

# Head First Android Development

A Brain-Friendly Guide



Design top-selling apps



Avoid embarrassing activities



See how Material Design can change your life



Master out-of-this-world concepts

Tap into the Android Location Service



Pool around in the Android Support Libraries



Dawn Griffiths & David Griffiths

# Android Development

## What will you learn from this book?

If you have an idea for a killer Android app, this book will help you build your first working application in a jiffy. You'll learn hands-on how to structure your app, design interfaces, create a database, make your app work on various smartphones and tablets, and much more. It's like having an experienced Android developer sitting right next to you! All you need is some Java know-how to get started.



## Why does this book look so different?

Based on the latest research in cognitive science and learning theory, *Head First Android Development* uses a visually rich format to engage your mind, rather than a text-heavy approach that puts you to sleep. Why waste your time struggling with new concepts? This multi-sensory learning experience is designed for the way your brain really works.

"A comprehensive beginner's guide to Android development, easy to read and full of excellent examples and exercises."

—Edward Yue Shang Wong  
(*Workangelofham*)

"This is, without a doubt, the best available book for learning Android development. If you can get only one, make it this one."

—Kenneth Krasner  
President, Krasner IT Inc.,  
and JavaDoc Book Star

"Become an able Android developer applying up-to-date patterns and create that next killer app. *Head First Android Development* will be your friendly, accurate, and fun-to-be-with master craftsman on that path."

—Ingo Knezly  
*Android Learner*

Programming / Android

US \$49.99

CAN \$57.99

ISBN: 978-1-449-36218-8



9 781449 362188



oreilly.com

---

# Head First Android Development

Wouldn't it be dreamy if there were a book on developing Android apps that was easier to understand than the space shuttle flight manual? I guess it's just a fantasy...



Dawn Griffiths  
David Griffiths

**O'REILLY**<sup>®</sup>

Beijing ■ Cambridge ■ Köln ■ Sebastopol ■ Tokyo

# Head First Android Development

by David Griffiths and David Golub

Copyright © 2011, David Golub, David Griffiths. All rights reserved.

Printed in the United States of America

Publisher: by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95971

O'Reilly Media has arranged to purchase the non-exclusive, worldwide, print, online, and audio rights available for this title (<http://my.safaribooksonline.com>). For more information, contact our copyright/information sales department: (800) 998-9938 or [copyright@oreilly.com](mailto:copyright@oreilly.com).

<b>Series Creators:</b>	Katie Sierra, Ben Jones
<b>Editor:</b>	Meghan Blasey-ette
<b>Cover Designer:</b>	Karen Montgomery
<b>Production Editor:</b>	Marie Wotoczko
<b>Production Services:</b>	Jessica Keaton
<b>Indexer:</b>	Rob Palfrey
<b>Page Viewers:</b>	Marianne Dakin-Graf

## Printing History:

June 2011: First Edition

Mum and Dad →



← Rob and Lorraine

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. The *Head First* series name, *Head First Android Development*, 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 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 the author assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

No kitchen was harmed in the making of this book, but several pizzas were eaten.

ISBN: 978-1-449-32188-3

[TS]

[978-1-449-32188-3]

---

To our friends and family. Thank you so  
much for all your love and support.



## Authors of Head First Android Development



Dawn Griffiths



David Griffiths

**Dawn Griffiths** started Hf as a mathematician at a top UK university when she was awarded a first-class honors degree in mathematics. She went on to pursue a career in software development and has 20 years' experience working in the IT industry.

Before writing *Head First Android Development*, Dawn wrote three other *Head First* books: *Head First Servlets*, *Head First J2Ee Concepts*, and *Head First Oracle*, has also worked on a host of other books in the series.

When Dawn is not working on *Head First*, books, you'll find her teaching her Tai Chi skills, reading, running, rucking, golfing, Lego, or rucking. She particularly enjoys spending time with her gentle-fell husband, David.

**David Griffiths** says he remembers being 12, when he saw a documentary on the work of Seymour Papert. At age 13, he wrote an implementation of Papert's educational language LOGO. After studying some mathematics during college, he began writing computer programs and magazines articles for fun. He's worked as a mobile developer and a garage attendant, but he's a full-time *Head First* author. He can write code in over 10 languages and knows, to just name one, when not writing, coding or teaching, he spends much of his spare time traveling with his lovely wife—total excitement—Dawn.

Before writing *Head First Android Development*, David wrote a number of other *Head First* books: *Head First Rails*, *Head First Programming*, and *Head First C*.

You can follow him on Twitter at <http://twitter.com/HeadFirstDev>.

