Chapter 3
Creating Your First Application
Chapter 3
Overview

- Verify that our Android environment is working correctly using a sample application
- Use the preinstalled Android Virtual Device
- Write your first application and learn about the core files and directories
- Understand the Android symbolic view and the traditional Project view
- Use Logcat for debugging and error logging
- Install and debug your application on real hardware
Note

- The Android SDK tools are updated frequently.
- For SDK updates:
- For book updates:
  - http://introductiontoandroid.blogspot.com
Testing Your Development Environment

- Working with sample applications to test your setup
- Where are the sample applications located?
  - Import the sample applications that are hosted on GitHub directly into Android Studio
- BorderlessButtons sample application
Importing the BorderlessButtons Sample into Android Studio

2. Click Import an Android code sample on the Welcome to Android Studio screen.
Importing the Sample into Android Studio (Cont’d)
3. On the Browse Samples screen, enter the keyword Borderless Buttons in the search bar to locate the sample.

– You should see two results appear in the Browse Samples listing.
– Select either of the Borderless Buttons keyword results as they both link to the same sample application.
– We selected the Borderless Buttons listing of the Design section.
– Once you have selected the sample, click Next.
Importing the Sample into Android Studio (Cont’d)
Importing the BorderlessButtons Sample into Android Studio

4. On the Sample Setup screen, you should see the Application name of BorderlessButtons already filled out and a Project location already selected.
   - Feel free to change to Project location to a directory of your choosing and then click Finish.
Importing the Sample into Android Studio (Cont’d)
Importing the `BorderlessButtons` Sample into Android Studio

- You should now see Android Studio import the `BorderlessButtons` sample application, and if everything worked correctly, there should not be any errors to appear in the Gradle Build messages.
- If there are any errors listed, you must fix them before you are able to proceed.
Importing the Sample into Android Studio (Cont’d)
Importing the BorderlessButtons Sample into Android Studio

- Now you know how to import a sample application into Android Studio.
- The sample applications are useful for learning how to write code using particular Android APIs.
- Feel free to import other samples to review how they have been developed.
- This is one of the fastest ways to learn how to write Android applications.
Using the Preinstalled AVD for Running Your Project

- In Chapter 2, “Setting Up for Development,” in the second step of the basic installation process of the “Configuring Your Development Environment” section, we recommended selecting all components for installation when installing Android Studio.

- One of the optional components for installation was a preconfigured Android Virtual Device.

- You will want to use this AVD profile to emulate when running the BorderlessButtons application.

- This AVD profile describes a default emulator configuration, and at the time of this writing, the preinstalled AVD configuration was a Nexus 5 running API 22 compatible with an x86 CPU.

- For the purposes of this example and other examples in this book, the provided AVD bundled with the default installation of Android Studio is sufficient.

- Note that the AVD configuration bundled with your installation may be different than the one used in these examples.

- You may also opt to create your own AVDs, too.
Using the Preinstalled AVD for Running Your Project

- You do not need to create new AVDs for each application, only for each device you want to emulate.
- You can specify different screen sizes and orientations, and you can specify whether the emulator has an SD card, and if it does, what capacity the card has.
- For the exact steps to configure an AVD for your BroderlessButtons project and for learning about the different configuration options, check out the section titled “Creating an AVD” in Appendix B.
Running the Application in the Android Emulator

1. With your application now open in Android Studio, press the Run icon ( ▶ ) on the toolbar.

2. A dialog will appear that prompts you to choose a device.
   - Make sure the Launch emulator option is chosen, and that the Android virtual device selected is the Nexus 5 API 22 x86, then click OK.

3. The Android emulator starts up.
   - This might take a few moments to initialize.
   - Once started, the application will be installed onto the emulator.
Running the Application in the Android Emulator (Cont’d)
Running the Application in the Android Emulator (Cont’d)

4. If necessary, to unlock the emulator, swipe the screen from bottom to top where it reads Swipe up to unlock.
Running the Application in the Android Emulator (Cont’d)
Running the Application in the Android Emulator (Cont’d)

5. The BorderlessButtons application starts and you can begin using the application.
Running the Application in the Android Emulator (Cont’d)

This sample demonstrates the use of borderless buttons, bottom button bars (OK and Cancel) and dividers to establish visual structure.
Running the Application in the Android Emulator (Cont’d)

- You can interact with the `BorderlessButtons` application through the emulator and use the application.
- You can also launch the `BorderlessButtons` application from the **All Apps** screen at any time by clicking its application icon.
- There is no need to shut down and restart the emulator every time you rebuild and reinstall your application for testing.
- Simply leave the emulator running in the background on your computer while you work in Android Studio and then redeploy using the **Run** button again.
Building Your First Android Application

- Hello World!
- The source code for this chapter is taken from the MyFirstAndroidApp found on the download site.
  - http://introductiontoandroid.blogspot.com
Creating and Configuring a New Android Project

Follow the steps in the book to create the MyFirstAndroidApp project:

1. From the Welcome to Android Studio dialog that appears after launching Android Studio, choose Start a new Android Studio project listed within the Quick Start options.
   - If you launch Android Studio and are brought into an already open project, rather than the Welcome to Android Studio dialog, be sure to close out of the project before launching Android Studio by selecting File, Close Project.
Creating and Configuring a New Android Project (Cont’d)
Creating and Configuring a New Android Project (Cont’d)

2. Choose an application name.
   - The application name is the “friendly” name of the application and the name shown with the icon on the application launcher.
   - Name the application My First Android App.
   - This will automatically create a Project location folder named MyFirstAndroidApp, but you are free to change this name and location to one of your choosing.
Creating and Configuring a New Android Project (Cont’d)

Configure your new project

Application name: My First Android App
Company Domain: introtoandroid.com
Package name: com.introtoandroid.myfirstandroidapp
Project location: C:\AndroidEnv\StudioProjects\Samples\Chapter03-FirstApp\MyFirstAndroidApp
Creating and Configuring a New Android Project (Cont’d)

3. You should also change the package name, using reverse domain name notation to com.introtoandroid.myfirstandroidapp.
   – Learn more about reverse domain notation here:
     • http://en.wikipedia.org/wiki/Reverse_domain_name_notation
   – To do this, edit the Company Domain field to something like introtoandroid.com.
   – You will see the Package name change automatically when you modify the Company Domain field.
   – You may also edit the Package name field directly by clicking the Edit link located to the far right of the Package name listing.
   – Once you are finished, click Next.
Creating and Configuring a New Android Project (Cont’d)

4. On the Target Android Devices screen seen on the next slide, you should have the Phone and Tablet option selected, in addition to choosing the Minimum SDK you want your application to support.

– At the time of this writing, Android 4.0.3 API Level 15 is the default Minimum SDK option preselected.
– This will allow you to support 94.0% of all devices compatible with Google Play Store applications.
– You are free to choose a different Minimum SDK, but for this application, keep it set to the API 15: Android 4.0.3 (IceCreamSandwich), if it is not already selected.
– You are also able to choose other form factors for your application to support, such as Wear, TV, Android Auto, and Glass, in addition to selecting a Minimum SDK for most of those options, but we are only interested in Phone and Tablets.
– Click Next.
Creating and Configuring a New Android Project (Cont’d)
5. On the Add an activity to Mobile screen, you are able to select what type of Activity you want to add to your application from a few common options.

- You are also free to choose Add No Activity.
- For this example, keep the Blank Activity option selected, as shown in the figure on the next slide.
- Choose Next.
Creating and Configuring a New Android Project (Cont’d)
Creating and Configuring a New Android Project (Cont’d)

6. The Customize the Activity screen allows you to provide an Activity Name.
   - Name the Activity MyFirstAndroidAppActivity.
   - You will notice the Layout Name, Title, and Menu Resource Name fields change as you edit the Activity Name field.
   - You are now ready to create your application.
   - Finally, click the Finish button to create the application.
Creating and Configuring a New Android Project (Cont’d)
Creating and Configuring a New Android Project (Cont’d)

7. Android Studio may take a short while to build your project.
   – Once it is complete, your first application will display with your layout file open and ready for editing.
   – See the next slide that shows a figure of what you should see.
Creating and Configuring a New Android Project (Cont’d)
Understanding the Android Symbolic View and the Traditional Project View

- When you created your first application in Android Studio, your project was opened in the Android project view.
  - The project hierarchy represented in the Android project view is only a symbolic representation of file and directory names, and not their actual filesystem location.
  - This Android view is the default view for managing your projects with Android Studio.

- If you prefer to view the actual filesystem location of your project files and directories, you may choose to switch from the Android view to the traditional Project view.
  - You can do so by clicking on the Android view drop-down, and selecting the Project view.

- The figure on the next slide details the difference between the two views.
Understanding the Android Symbolic View and the Traditional Project View (Cont’d)
Core Files and Directories of the Android Application

- build/
- libs/
- src/
- src/main/AndroidManifest.xml
- src/main/java/
- src/androidTest/
- src/main/res/
- src/main/res/drawable
- src/main/res/layout
- src/main/res/layout/activity_my_first_android_app.xml
- src/main/res/menu
Core Files and Directories of the Android Application

- `src/main/res/menu/menu_my_first_android_app.xml`
- `src/main/res/mipmap-*`
- `src/main/res/values*`
- `src/main/res/values/dimens.xml`
- `src/main/res/values/strings.xml`
- `src/main/res/values/styles.xml`
- `src/main/res/values-w820dp/dimens.xml`
- `proguard-rules.pro`
- `build.gradle`
- `app.iml`
- `.gitignore`
Running Your Android Application in the Emulator

- Now you can run the MyFirstAndroidApp project using the following steps found on the next few slides.
Running Your Android Application in the Emulator (Cont’d)

1. With the Run/Debug Configuration named `app` already selected, click the Run icon ( ▶ ) on the toolbar.
   
   - Here is the `app` configuration selected:
Running Your Android Application in the Emulator (Cont’d)

