LEARNING Android™ APPLICATION PROGRAMMING FOR THE Kindle Fire™

A Hands-On Guide to Building Your First Android Application

LAUREN DARCEY
SHANE CONDER

Praise for *Learning Android Application Programming for the Kindle Fire*

"Now is a great time to learn how to program for the Kindle Fire, and this book is the perfect companion for your journey! Distilled within the text are the proven techniques, best practices, and hard-won wisdom of two of the mobile industry's leading pioneers. You'll be programming Kindle Fire apps in no time!"

—**Mark Hammonds**, Mobile Engineer, and Managing Editor, Mobiletuts+

"Learning Android Application Programming for the Kindle Fire is a must-have developers' resource specific to the Kindle Fire. This book takes you from SDK installation to APK publication with lots of examples and tips along the way."

—**Jim Hathaway**, Web Developer

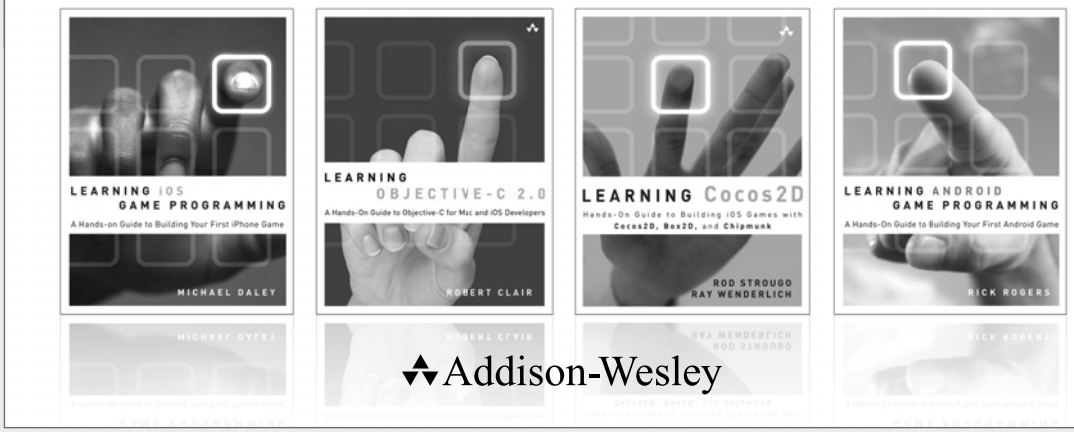
"If you want to bring your application to Amazon's exciting new Android-based tablet, look no further than this book. Darcey and Conder show you how to go from idea to application for the Kindle Fire quickly and clearly with a sample application that shows you everything you'll need to build your application. Along the way, you'll pick up valuable tips on developing software in general."

—**Ray Rischpater**, Senior Research Engineer, Nokia Research Center

"I used Lauren Darcey and Shane Conder's books to start developing Android apps fast. I knew iOS development but needed to learn the basics of Android development. This was a great book to show me how, and I started coding the same day."

—**Scott Walker**, Developer

Addison-Wesley Learning Series



Visit informit.com/learningseries for a complete list of available publications.

The Addison-Wesley Learning Series is a collection of hands-on programming guides that help you quickly learn a new technology or language so you can apply what you've learned right away.

Each title comes with sample code for the application or applications built in the text. This code is fully annotated and can be reused in your own projects with no strings attached. Many chapters end with a series of exercises to encourage you to reexamine what you have just learned, and to tweak or adjust the code as a way of learning.

Titles in this series take a simple approach: they get you going right away and leave you with the ability to walk off and build your own application and apply the language or technology to whatever you are working on.

Learning Android™ Application Programming for the Kindle Fire™

This page intentionally left blank

Learning Android™ Application Programming for the Kindle Fire™

A Hands-On Guide to Building Your First Android Application

Lauren Darcey
Shane Conder

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Cape Town • Sydney • Tokyo • Singapore • Mexico City

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 the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

Android™ is a trademark of Google, Inc. Kindle Fire™ is a trademark of Amazon.com, Inc.

Android™ and Google® are trademarks of Google, Inc. Kindle Fire™ and Amazon.com® are trademarks of Amazon.com, Inc. Neither Amazon.com, Inc., nor Google, Inc., have authorized or approved publication of this work and references to their marks herein are not intended to imply their sponsorship or affiliation with this work.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

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

For sales outside the United States, please contact:

International Sales
international@pearson.com

Visit us on the web: informit.com/aw

The Library of Congress cataloging-in-publication data is on file.

ISBN-13: 978-0-321-83397-6
ISBN-10: 0-321-83397-X

Text printed in the United States on recycled paper at R.R. Donnelley in Crawfordsville, Indiana.

First printing: July 2012

Editor-in-Chief
Mark Taub

Executive Editor
Laura Lewin

Development Editor
Sheri Cain

Managing Editor
Kristy Hart

Project Editor
Jovana San Nicolas-Shirley

Indexer
Larry Sweazy

Proofreader
Gill Editorial Services

Technical Reviewers
Jim Hathaway
Ray Rischpater
Mike Wallace

Publishing Coordinator
Olivia Basegio

Cover Designer
Chuti Prasertsith

Compositor
Nonie Ratcliff

❖

For Ellie

❖

Contents

Table of Contents

Acknowledgments

This book would never have been written without the guidance and encouragement we received from a number of very patient and supportive people, including our editorial team, coworkers, friends, and family.

Throughout this project, our editorial team at Pearson has been top notch. Special thanks go to Laura Lewin and Trina MacDonald, Olivia Basegio, Sheri Cain, and the rest of the Pearson publishing team. Our technical reviewers, Ray Rischpater, Mike Wallace, and Jim Hathaway, helped us ensure that this book provides accurate information. With each edition, this book gets better. However, it wouldn't be here without the help of many folks on past editions. Thanks go out to past reviewers, technical editors, and readers for their valuable feedback. Finally, we want to thank our friends and family members who supported us when we needed to make our deadlines.

About the Authors

Lauren Darcey is responsible for the technical leadership and direction of a small software company specializing in mobile technologies, including Android, Apple iOS, BlackBerry, Palm Pre, BREW, and J2ME and consulting services. With more than two decades of experience in professional software production, Lauren is a recognized authority in application architecture and the development of commercial-grade mobile applications. Lauren received a B.S. in Computer Science from the University of California, Santa Cruz.

Lauren spends her free time traveling the world with her geeky mobile-minded husband and daughter. She is an avid nature photographer whose work has been published in books and newspapers around the world. In South Africa, she dove with 4-meter-long great white sharks and got stuck between a herd of rampaging hippopotami and an irritated bull elephant. She's been attacked by monkeys in Japan, gotten stuck in a ravine with two hungry lions in Kenya, gotten thirsty in Egypt, narrowly avoided a coup d'état in Thailand, geocached her way through the Swiss Alps, drank her way through the beer halls of Germany, slept in the crumbling castles of Europe, and gotten her tongue stuck to an iceberg in Iceland (while being watched by a herd of suspicious wild reindeer).