# Table of Contents (Summary)

	Intro	xxiii
1	Getting Started: <i>Diving in</i>	1
2	Building Interactive Apps: <i>Apps that do something</i>	39
3	Multiple Activities and Intents: <i>State your intent</i>	73
4	The Activity Lifecycle: <i>Being an activity</i>	115
5	The User Interface: <i>Enjoy the view</i>	163
6	List Views and Adapters: <i>Getting organized</i>	227
7	Fragments: <i>Make it modular</i>	269
8	Nested Fragments: <i>Dealing with children</i>	325
9	Action Bars: <i>Taking shortcuts</i>	365
10	Navigation Drawers: <i>Going places</i>	397
11	SQLite Databases: <i>Fire up the database</i>	437
12	Cursors and AsyncTasks: <i>Connecting to databases</i>	471
13	Services: <i>At your service</i>	541
14	Material Design: <i>Living in a material world</i>	597
i	ART: <i>The Android Runtime</i>	649
ii	ADB: <i>The Android Debug Bridge</i>	653
iii	The Emulator: <i>The Android Emulator</i>	659
iv	Leftovers: <i>The top ten things (we didn't cover)</i>	663

# Table of Contents (the real thing)

## Intro

**Your brain on Android.** Here *you* are trying to *learn* something, while here your *brain* is, doing you a favor by making sure the learning doesn't *stick*. Your brain's thinking, "Better leave room for more important things, like which wild animals to avoid and whether naked snowboarding is a bad idea." So how *do* you trick your brain into thinking that your life depends on knowing how to develop Android apps?

	Who is this book for?	xxiv
	We know what you're thinking	xxv
	We know what your <i>brain</i> is thinking	xxv
	Metacognition: thinking about thinking	xxvii
	Here's what WE did:	xxviii
	Read me	xxx
	The technical review team	xxxii
	Acknowledgments	xxxiii

getting started

1

Diving In

Android has been taking the world by storm.

Everybody wants a smart phone or tablet, and Android devices are hugely popular. In this book we'll teach you how to develop your own apps, and we'll start by getting you to build a basic app and run it on an Android Virtual Device. Along the way you'll meet some of the basic components of all Android apps such as activities and layouts

All you need is a little Java know-how...



Welcome to Android development	2
The Android platform directory	3
Your development environment	4
Tools	6
Build a task app	7
Activities and layouts from HelloWorld	11
Building a basic app (continued)	13
Building a basic app (continued)	14
We've just created our first Android app	15
Android Studio: the development IDE for Android	16
Using Files in your project	17
Edit metadata in the Android Studio editor	18
Run the app in the Android emulator	19
Creating an Android Virtual Device	21
Run the app in the emulator	22
You can watch progress in the console	24
Run, draw	28
What, just a page of code?	31
Refining the app	31
What's in a package?	32
Activity: the main class that contains elements	33
The layout file: contains the framework's view objects to draw on the screen	34
Let's look at the using.xml file	34
Take the app for a test drive	34
Your Android Tutorial	34



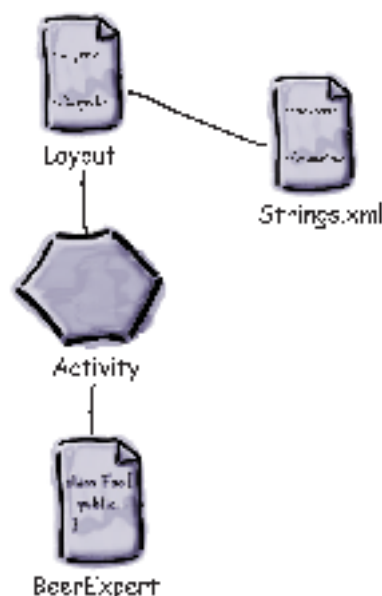


## building interactive apps

**Apps That Do Something****2**

**Most apps need to respond to the user in some way.**

In this chapter you'll see *how* you can make your apps a *bit more interactive*. You'll see *how* you can get your app to *do* something in response to the user, and *how to get your activity and layout talking to each other* like best buddies. Along the way we'll take you a *bit deeper* into *how Android actually works* by introducing you to R, the hidden gem that glues everything together.



You're going to build a beer app for a sp	4
Create the project	12
We've created a default activity and layout	17
Adding capabilities with the <code>android:usesCleare</code>	24
with <code>android:usesCleare</code> to a new button	27
Changes to the XML...	30
...are reflected in the design editor	34
Examining resources in <code>res/layout</code> for text	36
Change the layout to use the string resources	37
To make the app do some thing	37
add values to the spinner	37
Get the spinner to reference a string array	41
Test drive the spinner	54
We need to make the button do something	54
What activity code looks like	57
What activity code looks like	57
Add an <code>onOptionsItemSelected()</code> method to the activity	57
<code>onOptionsItemSelected()</code> needs to do some thing	58
Once you have a <code>View</code> , you can access its methods	58
Update the activity code	60
The first section of the activity	62
Test drive the changes	62
Building the custom layout class	63
Generate the activity or call the <code>findViewById</code> class so that we can get REAL advice	67
with your <code>findViewById</code>	67
What happens when you run the code	70
Test drive your app	71
Your Android Toolbox	72

## multiple activities and intents

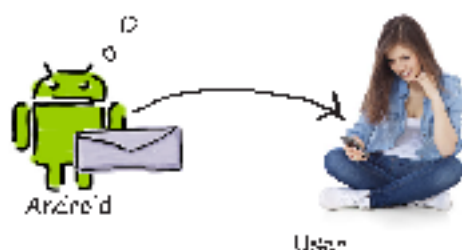
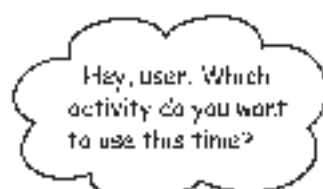
## State Your Intent

## 3

## Most apps need more than one activity.

So far we've just looked at single activity apps, which is fine for simple apps. But when things get more complicated, just having the one activity won't cut it. We're going to show you how to **build apps with multiple activities**, and how you can get your apps talking to each other using **intents**. We'll also look at how you can use intents to go **beyond the boundaries of your app and make activities in other apps on your device perform actions**. Things just got a whole lot more powerful.