2. You are now prompted to Choose a running device.
   - The default emulator you launched for the previous example should be listed as a running device.
   - If it is not already running, select Launch emulator and choose the appropriate AVD if it is not already selected.
   - Then click OK.
Running Your Android Application in the Emulator (Cont’d)
Running Your Android Application in the Emulator (Cont’d)

3. If not already started, the Android emulator starts up, which might take a moment.
4. Unlock the emulator if it is locked.
5. The application starts, as shown in the figure on the next slide.
Running Your Android Application in the Emulator (Cont’d)
Running Your Android Application in the Emulator (Cont’d)

6. Click the Back button in the emulator to end the application, or click Home to suspend it.

7. Click the All Apps button found in the Favorites tray to browse all installed applications from the All Apps screen.
8. Your screen should now present all the applications installed on the device.
   - Click the My First Android App icon to launch the application again.
Running Your Android Application in the Emulator (Cont’d)
Debugging Your Android Application in the Emulator

- Before going any further, you need to become familiar with debugging in the emulator.
- To illustrate some useful debugging tools, let’s manufacture an error in the *My First Android App.*
In your project, edit the source file called MyFirstAndroidAppActivity.java.

Create a new method called forceError() in your class and make a call to this method in your Activity class’s onCreate() method.

The forceError() method forces a new unhandled error in your application.
public void forceError() {
    if(true) {
        throw new Error("Whoops");
    }
}
Debugging Your Application in the Emulator (Cont’d)
Debugging Your Application in the Emulator (Cont’d)

- Shut down the application but keep the emulator running.
- Now it’s time to debug.
- You can Debug the MyFirstAndroidApp application using the following steps:
  1. With the Run/Debug Configuration named app selected, click the Debug icon ( ) on the toolbar.
  2. Continue as you did when launching the Run configuration and choose the appropriate emulator, unlocking it if necessary.
- It takes a moment for the debugger to attach.
Debugging Your Application in the Emulator (Cont’d)
Debugging Your Application in the Emulator (Cont’d)

- In Android Studio, use the **Debugger** tab to view breakpoints, step through code, and watch the **Logcat** logging information about your application.
- This time, when the application fails, you can determine the cause using the debugger.
- If you allow the application to continue after throwing the exception, you can examine the results in the **Debugger** of Android Studio.
Debugging Your Application in the Emulator (Cont’d)
Debugging Your Application in the Emulator (Cont’d)

- Specifically, there’s an `uncaughtException()` error of `java.lang.Error: Whoops`
- Back in the emulator, click the Force Close button.
- Now set a breakpoint on the line that starts with the `throw` statement of the `forceError()` method by clicking inside the column to the left side of the line of code so that a red circle appears.
Debugging Your Application in the Emulator (Cont’d)

- In the emulator, restart your application and step through your code.
- You see that your application has thrown the exception, and then the exception shows up in the Debugger.
- Expanding its contents shows that it is the “Whoops” error.
- This is a great time to crash your application repeatedly and get used to the controls.
Adding Logging Support to Your Android Application

- Before you start diving into the various features of the Android SDK, you should familiarize yourself with logging.
- Logging is a valuable resource for debugging and learning Android.
- Android logging features are in the Log class of the android.util package.
Adding Logging Support to Your Android Application (Cont’d)

- **Logging Methods**
  - Log.e()
  - Log.w()
  - Log.i()
  - Log.d()
  - Log.v()

- **Add the Log class using the import statement:**
  - import android.util.Log

- **Add a variable for collecting your debug messages:**
  - private static final String DEBUG_TAG = “MyFirstAppLogging”

- **In the onCreate() method add the log statement:**
  - Log.i(DEBUG_TAG, "In the onCreate() method of the MyFirstAndroidAppActivity Class");
Adding Logging Support to Your Android Application (Cont’d)
Debugging Your Application on Hardware

- You have mastered running applications in the emulator.
  - Now let’s put the application on real hardware.
  - Connect an Android device to your computer via USB and relaunch the application using the Debug option.
  - You should now see a real Android device listed as an option in the Choose Device dialog.
Debugging Your Application on Hardware (Cont’d)
Debugging Your Application on Hardware (Cont’d)

Allow USB debugging?

USB debugging is intended for development purposes only. Use it to copy data between your computer and your device, install apps on your device without notification, and read log data.

CANCEL  OK

Process Stats
Geeky stats about running processes

Debugging
USB debugging
Debug mode when USB is connected
Debugging Your Application on Hardware (Cont’d)

Allow USB debugging?

The computer's RSA key fingerprint is:

Always allow from this computer

CANCEL  OK
Debugging Your Application on Hardware (Cont’d)
Chapter 3

Summary

- We have learned how to add sample applications and create new projects of our own.
- We have learned how to build, run, and debug our projects.
- We have learned the difference between the Android symbolic view and Project view of Android Studio.
- We are now able to log errors.
References and More Information

- Android Training: “Getting Started”:
  - http://d.android.com/training/index.html
- Android SDK Reference regarding the application Activity class:
- Android SDK Reference regarding the application Log class:
- Android Tools: “Using Hardware Devices”:
- Android Tools: “Managing Projects Overview”:
- Android Samples: “Samples”:
  - http://d.android.com/samples/index.html