessage()` which will read messages from thread queue
- In worker thread - post message using `sendMessage()` method

Discussion

There are many situations when it is required to have a thread running in the background and send information to main Activity's UI thread. At the architectural level there are different approaches can be taken:

1. Use of Android AsyncTask class
2. Start a new thread

Though using AsyncTask is very convenient, there are a situations when you really need to construct a worker thread by yourself. In such situation you likely will need to send some information back to Activity thread. Keep in mind that Android doesn't allow other threads to modify any content of main UI thread. Instead you're required to wrap data into messages and send them through message queue. Implementation consists of two parts.

1) Add handler

Add an instance of Handler class to your MapActivity instance.

Example 21-1.

```
public class MyMap extends MapActivity {
    . . . . .
    public Handler _handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            Log.d(TAG, String.format("Handler.handleMessage(): msg=%s", msg));
            // This is where main activity thread receives messages
            // Put here your handling of incoming messages posted by other threads
            super.handleMessage(msg);
        }
    };
    . . . . .
}
```

2) Post Message

In the worker thread post a message to activity main queue whenever you need Add handler class instance to your MapActivity instance.

Example 21-2.

```
/**
 * Perfomes background job
 */
class MyThreadRunner implements Runnable {
    // @Override
    public void run() {
        while (!Thread.currentThread().isInterrupted()) {
```

```

// Just dummy message -- real implementation will put some meaningful data in it
    Message msg = Message.obtain();
    msg.what = 999;
    MyMap.this._handler.sendMessage(msg);
// Dummy code to simulate delay while working with remote server
    try {
        Thread.sleep(5000);
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
    }
}
}
}

```

21.3 Intercommunication amongst Applications

Rachee Singh

Problem

You need to get different applications deployed on a device to communicate. For instance, the running application wishes to notify another application (which is in the background) of a certain event.

Solution

To initiate communication, an application broadcasts an intent. The applications in the background that register a receiver for that particular intent will receive it and perform the necessary action in their BroadcastReceiver subclass.

Discussion

To broadcast an intent from your application, insert the following code:

Example 21-3.

```

Intent intent = new Intent("com.bluetooth.BLUE_LINK_BROKEN");
sendBroadcast(intent);
Log.d("Bluetooth.java", "Intent Sent");

```

The application which wishes to be notified of an action from your application, needs the following code:

A separate class which extends BroadcastReceiver. In this class, the application being notified can carry out the tasks like encrypting data, password protecting so on and so forth.

Example 21-4.

```

public class BluelinkIntents extends BroadcastReceiver {

```



```

@Override
public void onReceive(Context context, Intent intent) {
    Toast.makeText(context, "Application X Received the Intent!", oast.LENGTH_SHORT).show();
    Log.d("OI Safe", "Received the Intent!");
}
}

```

These few lines of code have to be added to the AndroidManifest of the application which is being notified.

Example 21-5.

```

<!-- Receiver for Intent -->
    <receiver android:name="org.openintents.intents.BluelinkIntents" android:enabled="true" >
        <intent-filter>
            <action android:name="com.bluetooth.BLUE_LINK_BROKEN" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </receiver>

```

With all this set up, when your application starts, it will broadcast an intent. All the other applications present in the background on that Android device and which have the BroadcastReceiver set up will receive the intent and display a toast 'Application X Received the Intent!'.

Contributors

22.1 Names

The following will contain a list of the contributors to this book.

About the Author

Colophon

Our look is the result of reader comments, our own experimentation, and feedback from distribution channels. Distinctive covers complement our distinctive approach to technical topics, breathing personality and life into potentially dry subjects.

The animal on the cover of *Android Cookbook* is FILL IN DESCRIPTION.

The cover image is FILL IN CREDITS. The cover font is Adobe ITC Garamond. The text font is Linotype Birka; the heading font is Adobe Myriad Condensed; and the code font is LucasFont's TheSans Mono Condensed.

