Lauren Darcey
Shane Conder

Full Color

Sams Teach Yourself

Android™ Application Development

in 24 Hours

SAMS

Sams Teach Yourself Android™ Application Development in 24 Hours

Visible Earth images owned by NASA, http://visibleearth.nasa.gov/.

ISBN-13: 978-0-321-67335-0
ISBN-10: 0-321-67335-2

Library of Congress Cataloging-in-Publication Data

Darcey, Lauren, 1977-

Sams teach yourself Android application development in 24 hours / Lauren Darcey, Shane Conder.

p. cm. — (Sams teach yourself in 24 Hours)

Includes index.

ISBN 978-0-321-67335-0 (pbk.)

1. Application software—Development. 2. Android (Electronic resource) 3. Mobile computing. I. Conder, Shane, 1975- II. Title.

QA76.76.A65D26 2010

005.1—dc22

2010011663

Printed in the United States of America

First Printing June 2010

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The authors and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the programs accompanying it.

Bulk Sales

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales
1-800-382-3419
corpsales@pearsontechgroup.com

For sales outside of the U.S., please contact

International Sales
international@pearson.com

Editor-in-Chief
Mark Taub

Acquisitions Editor
Trina MacDonald

Development Editor
Michael Thurston

Managing Editor
Kristy Hart

Project Editor
Betsy Harris

Copy Editor
Kitty Wilson

Indexer
Erika Millen

Proofreader
Sheri Cain

Technical Editor
Jonathan Jackson

Publishing Coordinator
Olivia Basegio

Book Designer
Gary Adair

Senior Compositor
Gloria Schurick

Introduction

The Android platform is packing some serious heat these days in the mobile marketplace and gaining traction worldwide. The platform has seen numerous advancements in terms of SDK functionality, handset availability, and feature set. A wide diversity of Android handsets and devices are now shipping and (finally) in consumers' hands—and we're not just talking about phones: Android has begun to ship on netbooks, Internet tablets (such as the ARCHOS 5), ebook readers (like the Barnes & Noble nook), digital photo frames, and a variety of other consumer electronics. There are even proof-of-concept appliances such as an Android microwave and washer/dryer combo. (Hey, why not? See http://bit.ly/bGqmZp.) Mobile operators and carriers are taking the platform seriously and spending gazillions on ad campaigns for Android phones—like Verizon's Droid campaign.

In the past year or so, the Android platform has transitioned from a "gearheads-only" platform to providing some serious competition to more established platforms. (Yes, we're talking about platforms such as the iPhone.)

But let's not digress into an argument over whose platform is better so early, okay? Because, honestly, you're wasting your time if you think there's one platform to rule them all. The reality is, people the world over use different phones in different places (CDMA, GSM) and for different reasons (price, availability, coverage quality, feature set, design, familiarity, compatibility). There is no one-size-fits-all answer to this debate.

Having developed for just about every major mobile platform out there, we are keenly aware of the benefits and drawbacks of each platform. We do not presume to claim that one platform is better than another in general; each platform has distinct advantages over the rest, and these advantages can be maximized.

The trick is to know which platform to use for a given project. Sometimes, the answer is to use as many platforms as possible. Lately, we've been finding that the answer is the Android platform: It's inexpensive and easy to develop for, it's available to millions of potential users worldwide, and it has fewer limitations than other platforms.

Still, the Android platform is relatively young and has not yet reached its full-fledged potential. This means frequent SDK updates, an explosion of new devices on the market, and a nearly full-time job keeping track of everything going on in the Android world.

In other words, it may be a bit of a bumpy ride, but there's still time to jump on this bandwagon, write some kick-butt applications, and make a name for yourself.

So let's get to it.

Who Should Read This Book?

There's no reason anyone with an Android handset and a good idea for a mobile application couldn't put this book to use for fun and profit. Whether you're a programmer looking to break into mobile technology or an entrepreneur with a cool app idea, this book is for you.

We make very few assumptions about you as a reader of this book. You may have a basic understanding of the Java programming language (understanding classes, methods, basic inheritance, and so on), but Android makes a fantastic platform for learning Java as well. We have avoided using any fancy or confusing Java in this book, so if you're just getting started with programming, you should be able to read the first few chapters of any introductory Java book or do an online tutorial and have enough Java knowledge to make it through this book alive.