Apps can contain more than one activity	7
How do the apps exist on	25
Create the project	27
Create the main activity and layout	28
Welcome to the Android world! Let	87
Use an intent to start the second activity	83
What happens when you start the app?	9
Test drive the app	37
But text is a terrible activity	38
Update the app view properties	86
Just a single activity is not a transition in an intent	82
Update the CreateMessage activity code	91
Get ReceiveMessagesActivity to use the information in the intent	97
What happens when the user clicks on Send Message button?	93
Test drive the app	95
How Android apps work	10
What happens when the user clicks	96
How Android uses the intent filter	107
You need to run your app on a REAL device	103
Test drive the app	102
Change the code to create a chooser	1
Test drive the app	10
Your Android app lives	111

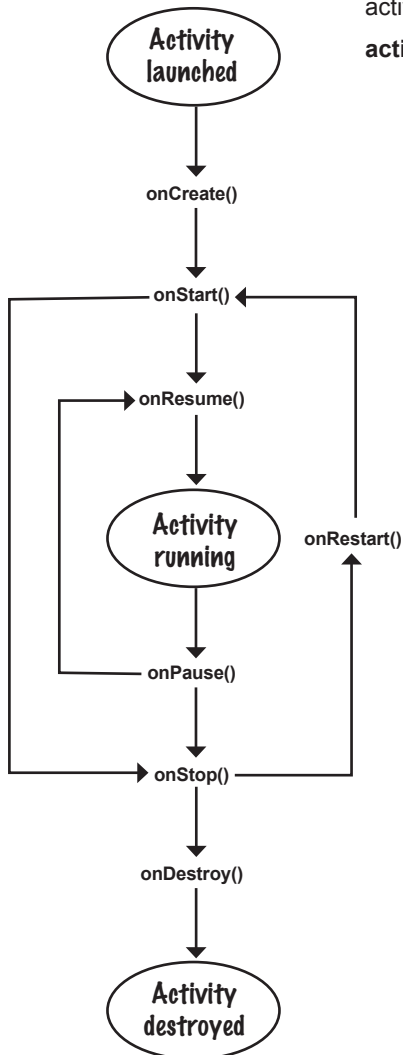


## the activity lifecycle

## 4

**Do one thing and do it well****Activities form the foundation of every Android app.**

So far you've seen how to create activities, and made one activity start another using an intent. But *what's really going on beneath the hood?* In this chapter we're going to dig a little deeper into **the activity lifecycle**. What happens when an activity is **created** and **destroyed**? Which methods get called when an activity is *made visible and appears in the foreground*, and which get called when the activity *loses the focus and is hidden*? And **how do you save and restore your activity's state**?



How do activities really work?	116
The Stopwatch app	118
The stopwatch layout code	119
Add code for the buttons	122
The runTimer() method	123
Handlers allow you to schedule code	124
The full runTimer() code	125
The full StopwatchActivity code	126
Rotating the screen changes the device configuration	132
From birth to death: the states of an activity	133
The activity lifecycle: from create to destroy	134
How do we deal with configuration changes?	136
What happens when you run the app	139
There's more to an activity's life than create and destroy	142
The activity lifecycle: the visible lifetime	143
The updated StopwatchActivity code	147
What happens when you run the app	148
Test drive the app	149
But what if an app is only partially visible?	150
The activity lifecycle: the foreground lifetime	151
Stop the stopwatch if the activity's paused	154
The complete activity code	157
Your handy guide to the lifecycle methods	161
Your Android Toolbox	162

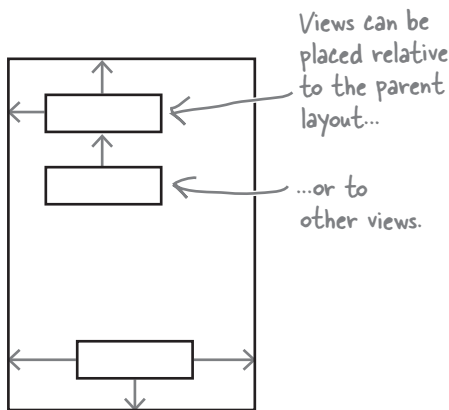
# the user interface

## 5

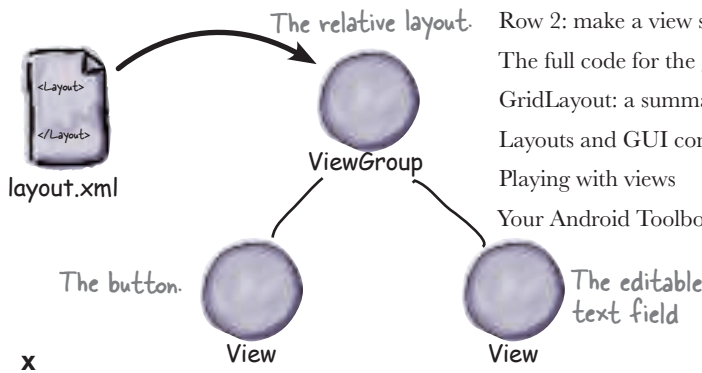
### Enjoy the View

Let's face it, you need to know how to create great layouts.