Shane Conder has extensive development experience and has focused his attention on mobile and embedded development for the past decade. He has designed and developed many commercial applications for Android, Apple iOS, BREW, BlackBerry, J2ME, Palm, and Windows Mobile—some of which have been installed on millions of phones worldwide. Shane has written extensively about the mobile industry and evaluated mobile-development platforms on his tech blogs and is well known within the blogosphere. Shane received a B.S. in Computer Science from the University of California.

A self-admitted gadget freak, Shane always has the latest smartphone, tablet, or other mobile device. He can often be found fiddling with the latest technologies, such as cloud services and mobile platforms, and other exciting, state-of-the-art technologies that activate the creative part of his brain. He is a very hands-on geek dad. He also enjoys traveling the world with his geeky wife, even if she did make him dive with 4-meter-long great white sharks and almost get eaten by a lion in Kenya. He admits that he has to take at least two phones with him when backpacking—even though there is no coverage—and that he snickered and whipped out his Android phone to take a picture when Laurie got her tongue stuck to that iceberg in Iceland, and that he has learned that he should be writing his own bio.

The authors have also published several other Android books, including *Android Wireless Application Development*, *Android Wireless Application Development Volume I: Android Essentials*, *Android Wireless Application Development Volume 2: Advanced Topics*, *Sams Teach Yourself Android Application Development*, and the mini-book *Introducing Android Development with Ice Cream Sandwich*. Lauren and Shane have also published numerous articles on mobile-software development for magazines, technical journals, and online publishers of educational content. You can find dozens of samples of their work in Linux User and Developer, *Smart Developer* magazine (Linux New Media), developer.com, Network World, Envato (MobileTuts+ and

CodeCanyon), and InformIT, among others. They also publish articles of interest to their readers at their own Android website: http://androidbook.blogspot.com. You can find a full list of the authors' publications at http://goo.gl/f0Vlj.

We Want to Hear from You!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

We welcome your comments. You can email or write to let us know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that we cannot help you with technical problems related to the topic of this book. Please contact the authors for technical help related to this book at androidwirelessdev+kf1@gmail.com.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the authors and editors who worked on the book.

Email: errata@informit.com

Mail: Addison-Wesley
ATTN: Reader Feedback
1330 Avenue of the Americas
35th Floor
New York, New York, 10019

Reader Services

Visit our website and register this book at http://informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.

This page intentionally left blank

Introduction

The Amazon Kindle Fire is one of the most exciting and popular new Android devices available to consumers and application developers alike. It's so popular, in fact, that in the six weeks the Amazon Kindle Fire was available for purchase at the end of 2011, 3.9 million units were sold, accounting for over 14 percent of the tablet market, making it the #1-selling Android tablet (http://goo.gl/f9hFM). This book covers Android Fundamentals for Kindle Fire application development. Here, you are introduced to Android, become familiar with the Android SDK and tools, install the development tools, write your first Android application, and deploy it to a Kindle Fire device. You are also introduced to the design principles necessary to write Android applications, including how Android applications are designed, developed, and published on the Amazon Appstore.

Target Audience for This Book

There's no reason anyone with an Amazon Kindle Fire device, a good idea for a mobile application, and some programming knowledge 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 can help you realize your goals of making killer Android apps for devices like the Kindle Fire.

We make as few assumptions about you, as a reader of this book, as possible. No wireless development experience is necessary. 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). We also assume that you own at least one Amazon Kindle Fire device and can navigate your way around it for testing purposes.

Android apps are written in Java. Therefore, we assume you have a reasonably solid understanding of the Java programming language (classes, methods, scoping, object-oriented programming [OOP], and so on), ideally using the Eclipse development environment. Familiarity with common Java packages, such as `java.lang`, `java.net`, and `java.util`, will serve you well.

Android can also be a fantastic platform for learning Java, provided that you have some background in OOP and adequate support, such as a professor or some really good Java programming references. We make every attempt to avoid using any fancy or confusing Java in this book, but you will find that, with Android, certain syntactical Java wizardry not often covered

in your typical beginner's Java book is used frequently: anonymous inner classes, method chaining, templates, reflection, and so on. With patience—and some good Java references—even beginning Java developers should be able to make it through this book alive; those with a solid understanding of Java should be able to take this book and run with it without issue.

Finally, regardless of your specific skill set, we expect you to use this book in conjunction with other supplementary resources, specifically the Android SDK reference and the sample source code that accompanies each coding chapter. The Android SDK reference provides exhaustive documentation about each package, class, and method of the Android SDK. It's searchable online. If we were to duplicate this data in book form, this book would weigh a ton—literally.

How This Book Is Structured

In just a handful of straightforward lessons, you'll design and develop a fully functional Android application specifically for Amazon Kindle Fire devices. Each lesson builds on your knowledge of newly introduced Android concepts, and you'll iteratively improve your application from chapter to chapter. You'll also find numerous exercises at the end of each chapter to help cement your newfound knowledge.

This book is divided into four parts:

- **Part I: Kindle Fire Fundamentals—** Here, you 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. You also learn how to start developing for a specific Android device: the Amazon Kindle Fire.
- **Part II: Building an Application Framework—** In this part, you begin developing an application called Have You Read That?, which serves as the primary teaching tool for the rest of this book. You start by developing its animated splash screen, followed by screens for the main menu, settings, help, and scores. You learn basic user-interface design principles, how to collect input from the user, and how to display dialogs to the user. You implement the core game logic and leverage the network. Finally, you explore some of the secondary issues of application development, like internationalization.
- **Part III: Publishing Your Kindle Fire Application—** Here, you learn what you need to do to prepare for and publish your Android applications to the Amazon Appstore.
- **Part IV: Appendixes—** Here is where you can find several helpful references for setting up your Android development environment for Amazon Kindle Fire development, using the Eclipse IDE, and accessing supplementary book materials, such as this book's websites and downloadable source code.

What Is (and Isn't) in This Book

First and foremost, this book provides a thorough introduction to the Android platform for Kindle Fire application development. In this, we begin with the fundamentals, try to cover the most important aspects of development, and provide information on where to go for more information. This is not an exhaustive reference on the Android SDK. We assume that you will use this book as a companion to the Android SDK documentation, available for download as part of the SDK and online at http://developer.android.com.

We only have a short amount of time to get you, the reader, up to speed on the fundamentals of Android development, so forgive us if we stay strictly to the topic at hand. Therefore, we take the prerequisites listed earlier seriously. This book will not teach you how to program, explain Java syntax and programming techniques, or stray too far into the details of supporting technologies often used by mobile applications, like algorithm design, network protocols, developing web servers, graphic design, database schema design, and other such peripheral topics; there are fantastic references available on each of these subjects.

The Android SDK and related tools are updated frequently (every few months). This means that, no matter how hard we try, some minor changes in step-by-step instructions may occur if you choose to use versions of the tools that do not match those listed in the next section, "Development Environment Used." This book is written for Kindle Fire app developers, so it focuses on the Android SDK version used by this specific device.

Because this book helps developers write applications for the Amazon Kindle Fire, we specifically targeted Android SDK Version 2.3.4 (API Level 10) for the tutorial in this book. Still, we make every effort to make this book compatible with other versions of Android, as well as work smoothly regardless of what version of the Android SDK you want to target.