We do assume that you're somewhat comfortable installing applications on a computer (for example, Eclipse, the Java JDK, and the Android SDK) and tools and drivers (for USB access to a phone), and we assume that you can navigate your way around an Android handset well enough to launch applications and such. No wireless development experience is necessary.

How This Book Is Structured

In 24 easy one-hour lessons, you'll design and develop a fully functional network- and LBS (Location-Based Services)-enabled Android application, complete with social features. Each lesson builds on your knowledge of newly introduced Android concepts, and you'll iteratively improve your application from chapter to chapter.

This book is divided into six parts:

- **Part I: Android Fundamentals**

  In Part I, you'll get an introduction to Android, become familiar with the Android SDK and tools, install the development tools, and write your first Android application. Part I also introduces the design principles necessary to write Android applications, including how Android applications are structured and configured, as well as how to incorporate application resources such as strings, graphics, and user interface components into your projects.

- **Part II: Building an Application Framework**

  In Part II, you'll begin developing an application framework that will serve as primary teaching-tool for the rest of the book. You'll start by developing an animated splash screen, followed by screens for main menu, settings, help, and scores. You'll

learn basic user interface design principles, how to collect input from the user, and how to display dialogs to the user. Finally, you'll implement the core application logic of the game screen.

- **Part III: Enhancing Your Application with Powerful Android Features**

  In Part III, you'll dive deeper into the Android SDK, adding more specialized features to the Been There, Done That! application. You'll learn how to work with graphics and the built-in camera, how to leverage LBS, how to network-enable your application, and how to enhance your application with social features.

- **Part IV: Adding Polish to Your Android Application**

  In Part IV, you'll learn how to customize your application for different handsets, screen sizes, and foreign languages. You'll also learn about different ways to test mobile applications.

- **Part V: Publishing Your Application**

  In Part V, you'll learn what you need to do to prepare for and publish your Android applications to the Android Market.

- **Part VI: Appendixes**

  In Part VI, you'll find several helpful references for setting up your Android development environment, using the Eclipse IDE, and accessing supplementary book materials, like the book websites and downloadable source code.

What Is (and Isn't) in This Book

While we specifically targeted Android SDK Version 2.1 in this book, many of the examples were tested on handsets running a variety of Android SDK versions.

The Android SDK is updated *very* frequently (every few months). We kept this in mind when choosing which features of the SDK to highlight to ensure maximum forward and backward compatibility. When necessary, we point out areas where the Android SDK version affects the features and functionality available to the developer.

This book is written in a beginner's tutorial style. If you're looking for an exhaustive reference on Android development, with cookbook-style code examples and a more thorough examination of all the features of the Android platform, we recommend our other, more advanced Android book, *Android Wireless Application Development*, which is part of the Addison-Wesley *Developer's Library* series.

What Development Environment Is Used?

The code in this book was written using the following development environments:

- Windows 7 and Mac OS X 10.6
- Eclipse Java IDE Version 3.5 (Galileo)
- Eclipse JDT plug-in and Web Tools Platform (WTP)
- Sun Java SE Development Kit (JDK) 6 Update 18
- Android SDK Version 2.1 (Primary target, developed and tested on a variety of SDK versions)
- Various Android handsets (Android SDK 1.6, 2.0.1, and 2.1)

What Conventions Are Used in This Book?

This book presents several types of sidebars for special kinds of information:

- **Did You Know?** messages provide useful information or hints related to the current text.
- **By the Way** messages provide additional information that might be interesting or relevant.
- **Watch Out!** messages provide hints or tips about pitfalls that may be encountered and how to avoid them.

This book uses the following code-related conventions:

- Code and programming terms are set in a `monospace` font.
- ➥ is used to signify that the code that follows should appear on the same line as the preceding code.
- Exception handling and error checking are often removed from printed code samples for clarity and to keep the book a reasonable length.

This book uses the following conventions for step-by-step instructions and explanations:

- The core application developed in this book is developed iteratively. Generally, this means that the first time a new concept is explained, every item related to the new concept is discussed in detail. As we move on to more advanced topics in later lessons,

we assume that you have mastered some of the more rudimentary aspects of Android development from previous chapters, and we do not repeat ourselves much. In some cases, we instruct you to implement something in an early lesson and then help you improve it in a later chapter.