If you're building apps you want people to *use*, you need to make sure they **look just the way you want**. So far we've only scratched the surface when it comes to creating layouts, so it's time to *look a little deeper*. We'll introduce you to more **types of layout** you can use, and we'll also take you on a tour of the **main GUI components** and *how you use them*. By the end of the chapter you'll see that even though they all look a little different, all layouts and GUI components have **more in common than you might think**.



Three key layouts: relative, linear, and grid	165
Positioning views relative to the parent layout	168
Positioning views relative to other views	170
Attributes for positioning views relative to other views	171
RelativeLayout: a summary	173
LinearLayout displays views in a single row or column	174
Let's change up a basic linear layout	176
Adding weight to one view	179
Adding weight to multiple views	180
Using the android:gravity attribute: a list of values	182
More values you can use with the android:layout_gravity attribute	184
The full linear layout code	185
LinearLayout: a summary	186
GridLayout displays views in a grid	189
Adding views to the grid layout	190
Let's create a new grid layout	191
Row 0: add views to specific rows and columns	193
Row 1: make a view span multiple columns	194
Row 2: make a view span multiple columns	195
The full code for the grid layout	196
GridLayout: a summary	197
Layouts and GUI components have a lot in common	201
Playing with views	205
Your Android Toolbox	225



## list views and adapters

## Getting Organized

## 6

## Want to know how best to structure your Android app?

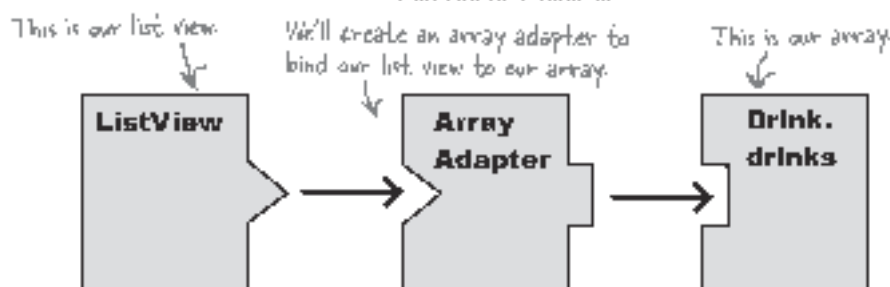
You've learned about some of the basic building blocks that are used to build apps, and now it's time to get organized. In this chapter we'll show you how you can take a bunch of ideas and *structure them into an awesome app*. We'll show you how lists of data can form the core part of your app design, and how linking them together can create a powerful and easy-to-use app. Along the way, you'll get your first glimpse of using event listeners and adapters to make your app more dynamic.

Display a start screen with a list of options.

Display a list of the drinks we sell.

Show details of each drink.

Using a queue of ideas	299
Categorize your ideas: two levels, category and local/edin activities	300
Navigating through the activities	310
Use ListView to associate data	351
What we're going to build: the Starbucks app	359
The drink details activity	383
The Starbucks app structure	384
The top-level class: MainActivity and what it does	388
The full top-level layout code	397
Get ListView to respond to clicks with a TextView	411
The full DrinkListActivity code	413
How to create a list activity	415
Connect the views to arrays with an array adapter	451
Adapt the array adapter to DrinkCategoryActivity	452
What happens when you touch a row	463
How we handle the click of the placeholder	456
The full DrinkCategoryActivity code	458
A detail activity displays duration, since rounded	455
Update the views with the data	461
The DrinkView activity	463
Test drive the app	465
Your Android Toolbox	468



fragments

7

**Make it Modular**

**You've seen how to create apps that work in the same way irrespective of the device they're running on.**

But what if you want your app to *look and behave differently* depending on whether it's running on a *phone* or a *tablet*? In this chapter we'll show you how to make your app choose the **most appropriate layout for the device screen size**. We'll also introduce you to **fragments**, a way of creating *modular code components* that can be *reused by different activities*.

The Workout app structure	273
The Workout class	275
How to add a fragment to your project	276
What fragment code looks like	278
Activity states revisited	282
The fragment lifecycle	283
Your fragment inherits the lifecycle methods	284
Test drive the app	286
How to create a list fragment	290
The updated WorkoutListFragment code	292
Test drive the app	294
Wiring up the list to the detail	295
Using fragment transactions	301
The updated MainActivity code	302
Test drive the app	303
The WorkoutDetailFragment code	305
The phone and tablet app structures	307
The different folder options	309
The MainActivity phone layout	315
The full DetailActivity code	319
The revised MainActivity code	321
Test drive the app	322
Your Android Toolbox	323

So the fragment will contain just a single list. I wonder... When we wanted to use an activity that contained a single list, we used a ListActivity. Is there something similar for fragments?