This book is written in a tutorial style. If you're looking for an exhaustive reference on Android development, with cookbook-style code examples and a more thorough examination of the many features of the Android platform, we recommend our more advanced Android books, *Android Wireless Application Development Volume I: Android Essentials* (Third Edition) (ISBN 978-0321813831) and *Android Wireless Application Development Volume II: Advanced Topics* (Third Edition) (ISBN 978-0321813848), which are part of the Addison-Wesley Developer's Library series.

Development Environment Used

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

- Windows 7 and Mac OS X 10.7.x.
- Eclipse Java IDE Version 3.7 (Indigo).
- Android ADT Plug-in for Eclipse, 16.0.1.

- Android SDK Tools, Release 16.
- Sun Java SE Development Kit (JDK) 6 Update 21.
- Code examples target Android SDK API Level 10.
- The code was tested on the original Android Kindle Fire (Android SDK 2.3.4, API Level 10).
- The network portions of the sample application leverage Google App Engine.

Conventions Used in This Book

This book uses the following code-related conventions:

- Code and programming terms are set in a `monospace` font.
- ➥ signifies 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 this 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."

About the Short Links

We chose to make many Internet links in this book short links. This benefits the readers of the print book by making it far easier and less prone to error when typing links. These links are all shortened with the goo.gl link shortener, which is a service provided by Google. If the target of the link goes away, neither the original link nor the shortened link would work. We're confident that this is the easiest way for readers to effectively use the links we provide. In addition, as authors, we get to see which links readers are actually using.

Sometimes, link shorteners are used as a way to hide nefarious links. Be assured that we have only shortened links that we believe to be valid and safe. In addition, Google provides screening of the target URLs for malware, phishing, and spam sites. Should a target link change hands and become a bad link, using the shortened link will provide you, the reader, with an extra layer of protection.

For more information on this subject, see http://www.google.com/support/websearch/bin/answer.py?answer=190768 (http://goo.gl/iv8c7).

Code Examples for This Book

The source code is available for download on the publisher's website (www.informit.com/title/9780321833976) and on the authors' website (http://androidbook.blogspot.com/p/book-code-downloads.html).

We provide complete, functional code projects for each coding chapter in this book. If you're having trouble building the tutorial application as you go along, compare your work to the sample code for that chapter. The sample code is not intended to be the "answer," but it is the complete code listings that could not otherwise be reproduced in a book of this length.

Supplementary Tools Available

Shane Conder and Lauren Darcey, the authors, also run a blog at http://androidbook.blogspot.com, where you can always download the latest source code for their books. This website also covers a variety of Android topics and reader discussions, questions, clarifications, the occasional exercise walkthrough, and lots of other information about Android development. You can also find links to their various technical articles online and in print.

Contacting the Authors

Feel free to contact us if you have specific questions; we often post addendum information or tool-change information on our book website:

http://androidbook.blogspot.com

You can also email us at androidwirelessdev+kf1@gmail.com.

This page intentionally left blank

1

Getting Started with Kindle Fire

Android is the first *complete*, *open*, and *free* mobile platform. Developers enjoy a comprehensive software development kit (SDK), with ample tools for developing powerful, feature-rich applications. The platform is open source, relying on tried-and-true open standards with which developers will be familiar. Best of all, there are no costly barriers to entry for developers: no required fees. (A modest fee is required to publish on third-party distribution mechanisms, such as the Android Market.) Android developers have numerous options for distributing and commercializing their applications.

Introducing Android

To understand where Android fits with other mobile technologies, let's take a minute to talk about how and why this platform came about.

Google and the Open Handset Alliance

In 2007, a group of handset manufacturers, wireless carriers, and software developers (notably, Google) formed the Open Handset Alliance, with the goal of developing the next generation of wireless platform. Unlike existing platforms, this new platform would be nonproprietary and based on open standards, which would lead to lower development costs and increased profits. Mobile software developers would also have unprecedented access to the handset features, allowing for greater innovation.

As proprietary platforms, such as RIM BlackBerry and Apple iPhone, gained traction, the mobile development community eagerly listened for news of this potential game-changing platform.

Android Makes Its Entrance

In 2007, the Open Handset Alliance announced the Android platform and launched a beta program for developers. Android went through the typical revisions of a new platform. Several preview versions of the Android SDK were released. The first Android handset (the T-Mobile G1) began shipping in late 2008. Throughout 2009 and 2010, new and exciting Android smartphones reached markets throughout the world, and the platform proved itself to industry and consumers alike. Over the last three years, numerous revisions to the Android platform have been rolled out, each providing compelling features for developers to leverage and users to enjoy. Recently, mobile platforms began to consider devices above and beyond the traditional smartphone paradigm to other devices, such as tablets, ebook readers, and set-top boxes, like Google TV.

As of this writing, hundreds of Android devices are available to consumers around the world—from high-end smartphones to low-end "free with contract" handsets and everything in between. This figure does not include the numerous Android tablet and e-book readers also available, the dozens of upcoming devices already announced, or the consumer electronics running Android. (For a nice list of Android devices, check out this Wikipedia link: http://goo.gl/fU2X5.) More than 450,000 applications are currently published on the Android Market (now called Google Play), and there are more than 30,000 applications on the Amazon Appstore for Android. In the United States, all major carriers now carry Android phones prominently in their product lines, as do many in Asia, Europe, Central/South America, and beyond. The rate of new Android devices reaching the world markets continues to increase.

Google has been a contributing member of the Open Handset Alliance from the beginning. The company hosts the Android open source project and the developer website (http://developer.android.com). This website is your go-to site for downloading the Android SDK, getting the latest platform documentation, and browsing the Android developer forums. Google also runs the most popular service for selling Android applications to end users: the Android Market. The Android mascot is a little green robot (see Figure 1.1).

Cheap and Easy Development

If there's one time when "cheap and easy" is a benefit, it's with mobile development. Wireless application development, with its ridiculously expensive compilers and preferential developer programs, has been notoriously expensive to break into compared to desktop development. Here, Android breaks the proprietary mold. Unlike other mobile platforms, there are virtually no costs to developing Android applications.

The Android SDK and tools are freely available on the Android developer website (http://developer.android.com [http://goo.gl/K8GgD]). The freely available Eclipse program has become the most popular integrated development environment (IDE) for Android application development; there is a powerful plug-in available on the Android developer site for facilitating Android development with Eclipse.

Figure 1.1 The Android Mascot (The Bugdroid)

So, we covered cheap; now let's talk about why Android development is easy. Android applications are written in Java, which is one of the most popular development languages around. Java developers will be familiar with many of the packages provided as part of the Android SDK, such as `java.net`. Experienced Java developers will be pleased to find that the learning curve for Android is reasonable.