- We assume that you'll read the chapters of this book in order. As you progress through the book, you'll note that we do not spell out each and every step that must be taken for each and every feature you implement to follow along in building the core application example. For example, if three buttons must be implemented on a screen, we walk you step-by-step through the implementation of the first button but leave the implementation of the other two buttons as an exercise for you. In a later chapter on a different topic, we might simply ask you to implement some buttons on another screen.
- Where we tell you to navigate through menu options, we separate options using commas. For example, if we told you to open a new document, we'd say "Select File, New Document."

HOUR 2

Mastering the Android Development Tools

What You'll Learn in This Hour:

- Using the Android documentation
- Debugging applications with DDMS
- Working with the Android Emulator
- Using the Android Debug Bridge (ADB)
- Working with Android virtual devices

Android developers are lucky to have more than a dozen development tools at their disposal to help facilitate the design of quality applications. Understanding what tools are available and what they can be used for is a task best done early in the Android learning process, so that when you are faced with a problem, you have some clue as to which utility might be able to help you find a solution. The Android development tools are found in the `/tools` subdirectory of the Android SDK installation. During this hour, we walk through a number of the most important tools available for use with Android. This information will help you develop Android applications faster and with fewer roadblocks.

Using the Android Documentation

Although it is not a tool, per se, the Android documentation is a key resource for Android developers. An HTML version of the Android documentation is provided in the `/docs` subfolder of the Android SDK documentation, and this should always be your first stop when you encounter a problem. You can also access the latest help documentation online at the Android Developer website, http://developer.android.com.

The Android documentation is divided into six sections (see Figure 2.1):

- **SDK**—This tab provides important information about the SDK version installed on your machine. One of the most important features of this tab is the release notes, which describe any known issues for the specific installation. This information is also useful if the online help has been upgraded but you want to develop to an older version of the SDK.
- **Dev Guide**—This tab links to the Android Developer's Guide, which includes a number of FAQs for developers, as well as step-by-step examples and a useful glossary of Android terminology for those new to the platform.
- **Reference**—This tab includes a searchable package and class index of all Android APIs provided as part of the Android SDK.
- **Blog**—This tab links to the official Android developer blog. Check here for the latest news and announcements about the Android platform. This is a great place to find how-to examples, learn how to optimize Android applications, and hear about new SDK releases and Android Developer Challenges.
- **Videos**—This tab, which is available online only, is your resource for Android training videos. Here, you'll find videos about the Android platform, developer tips, and the Google I/O conference sessions.
- **Community**—This tab is your gateway to the Android developer forums. There are a number of Google groups you can join, depending on your interests.

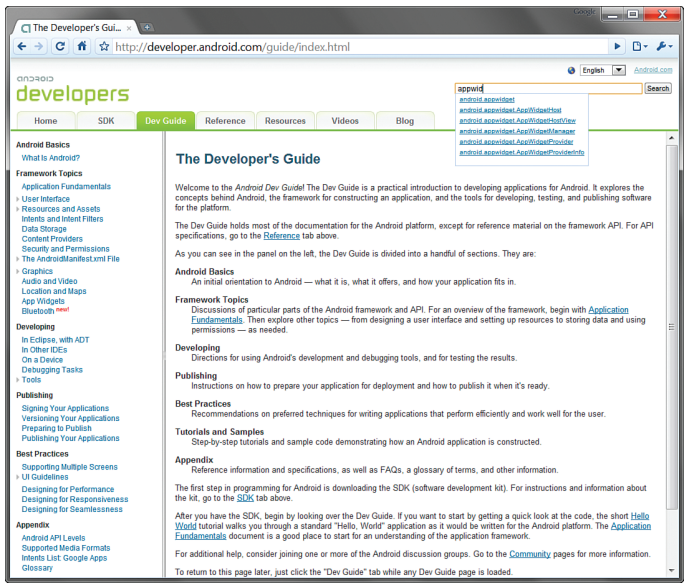


FIGURE 2.1
Android developer documentation (online version).

Now is a good time to get to know your way around the Android SDK documentation. First, try the local documentation and then check out the online documentation.

Debugging Applications with DDMS

The Dalvik Debug Monitor Service (DDMS) is a debugging utility that is integrated into Eclipse through the DDMS perspective. The DDMS perspective provides a number of useful features for interacting with emulators and handsets (see Figure 2.2).