## nested fragments

## 8

**Dealing with Children**

**You've seen how using fragments in activities allow you to reuse code and make your apps more flexible.**

In this chapter we're going to show you how to nest one fragment inside another. You'll see how to use the child fragment manager to take care of fragment transactions. Along the way you'll see why knowing the differences between activities and fragments is so important.

Creating nested fragments	326
The <code>FragmentManager</code> class	329
The <code>FragmentManager.Fragment</code> class	333
<code>FragmentManager</code> creates transactions at the activity level	340
Nested fragments need nested transactions	341
The <code>FragmentManager</code> class	343
The <code>FragmentManager</code> class	344
Why does the app crash if you press a button?	345
Let's look at the <code>FragmentManager</code> again, code	346
Make the <code>FragmentManager</code> implement <code>OnClickListener</code>	349
Attach the <code>OnClickListener</code> to the buttons	351
The <code>FragmentManager</code> class	352
The <code>FragmentManager</code> class	353
The <code>FragmentManager</code> class	354
Run, draw the app	355
Your Android, to draw	356



## action bars

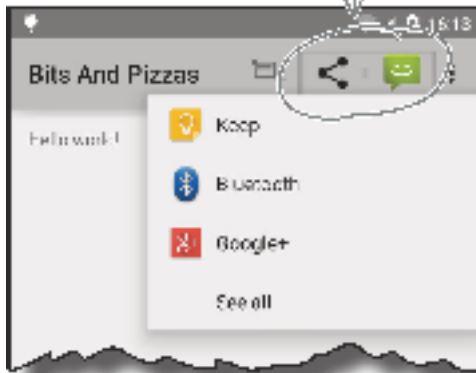
### Taking Shortcuts

# 9

#### Everybody likes a shortcut.

And in this chapter you'll see how to add shortcuts to your apps using **action bars**. We'll show you how to start other activities by adding **action items** to your action bar, how to share content with other apps using the **share action provider**, and how to navigate up your app's hierarchy by implementing **the action bar's Up button**. Along the way you'll see how to give your app a consistent look and feel using **themes** and introduce you to the **Android support library package**.

This is what the share action looks like on the action bar. When you click on it, it gives you a list of apps to share content using.



Great apps have a clear structure	366
Different types of navigation	367
Let's start with the action bar	368
The Android support libraries	369
Next, pick a color and a shape for the bar	370
We'll get the app to use up to three themes	371
Apply a theme in AndroidManifest.xml	372
Define styles in style resources files	373
Set the default theme using a style	374
What happens when you click the app	375
Adding action items to the action bar	376
The main resource file	377
The main class in AndroidManifest.xml	378
Add a new class, onCreate	379
Create OrderActivity	380
Start OrderActivity with the Create Order action item	381
The onCreate method in onCreate code	382
Showing content on the action bar	383
Specify the content with an intent	384
The onCreate method in onCreate code	385
Enabling Up navigation	386
Setting an activity's parent	387
Adding the Up button	388
Test drive the app	389
Next, Android Studio	390

API 21? A perfect match.

xiv



Android



Name: AppTheme  
Parent: Theme.Material.Light

## navigation drawers

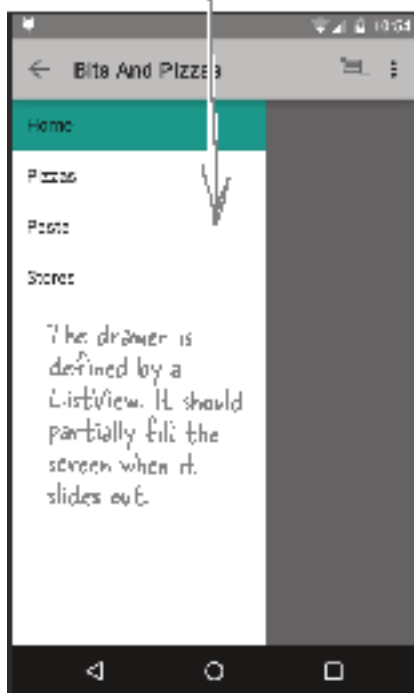
## 10

## Going Places

Apps are so much better when they're easy to navigate.

In this chapter we're going to introduce you to the navigation drawer, a slide-out panel that appears when you swipe the screen with your finger or click an icon on the action bar. We'll show you how to use it to display a *list of links* that take you to **all the major hubs** of your app. You'll also see how *switching fragments* makes those hubs **easy to get to** and **fast to display**.

The content goes in a `FrameLayout`. You want the content to fill the screen. At the moment it's partially hidden by the drawer.