In this book, we focus on the most common, popular, and simple setup for developing Android applications:

- We use the most common and supported development language: Java. Although we do not teach you Java, we try our best to keep the Java code we use simple and straightforward so that even beginners won't wrestle with syntax. Even so, if you are new to Java, we recommend *Sams Teach Yourself Java in 24 Hours* by Rogers Cadenhead and *Thinking in Java* by Bruce Eckel, Fourth Edition in Print (Third Edition free from http://goo.gl/mtjoz—a zip file from Bruce Eckel's site at http://www.mindviewinc.com/Books/).
- We use the most popular development environment: Eclipse. It's free, it's well supported by the Android team, and it's the only supported IDE compatible with the Android Development Tools plug-in. Did we mention it's free?

- We write instructions for the most common operating system used by developers: Windows. Users of Linux or Mac may need to translate some keyboard commands, paths, and installation procedures.
- We focus on the Android platform version available on the Amazon Kindle Fire: Android 2.3.4 (API Level 10).

If you haven't installed the development tools needed to develop Android applications or the Android SDK and tools yet, do so at this time.

Let's get started!

> **Note**
>
> You can find all the details of how to install and configure your computer for Android application development in Appendix A, "Configuring Your Android Development Environment." You will need to install and configure Java, Eclipse, the Android SDK, and the ADT plug-in for Eclipse. You will need to configure ADB and possibly USB drivers for connecting your development machine to Kindle Fire for debugging. Again, all this is covered in Appendix A.

Familiarizing Yourself with Eclipse

Let's begin by writing a simple Android "Hello, World" application that displays a line of text to the user. As you do so, you also tour the Eclipse environment. Specifically, you learn about some of the features offered by the Android development tools (ADT) plug-in for Eclipse. The ADT plug-in provides functionality for developing, compiling, packaging, and deploying Android applications. Specifically, the ADT plug-in provides the following features:

- The Android project wizard, which generates all the required project files
- Android-specific resource editors, including a graphical layout editor for designing Android application user interfaces
- The Android SDK and the AVD Manager
- The Eclipse DDMS perspective for monitoring and debugging Android applications
- Integration with the Android LogCat logging utility
- Integration with the Android Hierarchy Viewer layout utility
- Automated builds and application deployment to Android emulators and devices
- Application packaging and code signing tools for release deployment, including ProGuard support for code optimization and obfuscation

Now, let's take some of these features for a spin.

Creating Android Projects

The Android project wizard creates all the required files for an Android application. Open Eclipse and follow these steps to create a new project:

1. Choose File, New, Android Project or click the Android Project creator icon on the Eclipse toolbar.

> **Note**
>
> The first time you try to create an Android Project in Eclipse, you might need to choose File, New, Project, and then select Android, Android Project. After you do this once, the Android project type appears in the Eclipse project types, and you can use the method described in step 1.

2. Choose a project name. In this case, name the project `HelloKindle`.
3. Choose a location for the project source code. Because this is a new project, select the Create New Project in Workspace radio button. If you prefer to store your project files in a location other than the default, simply uncheck the Use Default Location checkbox and browse to the directory of your choice. The settings should look like Figure 1.2.

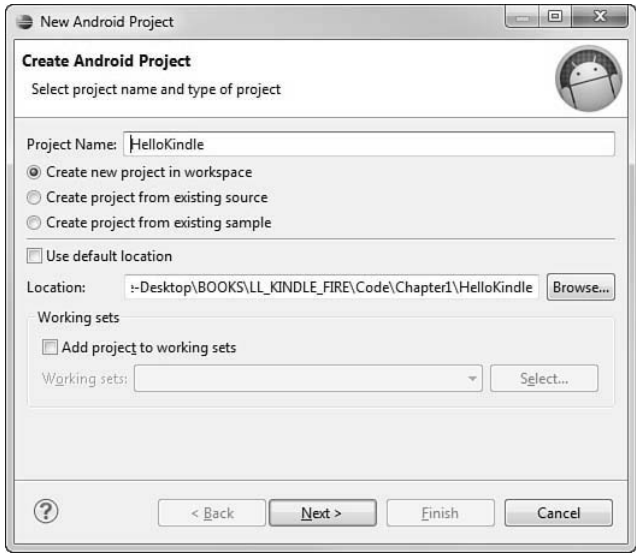


Figure 1.2 Project Name and Location

4. Click the Next button.
5. Select a build target for your application, as shown in Figure 1.3. For most applications, you want to select the version of Android that's most appropriate for the devices used by your target audience and the needs of your application. For Kindle development, choose API Level 10 (Android 2.3.3) using the Android Open Source Project vender version (not the Google, Inc., vender version). Kindle Fire devices do not have access to Google add-ons.

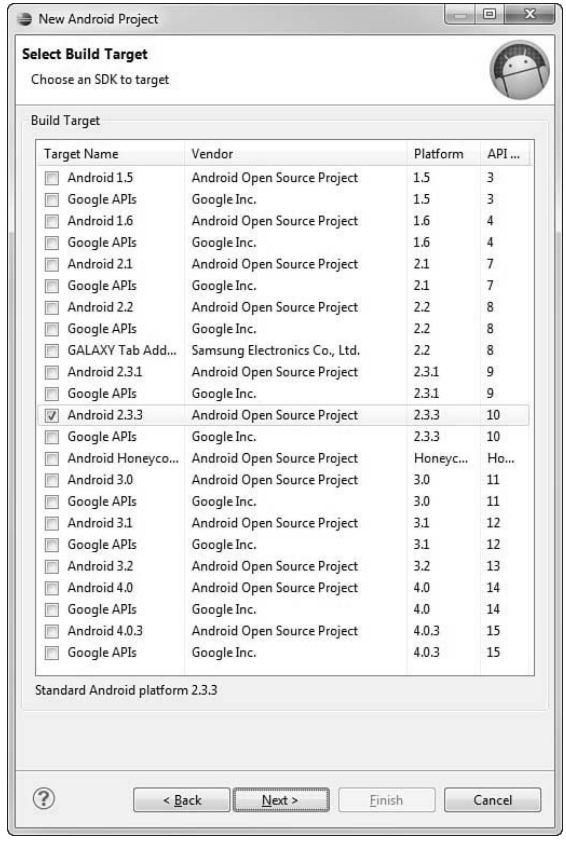


Figure 1.3 Choose SDK Target

6. Click the Next button.
7. Specify an application name. This name is what users will see. In this case, call the application `Hello Kindle`.

8. Specify a package name, following standard package-namespace conventions for Java. Because all code in this book falls under the `com.kindlebook.*` namespace, use the package name `com.kindlebook.hellokindle`.
9. If needed, check the Create Activity checkbox. This instructs the wizard to create a default launch `Activity` class for the application. Call your activity `HelloKindleActivity`.
10. Confirm that the Minimum SDK field is correct. This field will be set to the API level of the build target by default. (Android 2.3.3 is API Level 10.) If you want to support older versions of the Android SDK, you need to change this value. For example, to support devices with Android 1.6, set the Minimum SDK field to API Level 4. The Kindle is based on API Level 10, however, so an application just targeting the Kindle does not need to worry about this. Your project settings will look like what's shown in Figure 1.4.
11. The Android project wizard allows you to create a test project in conjunction with your Android application, also shown in Figure 1.4. For this example, a test project is unnecessary. However, you can always add a test project later by clicking the Android Test Project creator icon, which is to the right of the Android project wizard icon on the Eclipse toolbar. Test projects are discussed further in Chapter 16, "Testing Kindle Fire Applications."
12. Click the Finish button.