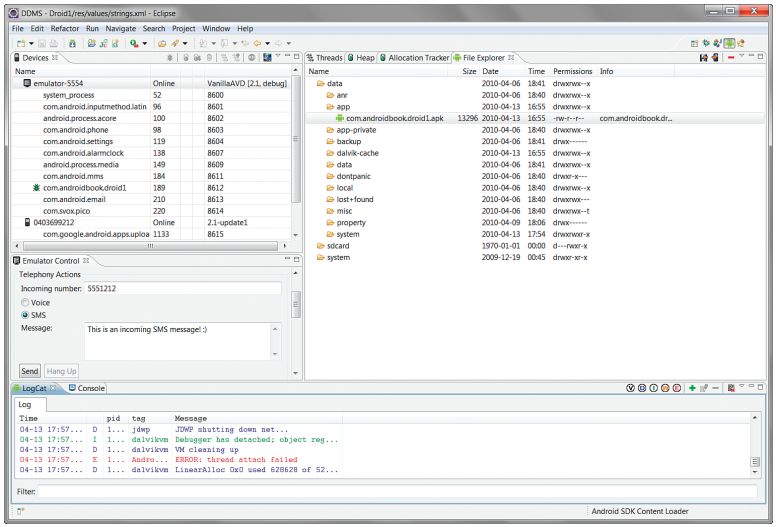


FIGURE 2.2
The DDMS perspective, with one emulator and one Android device connected.

The features of DDMS are roughly divided into five functional areas:

- Task management
- File management
- Emulator interaction
- Logging
- Screen captures

DDMS and the DDMS perspective are essential debugging tools. Now let's take a look at how to use these features in a bit more detail.

Did you Know?

The DDMS tool can be launched separately from Eclipse. You can find it in the Android SDK `/tools` directory.

Managing Tasks

The top-left corner of the DDMS lists the emulators and handsets currently connected. You can select individual instances and inspect processes and threads. You can inspect threads by clicking on the device process you are interested in—for example, com.androidbook.droid1—and clicking the Update Threads button (), as shown in Figure 2.3. You can also prompt garbage collection on a process and then view the heap updates by clicking the green cylinder button (). Finally, you can stop a process by clicking the button that resembles a stop sign ().

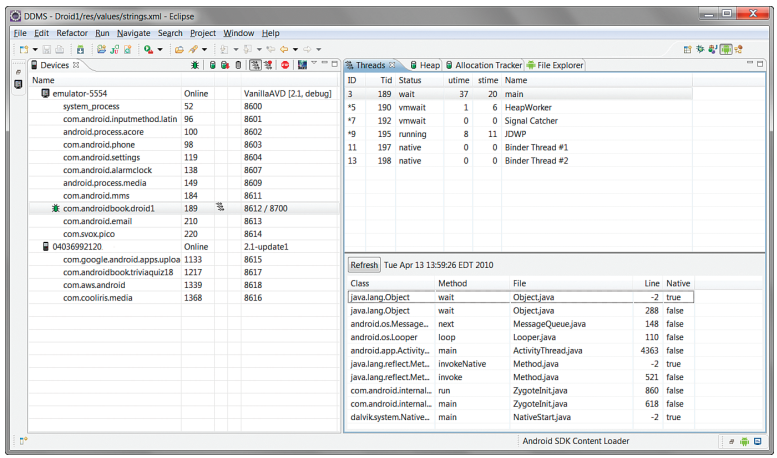


FIGURE 2.3 Using DDMS to examine thread activity for the Droid1 application.

Debugging from the DDMS Perspective

Within the DDMS perspective, you can choose a specific process on an emulator or a handset and then click the little green bug () to attach a debugger to that process. You need to have the source code in your Eclipse workspace for this to work properly. This works only in Eclipse, not in the standalone version of DDMS.

Browsing the Android File System

You can use the DDMS File Explorer to browse files and directories on the emulator or a device (see Figure 2.4). You can copy files between the Android file system and your development machine by using the push ( ) and pull ( ) icons.