The <code>Pizza</code> app revealed	338
Navigation drawer: the structure	339
The <code>Pizza</code> app structure	340
Create <code>TopFragment</code>	341
Create <code>PizzaFragment</code>	342
Create <code>PostsFragment</code>	343
Create <code>StoresFragment</code>	344
Add the <code>DrawerLayout</code>	345
Use <code>findViewById</code> for activity initialization	346
Initialize the drawer's <code>List</code>	347
Changing the drawer's <code>List</code> items	348
Changing the navigation drawer	349
The updated <code>MainActivity.java</code> code	351
Using an <code>ActionBar.DrawerToggle</code>	352
Modifying <code>onOptionsItemSelected</code>	353
The updated <code>MainActivity.java</code> code	354
Providing the <code>DrawerToggle</code> class	356
Synchronizing the <code>ActionBar.DrawerToggle</code> 's state	357
The updated <code>MainActivity.java</code> code	358
Dealing with configuration changes	359
Reacting to changes on the back stack	360
Adding <code>onClick</code> to <code>DrawerToggle</code>	361
The full <code>MainActivity.java</code> code	362
The <code>DrawerToggle</code> class	363
Your Android toolbox	386

## SQLite databases

## Fire up the Database

## 11

If you're recording high scores or saving tweets, your app will need to store data. And on Android you usually keep your data safe inside a **SQLite database**. In this chapter, we'll show you how to *create a database*, *add tables to it*, and *populate it with data*, all with the help of the friendly **SQLite helper**. You'll then see how you can cleanly roll out *upgrades* to your database structure, and *how to downgrade* if you need to pull any changes



SQLite database

Name: "storbuzz"  
Version: 1

Back to Storbuzz	424
Android uses SQLite databases to persist data	425
Android comes with SQLite classes	430
The onCreate() method as a presentation	431
The SQLite helper manages your database	433
The SQLite helper	443
Create the SQLite helper	444
To store a SQLite database	446
You can create tables using Struct and Query's language (SQL)	447
Insert data using the insert() method	448
Update records with the update() method	449
Backup commands	450
The SQLiteDatabase class: The get code	451
What if the SQLite helper code fails	452
What if you need to change the database?	455
SQLite databases have a version number	456
Upgrading the database: an overview	457
How the SQLite helper manages updates	458
Upgrade your database with onUpgrade()	460
Downgrade your database with onDowngrade()	461
Let's upgrade the database	462
Upgrading an existing database	463
Recording scores	464
The full SQLite helper code	467
The SQLite helper once instantiated	468
What happens when the code runs	469
Your Android Review	470

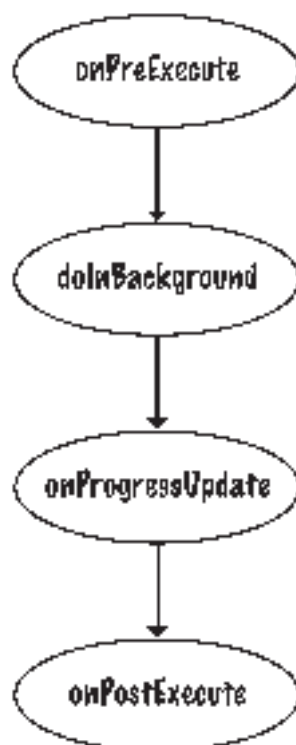
# 12

## cursors and AsyncTask

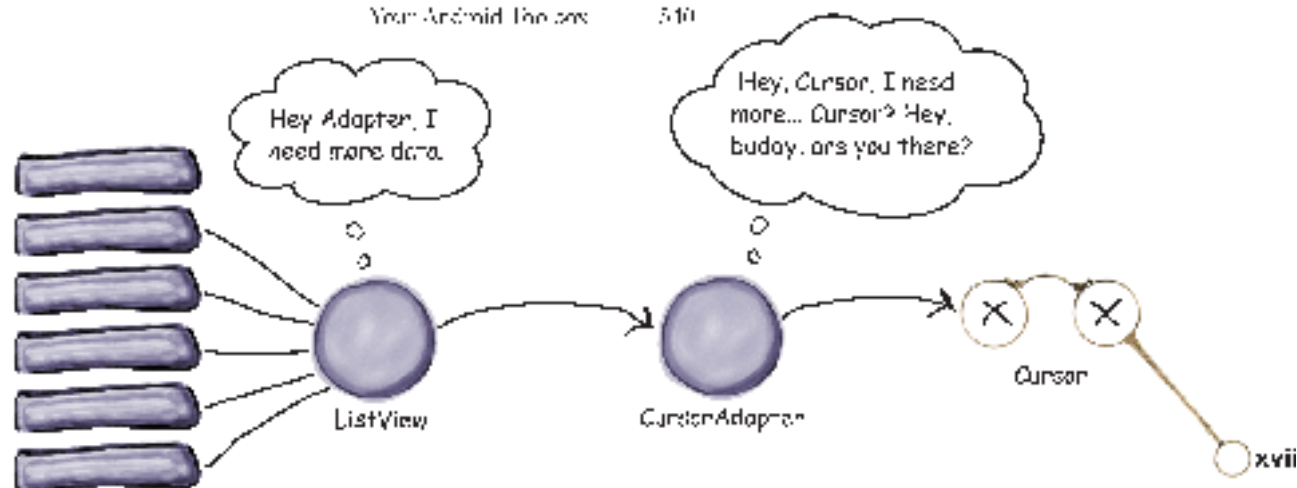
### Connecting to Databases

#### So how do you connect your app to a SQLite database?

So far you've seen how to create a SQLite database using a SQLite helper. The next step is to get your activities to access it. In this chapter you'll find out how to use *cursors* to get data from the database, how to navigate cursors and how to get data from them. You'll then find out how to use *cursor adapters* to connect them to list views. Finally, you'll see how writing efficient multi-threaded code with *AsyncTasks* will keep your app speedy.