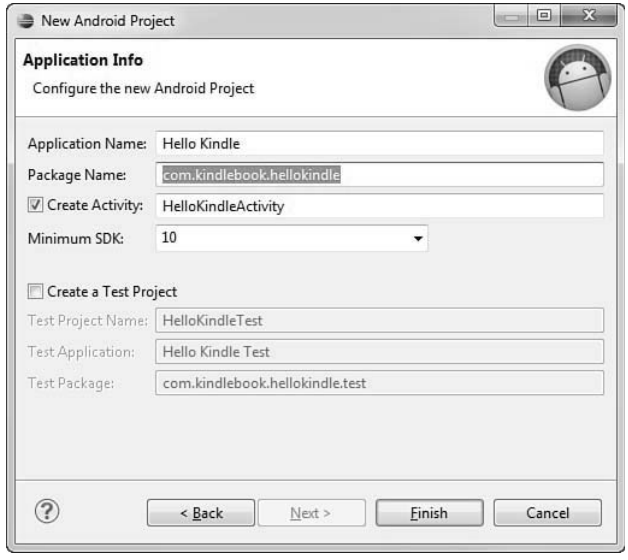


Figure 1.4 Configure Package Name, Initial Activity, and Minimum SDK

> **Note**
>
> You can also add existing Android projects to Eclipse by using the Android project wizard. To do this, simply select Create Project from Existing Source instead of the default Create New Project in Workspace in the New Android Project dialog (refer to Figure 1.2). Several sample projects are provided in the `/samples` directory of the Android SDK, under the specific platform they support. For example, the Android SDK sample projects are found in the directory `/platforms/android-xxx/samples` (where *xxx* is the platform level number, such as 10).
>
> You can also select a third option: Create Project from Existing Sample, which will do as it says. However, make sure that you choose the build target first option to get the list of sample projects that you can create.

Exploring Your Android Project Files

You will now see a new Android project called `HelloKindle` in the Eclipse File Explorer. In addition to linking the appropriate Android SDK jar file, the following core files and directories are created:

- `AndroidManifest.xml`—The central configuration file for the application.
- `project.properties`—A generated build file used by Eclipse and the Android ADT plug-in. Do not edit this file.
- `proguard.cfg`—A generated build file used by Eclipse, ProGuard, and the Android ADT plug-in. Edit this file to configure your code optimization and obfuscation settings for release builds.
- `/src` **folder**—Required folder for all source code.
- `/src/com.kindlebook.hellokindle/HelloKindleActivity.java`—Main entry point to this application, named `HelloKindleActivity`. This activity has been defined as the default launch activity in the Android manifest file.
- `/gen/com.kindlebook.hellokindle/R.java`—A generated resource management source file. Do not edit this file.
- `/assets` **folder**—Required folder where uncompiled file resources can be included in the project.
- `/res` **folder**—Required folder where all application resources are managed. Application resources include animations, drawable graphics, layout files, data-like strings and numbers, and raw files.
- `/res/drawable-*` **folders**—Application icon graphic resources are included in several sizes for different device screen resolutions.
- `/res/layout/main.xml`—Layout resource file used by `HelloKindleActivity` to organize controls on the main application screen.
- `/res/values/strings.xml`—The resource file where string resources are defined.

Editing Project Resources

The Android manifest file is the central configuration file for an Android application. Double-click the `AndroidManifest.xml` file within your new project to launch the Android manifest file editor (see Figure 1.5).

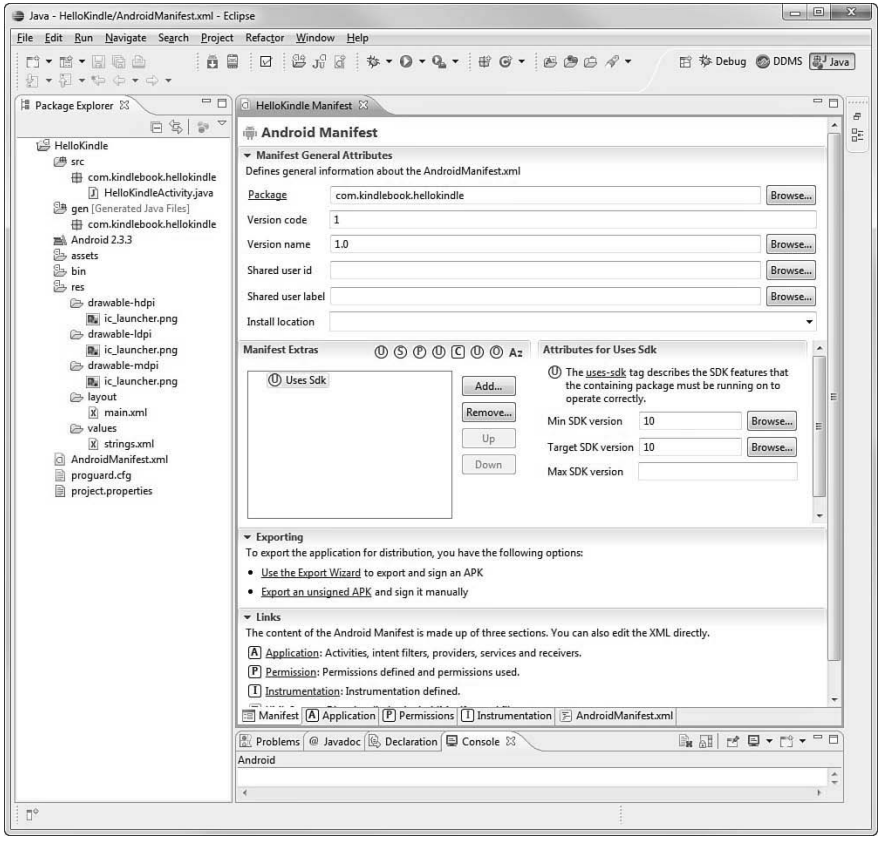


Figure 1.5 Editing the Android Manifest File in Eclipse

Editing the Android Manifest File

The Android manifest file editor organizes the manifest information into numerous tabs:

- **Manifest**—This tab, shown in Figure 1.5, is used for general application-wide settings, such as the package name and application version information (used for installation and upgrade purposes).

- **Application**—This tab is used to define application details, such as the name and icon the application displays, as well as the "guts" of the application, such as what activities can be run (including the default launch `HelloKindleActivity`) and other functionality and services that the application provides.
- **Permissions**—This tab is used to define the application's permissions. For example, if the application requires the ability to access Internet resources, it must register a `uses-permission` tag within the manifest, with the name `android.permission.INTERNET`.
- **Instrumentation**—This tab is used for unit testing, using the various instrumentation classes available within the Android SDK.
- **AndroidManifest.xml**—This tab provides a simple XML editor to directly edit the manifest file. Because all Android resource files, including the Android manifest file, are simply XML files, you can always edit the XML instead of using the resource editors. You can create a new Android XML resource file by clicking the Android XML creator icon  on the Eclipse toolbar.