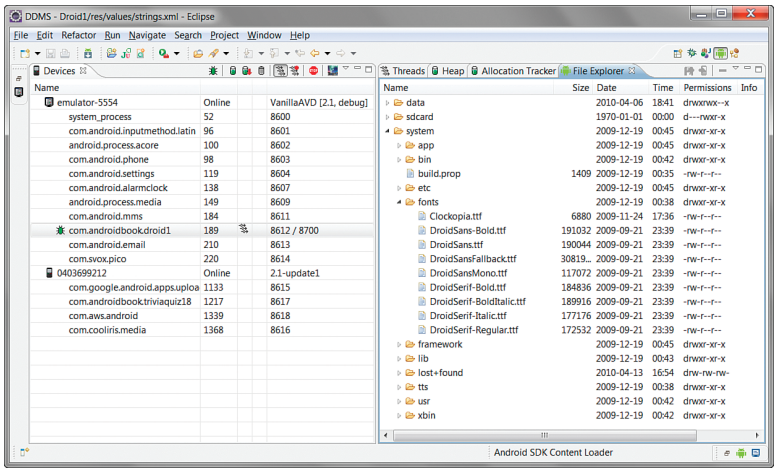


FIGURE 2.4 Using the DDMS File Explorer to browse system fonts on the handset.

You can also delete files and directories by using the minus button ( ) or just pressing Delete. There is no confirmation for this Delete operation, nor can it be undone.

Interacting with Emulators

DDMS can send a number of events, such as simulated calls, SMS messages, and location coordinates, to specific emulator instances. These features are found under the Emulator Control tab in DDMS. These events are all "one way," meaning that they can be initiated from DDMS, not from the emulator to DDMS.

> These features work for emulators only, not for handsets. For handsets, you must use real calls and real messages.

By the Way

Simulating Incoming Calls to the Emulator

You can simulate incoming voice calls by using the DDMS Emulator Control tab (see Figure 2.5). This is not a real call; no data (voice or otherwise) is transmitted between the caller and the receiver.

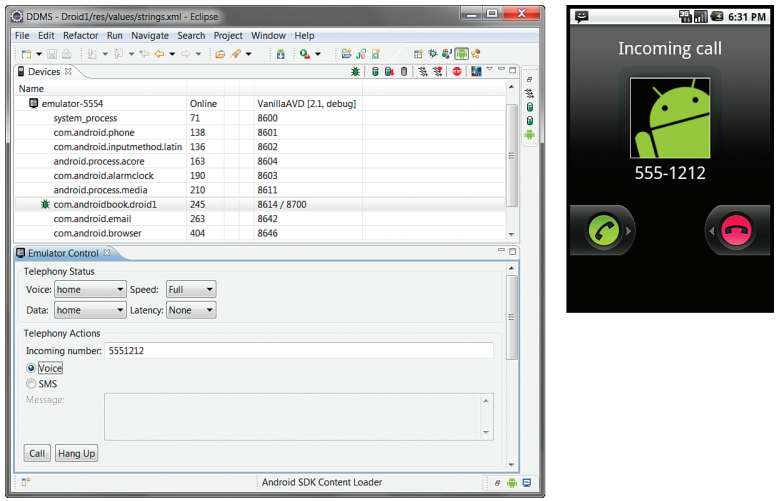


FIGURE 2.5
Using the DDMS Emulator Control tab (left) to place a call to the emulator (right).

Try It Yourself

Simulate an Incoming Call to an Emulator

To simulate an incoming call to an emulator running on your machine, follow these steps:

1. In DDMS, choose the emulator you want to call.
2. On the Emulator Control tab, input the incoming phone number (for example, 5551212) in the Telephony Actions section.
3. Select the Voice radio button.
4. Click the Call button.
5. In the emulator, you should see an incoming call. Answer the call by clicking the Send button in the emulator.
6. End the call at any time by clicking the End button in the emulator or by clicking the Hang Up button on the DDMS Emulator Control tab.

Simulating Incoming SMS Messages to the Emulator

You can simulate incoming SMS messages by using the DDMS Emulator DDMS (see Figure 2.6). You send an SMS much as you initiate a voice call.