- The current DrinkActivity code 474
- Specifying table and columns 474
- Applying multiple conditions to your query 479
- Using SQL functions in queries 481
- Navigating cursors 492
- The DrinkActivity code 490
- Adding data to DrinkActivity 504
- The DrinkActivity code 513
- The new top-down activity code 513
- The revised TapFeverActivity.java code 521
- The onInBackground() method 521
- The doInBackground() method 523
- The onPostExecute() method 524
- The AsyncTask class 525
- The DrinkActivity.java code 527
- Your Android IDE box 540



## 13

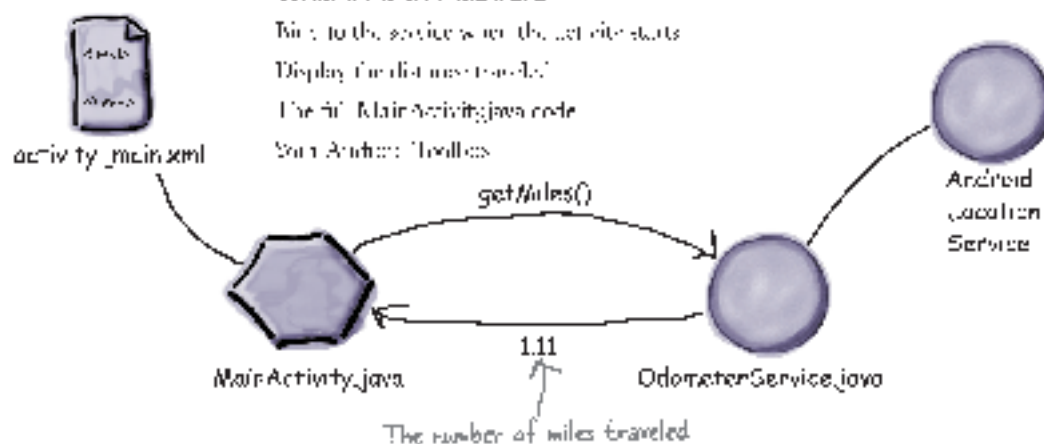
## services

**At Your Service**

**There are some operations you want to keep on running, irrespective of which app has the focus.**

As an example, if you start playing a music file in a music app, you'd probably expect it to keep on playing when you switch to another app. In this chapter you'll see how to use **services** to deal with situations just like this. Along the way you'll see how to use some of **Android's built-in services**. You'll see how to keep your users informed with the **notification service**, and how the **location service** can tell you where you're located.

The starter service app	513
The <code>MainActivity.java</code> file	515
How to log a message	545
The <code>DelayedMessageService.java</code>	517
The <code>OdoraterMessageService.java</code> code	551
How you use the notification service	557
Getting your first location as soon as possible	558
Send the notification using the notification service	561
The <code>OdoraterMessageService.java</code>	569
The step-by-step process of the Odorater Service	571
Define the Binder	573
The Service class has four constructors	575
Add the <code>LocationListener</code> to the server	577
Registering the LocationListener	583
The <code>OdoraterService.java</code> code	591
Update <code>AndroidManifest.xml</code>	595
Update <code>MainActivity.java</code>	598
Create a ServiceConnection	587
Bind to the service when the activity starts	588
Display the distance traveled	596
The <code>MainActivity.java</code> code	599
Start Android Studio	595





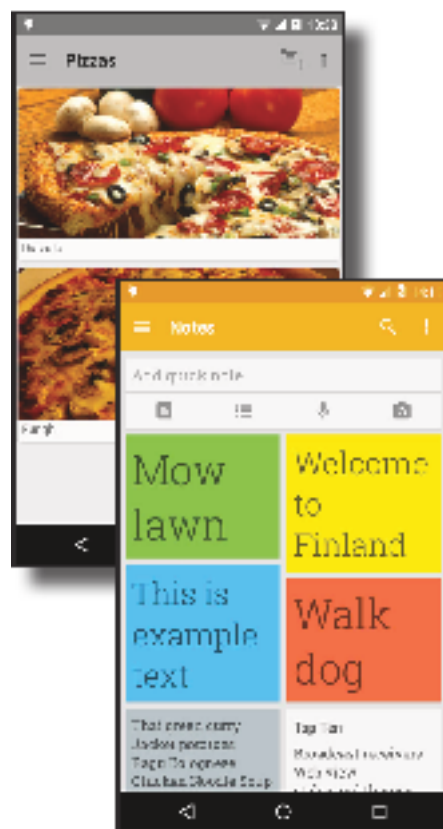
## material design

## 14

## Living in a Material World

## With API level 21, Google introduced Material Design.

In this chapter we'll look at **what Material Design is**, and how to make your apps fit in with it. We'll start by introducing you to **card views** you can reuse across your app for a *consistent look and feel*. Then we'll introduce you to the **recycler view**, the list view's *flexible friend*. Along the way you'll see how to **create your own adapters**, and how to **completely change the look of a recycler view** with just *few lines of code*.