If you switch to the AndroidManifest.xml tab, your manifest file will look something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.kindlebook.hellokindle"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="10" />
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".HelloKindleActivity"
            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>
</manifest>
```

Editing Other Resource Files

Android applications are made up of functions (Java code, classes) and data (including resources like graphics, strings, and so on). Most Android application resources are stored under the `/res`

subdirectory of the project. The following subdirectories are also available by default in a new Android project:

- `/drawable-ldpi, /drawable-hdpi, /drawable-mdpi`—These subdirectories store graphics and drawable resource files for different screen densities and resolutions. If you browse through these directories using the Eclipse Project Explorer, you will find the `icon.png` graphics file in each one; this is your application's icon.
- `/layout`—This subdirectory stores user interface layout files. Within this subdirectory, you will find the `main.xml` screen layout resource file, which defines the user interface for the one activity in this simple application.
- `/values`—This subdirectory organizes the various types of resources, such as text strings, color values, and other primitive types. Here, you find the `strings.xml` resource file, which contains all the string resources used by the application.

If you double-click any of the resource files, the resource editor launches. Remember that you can always directly edit the XML. For example, let's try editing a string resource file. If you inspect the `main.xml` layout file of the project, you notice that it displays a simple layout with a single `TextView` control. This user-interface control simply displays a string. In this case, the string displayed is defined in the string resource called `@string/hello`. To edit the string resource called `@string/hello` using the string resource editor, follow these steps:

1. Open the `strings.xml` file in the resource editor by double-clicking it in the Package Explorer of Eclipse.
2. Select the `String` called `hello` and note the name (`hello`) and value (`Hello World, HelloKindleActivity!`) shown in the resource editor.
3. Within the Value field, change the text to `Hello, Kindle Fire`.
4. Save the file.

If you switch to the `strings.xml` tab and look through the raw XML, you notice that two string elements are defined within a `<resources>` block:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello, Kindle Fire</string>
    <string name="app_name">Hello Kindle</string>
</resources>
```

The first resource is the string called `@string/hello`. The second resource is the string called `@string/app_name`, which contains the name label for the application. If you look at the Android manifest file again, you see `@string/app_name` used in the application configuration.

We talk more about project resources in Chapter 4, "Managing Application Resources." For now, let's move on to compiling and running the application.

Running and Debugging Applications

To build and debug an Android application, you must first configure your project for debugging. The ADT plug-in enables you to do this entirely within the Eclipse development environment. Specifically, you need to do the following:

- Create and configure an Android Virtual Device (AVD).
- Create an Eclipse debug configuration for your project.
- Build the Android project and launch the emulator with the new AVD.

When you complete each of these tasks, Eclipse attaches its debugger to the Android emulator (or Android device connected via USB), and you are free to run and debug the application as desired.

Managing Android Virtual Devices

To run an application in the Android emulator, you must configure an AVD. The AVD profile describes the type of device you want the emulator to simulate, including which Android platform to support. You can specify different screen sizes and resolutions, and you can specify whether the emulator has an SD card and, if so, its capacity. In this case, a slightly modified AVD for the default installation of Android 2.3.3 will suffice. Here are the steps for creating a basic AVD:

1. Launch the Android Virtual Device Manager from within Eclipse by clicking the little Android icon with the bugdroid in mini-phone  on the toolbar. You can also launch the manager by selecting Window, AVD Manager in Eclipse.
2. Click the New button to create a new AVD.
3. Choose a name for the AVD. Because you are going to take all the defaults, name this AVD `KindleFire-Portrait`.
4. Choose a build target. The Kindle Fire is based on API Level 10 (Android 2.3.3). Remember not to use a Google API version, because Kindle Fire does not support this.
5. Choose an SD card capacity, in either kibibytes or mibibytes. (Not familiar with kibibytes? See this Wikipedia entry: http://goo.gl/N3Rdd.)

> **Note**
>
> Although to mimic a Kindle Fire, you'd choose 8GiB, we recommend choosing something fairly small, because a file of the size of the SD card will be allocated on your drive each time you create a new AVD; these can add up quickly. Unless your application requires substantial storage, we recommend something like 64MiB.

6. Choose a skin. This option controls the different visual looks of the emulator. In this case, we use the effective resolution of the Kindle Fire screen of 600 pixels wide and

1004 pixels high (the default portrait resolution). Alternatively, we could create an AVD for landscape mode, where we'd need to use 1024-pixels wide and 580-pixels high. The Kindle Fire reserves some space for a soft key menu.

7. Under Hardware, change the Abstracted LCD Density to 169 and change the Device RAM Size to 512 to better emulate the Kindle Fire device characteristics.
8. Optionally, enable the Snapshot feature. This allows you to save and restore the state of an emulator session, which dramatically improves the speed with which it launches.

   Your project settings should look like what's shown in Figure 1.6.
9. Click the Create AVD button and wait for the operation to complete. This may take a few seconds if your SD card capacity is large, because the memory allocated for the SD card emulation is formatted as part of the AVD creation process. You should now see your newly created AVD in the list.

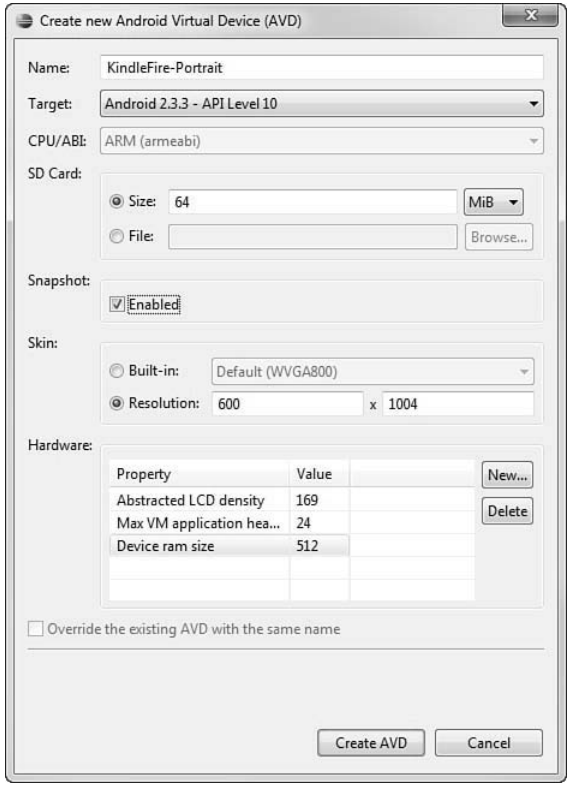


Figure 1.6 Creating a New AVD in Eclipse

Creating Debug and Run Configurations in Eclipse

You are almost ready to launch your application. You have one last task remaining: You need to create a debug configuration (or a run configuration) for your project in Eclipse. To do this, follow these steps:

1. In Eclipse, choose Run, Debug Configurations from the menu or, alternatively, click the dropdown menu next to the debug icon on the Eclipse toolbar and choose the Debug Configurations option.
2. Double-click the Android Application item to create a new entry.
3. Edit that new entry, currently called `New_configuration`, by clicking it in the left pane.
4. Change the name of the configuration to `HelloKindleDebug`.
5. Set the project by clicking the Browse button and choosing the `HelloKindle` project.
6. On the Target tab, check the box next to the AVD you created.
7. Apply your changes by clicking the Apply button. Your Debug Configurations dialog should look like what's shown in Figure 1.7.