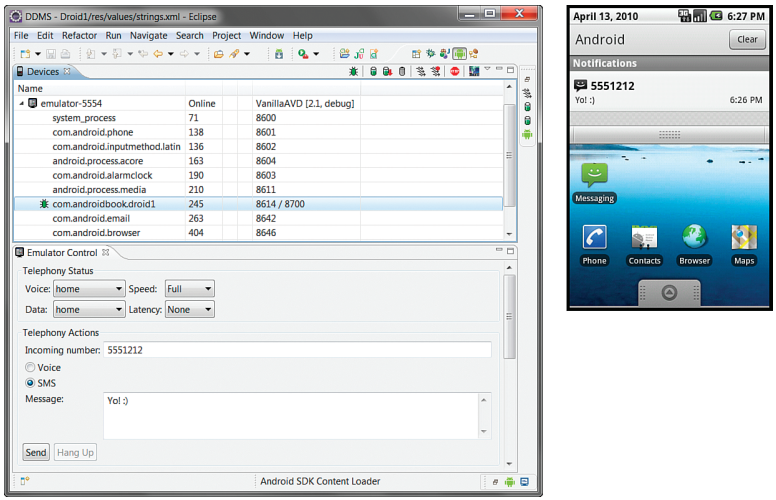


FIGURE 2.6 Using the DDMS Emulator Control tab (left) to send an SMS message to the emulator (right).

Try It Yourself

Send an SMS to the Emulator

To send an SMS message to an emulator running on your machine, follow these steps:

1. In DDMS, choose the emulator you want a send an SMS to.
2. On the Emulator Control tab, input the Incoming phone number (for example, 5551212) in the Telephony Actions section.
3. Select the SMS radio button.
4. Type an SMS message.
5. Click the Send button. In the emulator, you should see an incoming SMS notification.

Taking Screenshots of the Emulator or Handset

One feature that can be particularly useful for debugging both handsets and emulators is the ability to take screenshots of the current screen (see Figure 2.7).

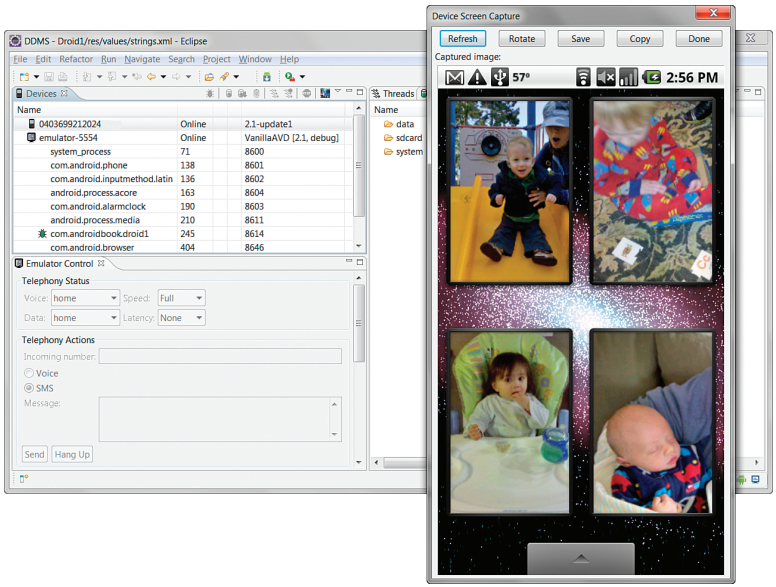


FIGURE 2.7
Using the DDMS Screen Capture button to take a screenshot of the handset.

Try It Yourself

Take a Screen Capture

The screenshot feature is particularly useful when used with true handsets. To take a screen capture, follow these steps:

1. In DDMS, choose the device (or emulator) you want a screenshot of.
2. On that device or emulator, make sure you have the screen you want. Navigate to it, if necessary.
3. Choose the multicolored square picture icon ( ) to take a screen capture. This launches a capture screen dialog.
4. Within the capture screen, click Save to save the screenshot to your local hard drive.

Viewing Log Information

The LogCat logging utility that is integrated into DDMS allows you to view the Android logging console. You may have noted the LogCat logging tab, with its diagnostic output, in many of the figures shown so far in this chapter. We will talk more about how to implement your own custom application logging in Hour 3, "Building Android Applications."

Filtering Log Information

Eclipse has the ability to filter logs by log severity. You can also create custom log filters by using tags. For more information on how to do this, see Appendix B, "Eclipse IDE Tips and Tricks."

Working with the Android Emulator

The Android emulator is probably the most powerful tool at a developer's disposal. It is important for developers to learn to use the emulator and understand its limitations.