Welcome to Material Design	500
The Pizza app structure	501
Create the CardView	503
The <code>CardLayout</code> and <code>LinearLayout</code> classes	506
Create the base adapter	508
Define the adapter's ViewHolder	507
Create the ViewHolder	508
Each card view displays an image and a caption	508
Add the data to the card view	511
The <code>CardView</code> for <code>CaptionedImageAdapter</code>	511
Create the recycler view	512
Add the RecyclerView to the layout	513
The <code>PizzaMaterialImageAdapter</code> class	515
A RecyclerView with a layout manager and an adapter	515
Specifying the layout manager	516
The <code>ViewHolder</code> for <code>PizzaMaterialImageAdapter</code>	517
Get MainActivity to use the new <code>PizzaMaterialImageAdapter</code>	518
What happens when the user clicks	518
Create <code>PizzaDetailActivity</code>	521
What <code>PizzaDetailActivity</code> does to use	522
The code for <code>PizzaDetailActivity.java</code>	524
Get the <code>RecyclerView</code> to respond to clicks	531
Add the interface for the adapter	535
Implement the interface in <code>PizzaMaterialImageAdapter</code>	537
Bring the adapter forward	539
The <code>ViewHolder</code> for <code>ImageAdapter</code>	541
The <code>ViewHolder</code> for <code>ImageAdapter.java</code>	547
Very Award Yourself	547

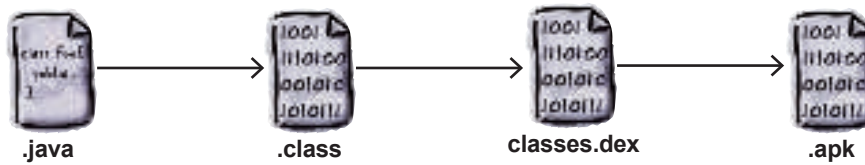
art

## The Android Runtime



**Android apps need to run on devices with low powered processors and very little memory.**

Java apps can take up a lot of memory and because they run inside their own Java Virtual Machine (JVM), Java apps can take a long time to start when they're running on low-powered machines. Android deals with this by not using the JVM for its apps. Instead it uses a very different virtual machine called the Android Runtime (ART). In this appendix we'll look at how ART gets your Java apps to run well on a small, low-powered device.

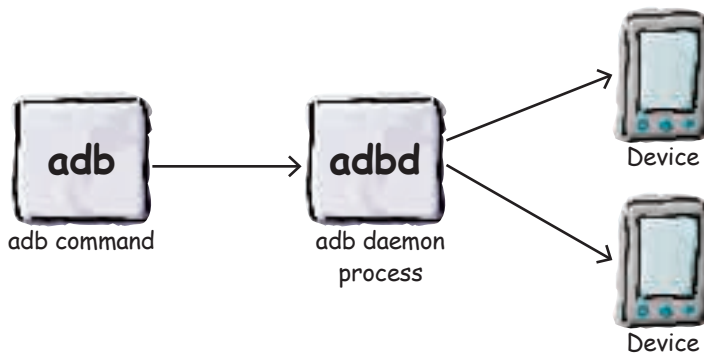


adb

## The Android Debug Bridge



**In this book we've focused on using an IDE for all your Android needs.** But there are times when using a command line tool can be plain useful, like those times when Android Studio can't see your Android device but you just *know* it's there. In this chapter we'll introduce you to the Android Debug Bridge (or adb), a command line tool you can use to communicate with the emulator or Android devices.



## the emulator



### The Android Emulator

**Ever felt like you were spending all your time waiting for the emulator?**

There's no doubt that using the Android emulator is useful. It allows you to see how your app will run on devices other than the physical ones you have access to. But at times, it can feel a little... sluggish. In this appendix we're going to explain why the emulator can seem slow. Even better, we'll give you a few tips we've earned for speeding it up.



## inlayors



### The Top Ten Things (we didn't cover)

**Even after all that, there's still a little more.**

There are just a few more things we think you need to know. We wouldn't feel right about ignoring them, and we really wanted to give you a book you'd be able to lift without extensive training at the local gym. Before you put down the book, **read through these tidbits.**

The battery's running low, in case anyone's interested.



- 1. Disabling your app 606
- 2. Common problems 606
- 3. The WebView class 607
- 4. Animation 607
- 5. Maps 608
- 6. Game Joystick 608
- 7. Touch gestures 610
- 8. Splash screens 621
- 9. Nine-patch graphics 622
- 10. Testing 622



- [\*\*download online \*Statistics of Financial Markets: Exercises and Solutions \(2nd Edition\)\* \(Universitext\) online\*\*](#)
- [download online \*The Demon-Haunted World: Science as a Candle in the Dark\* book](#)
- [click \*Einstein's Clocks and Poincare's Maps: Empires of Time: Empires of Time\*](#)
- [Amnesia \(Peter Zak, Book 1\) book](#)
  
- <http://www.shreesaiexport.com/library/The-Common-Law-Mind--Medieval-and-Early-Modern-Conceptions.pdf>
- <http://www.shreesaiexport.com/library/The-Demon-Haunted-World--Science-as-a-Candle-in-the-Dark.pdf>
- <http://fitnessfatale.com/freebooks/Beatrix-Potter-s-Journal.pdf>
- <http://kamallubana.com/?library/Introductory-Botany--Plants--People--and-the-Environment--2nd-Edition-.pdf>