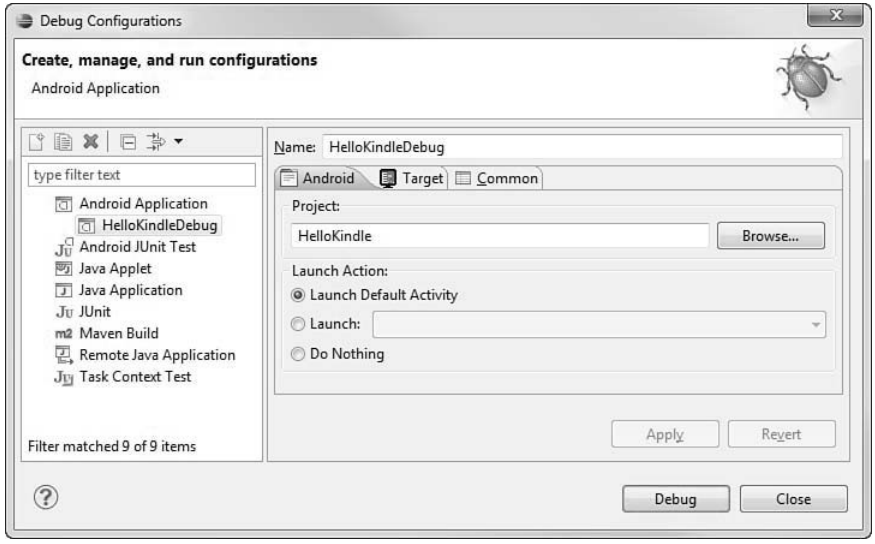


Figure 1.7 The HelloKindleDebug Debug Configuration in Eclipse

Launching Android Applications Using the Emulator

It's launch time, and your application is ready to go! To launch the application, you can simply click the Debug button from within the Launch Configuration screen, or you can do

it from the project by clicking the little green bug icon on the Eclipse toolbar. Then, select `HelloKindleDebug` debug configuration from the list.

> **Note**
>
> The first time you try to select `HelloKindleDebug` debug configuration from the little green bug dropdown, you have to navigate through the Debug Configuration Manager. Future attempts will show the `HelloKindleDebug` configuration for convenient access.

After you click the Debug button, the emulator launches (see Figure 1.8). This can take some time, so be patient.

Figure 1.8 The Android Emulator Home Screen

Now, the Eclipse debugger is attached, and your application runs, as shown in Figure 1.9.

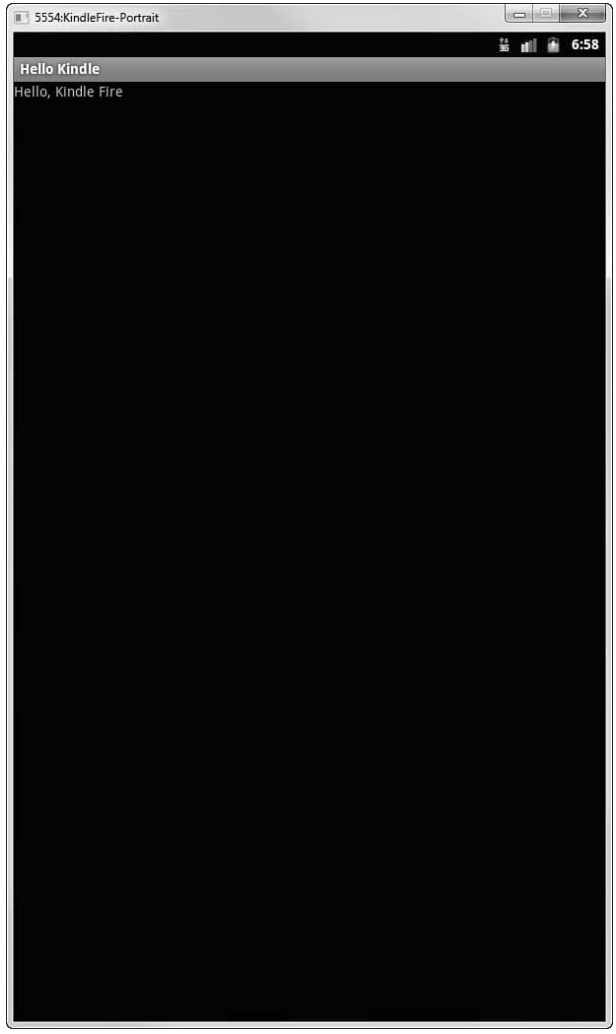


Figure 1.9 The Application Running

As you can see, the application is simple. It displays a single `TextView` control with a line of text. The application does nothing else.

The emulator's home screen doesn't look anything like the home screen on a real Kindle Fire device, because it has been redesigned by Amazon. Among other things, this means that the

emulator won't work for full application testing. You need to get a real Kindle Fire device for that.

> **Controlling the Emulator**
>
> When you create a custom AVD in this way, it will not have the keyboard and control buttons to the left of the screen, like you might be used to with the default emulators. All the commands are available through your development machine keyboard. For example, the Home key maps conveniently to the Home key. The menu key maps to F2 or page-up. Search maps to F5. Back maps to Esc. There are many more; find them in the Android documentation at http://goo.gl/5DMiI.

Debugging Android Applications Using DDMS

In addition to the normal Debug perspective built into Eclipse for stepping through code and debugging, the ADT plug-in adds the DDMS perspective. While you have the application running, quickly look at this perspective in Eclipse. You can get to the DDMS perspective (see Figure 1.10) by clicking the Android DDMS icon 

in the top-right corner of Eclipse. To switch back to the Eclipse Project Explorer, simply choose the Java perspective from the top-right corner of Eclipse.

The DDMS perspective can be used to monitor application processes, as well as interact with the emulator. You can simulate voice calls and send SMS messages to the emulator. You can send a mock location fix to the emulator to mimic location-based services. You learn more about DDMS and the other tools available to Android developers in Chapter 2, "Mastering the Android Development Tools."

The LogCat logging tool is displayed on both the DDMS perspective and the Debug perspective. This tool displays logging information from the emulator or the device, if a device is plugged in via USB.

Launching Android Applications on a Device

It's time to load your application onto a real Kindle Fire device. To do this, you need to connect the Kindle Fire to your computer using a USB data cable. Make sure that you have your machine configured for Kindle Fire debugging, as discussed in Appendix A.