The Android emulator is integrated with Eclipse, using the ADT plug-in for the Eclipse IDE.

Emulator Limitations

The Android emulator is a convenient tool, but it has a number of limitations:

- The emulator is not a device. It simulates general handset behavior, not specific hardware implementations.
- Sensor data, such as satellite location information, battery and power settings, and network connectivity, are all simulated using your computer.
- Peripherals such as camera hardware are not fully functional.
- Phone calls cannot be placed or received but are simulated. SMS messages are also simulated and do not use a real network.
- No USB or Bluetooth support is available.

Using Android emulator is not a substitute for testing on a true target handset or device.

Providing Input to the Emulator

As a developer, you can provide input to the emulator in a number of ways:

- Use your computer mouse to click, scroll, and drag items (for example, side volume controls) onscreen as well as on the emulator skin.
- Use your computer keyboard to input text into controls.
- Use your mouse to simulate individual finger presses on the soft keyboard or physical emulator keyboard.
- Use a number of emulator keyboard commands to control specific emulator states.

Try It Yourself

Try out some of the methods of interacting with the emulator:

1. In Eclipse, launch the Droid1 application you created in Hour 1, "Getting Started with Android."
2. While your application is running, press the control-F11 and control-F12 keys to toggle the emulator orientation. Note how your application redraws the simple screen in portrait and landscape modes.
3. Press Alt+Enter to enter full screen mode with the emulator. Then press Alt+Enter again to return to normal mode.

Many useful commands are available. For an exhaustive list, see the official emulator documentation that was installed with the Android SDK documentation and is also available online, at http://developer.android.com/guide/developing/tools/emulator.html.

Exploring the Android System

If you're not already familiar with Android devices, now is a good time to learn your way around the Android system as users see it. Table 2.1 lists some important features of Android.

TABLE 2.1 Android System Screens and Features

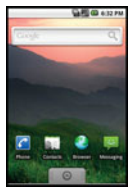
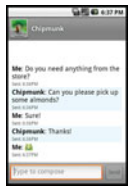
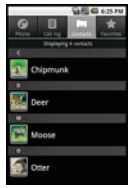
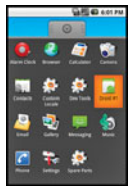
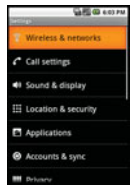
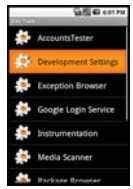
Feature	Description	Appearance
Home screen	Default screen. This is a common location for app widgets and live folders.	
Dialer application	Built-in application for making and receiving phone calls. Note that the emulator has limited phone features.	
Messaging application	Built-in application for sending and receiving SMS messages. Note that the emulator has limited messaging features.	
Browser application	Built-in web browser. Note that the emulator has an Internet connection, provided that your machine has one.	
Contacts application	Database of contact information.	
Application sliding drawer	Shows all installed applications. From the Home screen, pull the gray sliding drawer tab to see all installed applications.	

TABLE 2.1 continued

Feature	Description	Appearance
Settings application	Built-in application to configure a wide variety of "phone" settings for the emulator, such as application management, sound and display settings, and localization.	
Dev Tools application	Built-in application to configure configure development tool settings.	

Using Emulator Skins

Emulator features such as screen size, screen orientation, and whether the emulator has a hardware or soft keyboard are dictated by the emulator skin. The Android SDK supports a number of different skins which emulate various handset screen resolutions (the default being HVGA). The specific skins available depends on the target build platform. Determining the appropriate skin is part of the AVD configuration process.

Using SD Card Images with the Emulator

To save data with the emulator, there must be an SD card image configured. For example, you must have a properly configured SD card image to save media files like camera graphics and sound files to the emulator.

The most convenient way to create SD card images for use with the emulator is to create them as part of the AVD process, as you did in Hour 1. SD card images should be at least 9 MiB.

Using Other Android Tools

Although we've already covered the most important tools, a number of other special-purpose utilities are included with the Android SDK:

- **Android Hierarchy Viewer**—Allows developers to inspect application user interface components such as View Properties while the application is running.

- **Draw 9-Patch tool**—Helps developers design stretchable PNG files.
- **AIDL Compiler**—Helps developers create remote interfaces to facilitate inter-process communication (IPC) on the Android platform.
- `mksdcard` **command-line utility**—Allows developers to create stand-alone SD card images for use within AVDs and the emulator.