To ensure that you debug using the correct settings, follow these steps:

1. In Eclipse, from the Java perspective (as opposed to the DDMS perspective), choose Run, Debug Configurations.
2. Single-click HelloKindleDebug Debug Configuration.
3. On the Target tab, change Deployment Target Selection Mode to Manual. You can always change it back to Automatic later, but choosing Manual forces you to choose whether to debug within the emulator (with a specific AVD) or a device, if one is plugged in via USB, whenever you choose to deploy and debug your application from Eclipse.

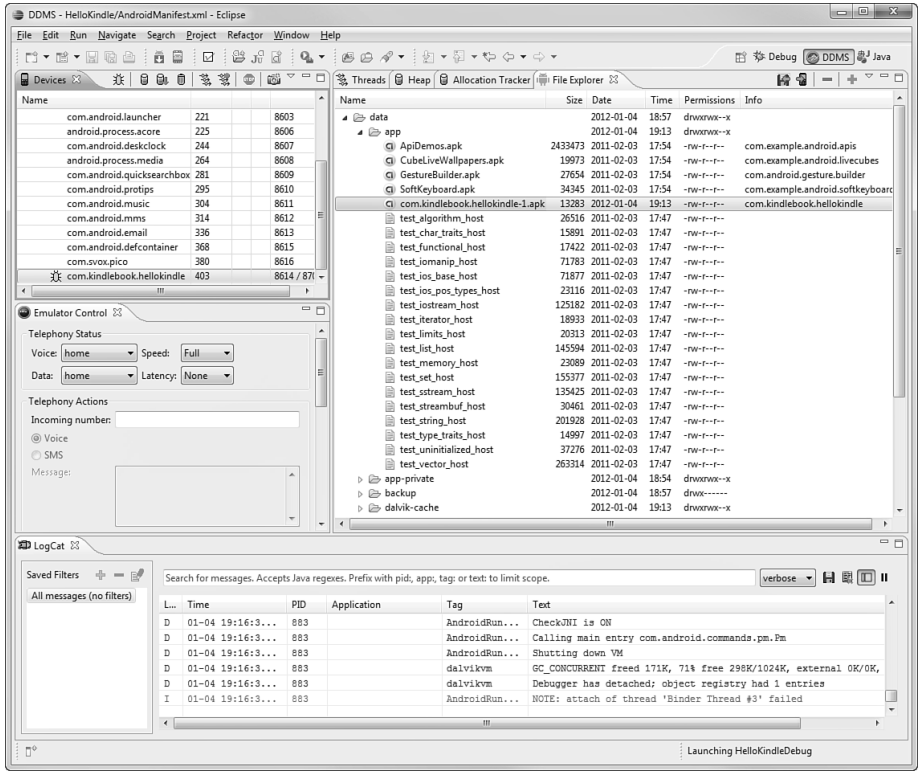


Figure 1.10 The DDMS Perspective

4. Apply your changes by clicking the Apply button.
5. Plug a Kindle Fire device into your development computer by using a USB cable.
6. Click the Debug button within Eclipse. A dialog appears (see Figure 1.11), showing all available configurations for running and debugging your application. All physical devices are listed, as are existing emulators that are running. You can also launch new emulator instances by using other AVDs that you have created.
7. Choose the available Kindle Fire device. If you do not see the Kindle Fire listed, check your cables and make sure that you installed the appropriate drivers, as explained in Appendix A.

Eclipse now installs the Android application onto your Kindle Fire, attaches the debugger, and runs your application. Your device will show a screen similar to the one you saw in the emulator. If you look at the DDMS perspective in Eclipse, you see that logging information is available, and many features of the DDMS perspective work with physical devices and the emulator, such as taking a screenshot (see Figure 1.12).

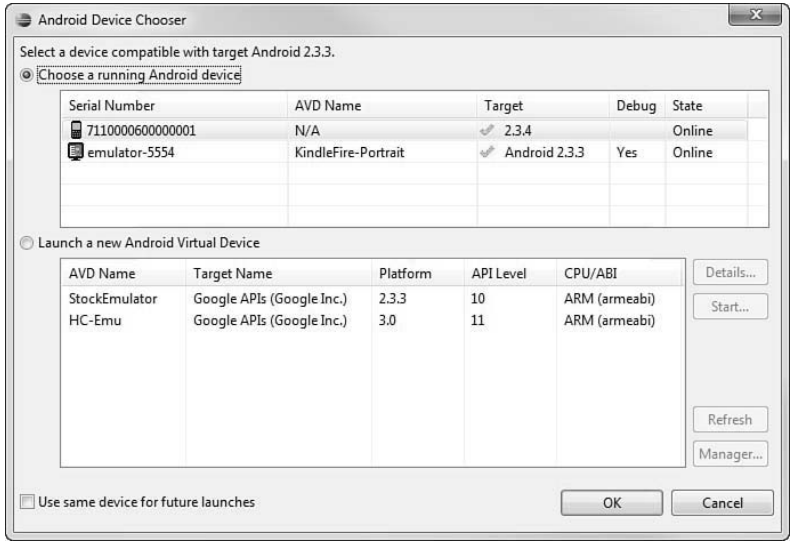


Figure 1.11 Choosing an Application Deployment Target

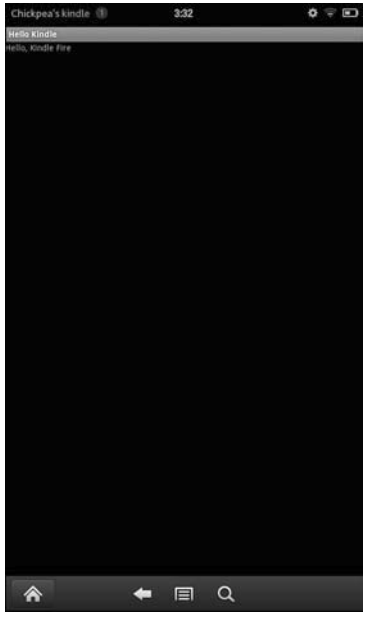


Figure 1.12 The Application Running on a Kindle Fire

> **New to Eclipse?**
>
> If you're still learning the ropes of the Eclipse development environment, now is a great time to check out Appendix B, "Eclipse IDE Tips and Tricks."

Summary

Congratulations! You are now a Kindle Fire Android developer. You have begun to learn your way around the Eclipse development environment. You created your first Android project. You reviewed and compiled working Android code. Finally, you ran your newly created Android application on the Android emulator and on a real Kindle Fire.

Exercises

1. Visit the Android website at http://developer.android.com and look around. Check out the online Developer's Guide and reference materials. Check out the Community tab and seriously consider signing up for the Android Beginners and Android Developers Google Groups.
2. Visit the Eclipse website and look around. Check out the online documentation at http://www.eclipse.org/documentation/ (http://goo.gl/fc406). Eclipse is an open source project made freely available; check out the Contribute link (http://www.eclipse.org/contribute/) and consider how you might give back to this great project in some way—either by reporting bugs or doing one of the many other options suggested.
3. Visit the Amazon Appstore Developer portal and look around. You can get started here: https://developer.amazon.com/welcome.html. While you're at it, head over to Amazon Appstore Developer Blog at http://www.amazonappstoredev.com.

Index

A

B

C

D

E

F

G

H

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X–Z