Developing Android Applications Without Eclipse

Eclipse is the preferred development environment for Android, but it is not required for development. The ADT plug-in for Eclipse provides a convenient entry point for many of the underlying development tools for creating, debugging, packaging, and signing Android applications.

Developers who do not use Eclipse or require some of the more powerful debugging features not available in the Eclipse ADT plug-in can access these underlying tools directly from the command line. Tools such as the following are found in the /tools directory of the Android SDK:

- `android`—Creates Android project files and to manage AVDs.
- `aapt` **(Android Asset Packaging Tool)**—Packages Android project files into .apk files for installation on the emulator and handset.
- `ddms` **(Dalvik Debug Monitor Service)**—Has a user interface of its own, which resembles the Eclipse DDMS perspective.
- `adb` **(Android Debug Bridge)**—Has a command-line interface for interacting with the emulator and the device.

Summary

The Android SDK ships with a number of powerful tools to help with common Android development tasks. The Android documentation is an essential reference for developers. The DDMS debugging tool, which is integrated into the Eclipse development environment, is useful for monitoring emulators and devices. The Android emulator can be used for running and debugging Android applications virtually, without the need for an actual device. There are also a number of other tools for interacting with handsets and emulators at the command-line level, as well as specialized utilities for designing Android application user interfaces and graphics, as well as packaging applications.

Q&A

Q. *Is the Android documentation installed with the Android SDK the same as the documentation found at http://developer.android.com?*

A. No. The documentation installed with the SDK was "frozen" at the time the SDK was released, which means it is more specific to the version of the Android SDK you installed. The online documentation will always be the latest version of the Android SDK. We recommend using the online documentation, unless you are working offline or have a slow Internet connection, in which case the local SDK documentation will suffice.

Q. *Do the different emulator skins have different features?*

A. Yes. The emulator skins correspond to different screen sizes and orientations. They also have keypads and buttons. Some have hardware keyboards, and others rely on soft keyboard support.

Q. *Is testing your application on the emulator alone sufficient?*

A. No. The Android emulator simulates the functionality of a real device and can be a big time- and cost-saving tool for Android projects. It is a convenient tool for testing, but it can only pretend at real device behavior. The emulator cannot actually determine your real location or make a phone call. Also, the emulator is a generic device and does not attempt to simulate any quirky details of a specific handset. Just because your application runs fine on the emulator does not guarantee that it will work on the device.

Workshop

Quiz

1. Which features are available in the DDMS perspective?

   A. Taking screenshots of emulator and handset screens

   B. Browsing the file system of the emulator or handset

   C. Monitoring thread and heap information on the Android system

   D. Stopping processes

   E. Simulating incoming phone calls and SMS messages to emulators

   F. All of the above

2. True or False: You must use the Android emulator for debugging.
3. Which target platforms can Android applications be written for?
4. True or False: The Android emulator is a generic device that supports only one screen configuration.

Answers

1. F. All of the above. The DDMS perspective can be used to monitor, browse, and interact with emulators and handsets in a variety of ways.
2. False. The Android emulator is useful for debugging, but you can also connect the debugger to an actual device and debug directly.
3. There are a number of target platforms available and more are added with each new SDK release. Some important platform targets include Android 1.1, Android 1.5, Android 1.6, Android 2.0., Android 2.0.1, and Android 2.1. Targets higher than Android 1.1 can include the Google APIs, if desired. These targets map to the AVD profiles you must create in order to use the Android emulator.
4. False. The Android emulator is a generic device, but it can support several different skins. For a complete list of skins supported, see the Android SDK and AVD Manager in Eclipse.

Exercises

1. Launch the Android emulator and browse the settings available. Change the language settings. Uninstall an application.
2. Launch the Android emulator and customize your home screen. Change the wallpaper. Install an AppWidget. Get familiar with how the emulator tries to mimic a real handset. Note the limitations, such as how the dialer works.
3. Try launching the Hierarchy Viewer tool with the Droid1 project you created in Hour 1. Note how you can drill down to see the `TextView` controls you created.

Index

A

B

C

D

E

F

G

H

I

J-K

L

M

N

O

P

Q

R

S

T

U

V

W

X-Y-Z