

digit **FastTrack**

YOUR HANDY GUIDE TO EVERYDAY TECHNOLOGY

To **ANDROID STUDIO**

- 
- INSTALLATION AND FIRST RUN
 - UNDERSTANDING THE USER INTERFACE
 - EXPLORING THE USER INTERFACE
 - WORKING WITH PROJECTS
 - DEVELOPING YOUR OWN APP
 - ESSENTIAL PLUGINS/TOOLS
 - EXPORTING AND IMPORTING FROM GITHUB



www.digit.in/forum

Join the forum to express your views and resolve your differences in a more civilised way.

digit.in
FORUM

Post your queries and get instant answers to all your technology related questions



One of the most active online technology forums not only in India but world-wide

**JOIN
NOW**

 **digit.in**



ANDROID STUDIO

powered by

digit
YOUR TECHNOLOGY NAVIGATOR

CHAPTERS

ANDROID STUDIO

JUNE 2016

06

PAGE

Installation and first run

Before you begin twiddling your fingers crunching code, you'll have to set up Android Studio. Here's how.

12

PAGE

Understanding the User Interface

Android Studio has come a long way towards becoming more user friendly. Here's a low down about the different UI elements that you'll be working with.

24

PAGE

Exploring the User Interface

Now that you've figured out what the different elements of the Android Studio UI, here's how you can go about configuring them to your liking.

CREDITS

The people behind this book

EDITORIAL

Executive Editor

Robert Sovereign-Smith

Managing Editor

Siddharth Parwatay

Technical Editor

Jayesh Shinde

Senior Reviewer

Mithun Mohandas

Writers

Anand N

Ashwin Krishnan

Kshitij Bedekar

Varad Chaudhari

Vishal Kumar More

DESIGN

Creative Director

Tharakaram G

Sr. Art Director

Anil VK

Associate Art Director

Anil T

Visualiser

Baiju NV

30
PAGE

Working with projects

Having familiarised yourself with the nuts, bolts, buttons and tabs of Android Studio 2, it's time to start some real work - here's a quick primer on creating and modifying projects, and getting started towards making your own Android app.

39
PAGE

Developing your own app

Hopefully, you've got the hang of Android Studio by now. So let's make your very first app using the software.

72
PAGE

Essential Plugins, Libraries, Tools and Resources

There's a plugin for everything and why shouldn't there be? After all they make your job easier. Here are some tools that you shouldn't miss out on.

88
PAGE

Exporting and Importing from GitHub

There's no denying that version control plays a huge part in a software's development cycle. Here's how you can use GitHub with Android Studio.

© 9.9 Mediaworx Pvt. Ltd.

Published by 9.9 Mediaworx

No part of this book may be reproduced, stored, or transmitted in any form or by any means without the prior written permission of the publisher.

June 2016

Free with Digit. If you have paid to buy this Fast Track from any source other than 9.9 Mediaworx Pvt. Ltd., please write to editor@digit.in with details

Custom publishing

If you want us to create a customised Fast Track for you in order to demystify technology for your community, employees or students contact editor@digit.in

readywhere



COVER DESIGN: ANIL T

Apps made easy

Every other person you meet has an idea for an app. And if you lend them an ear, you'll be surprised that many a brilliant ideas lurk in limbo, stuck in the minds of those who cannot code. Or perhaps aren't comfortable with coding. While we don't aim to teach you how to code, we do hope that this FastTrack helps you start off on your Android App development journey.


And it isn't a hard journey. Google sure has gone out of their way to make the entire development process as easy as possible. With Android Studio 2.1 (2.2 by the time you'll be reading this) they've bridged the gap quite well. If you've ever used an IDE in your life then you'll feel right at home with Android Studio. And from the time of its inception, to the time we wrote this FastTrack, Android Studio has tremendously improved. Android being the most popular platform in the world, running on more than 1.4 billion active devices as of September 2015 enjoys the attention of a lot of developers. Not to mention, the attention of a heck lot more budding entrepreneurs, including you. *cue Uncle Sam-esque poster*.

So with the aim of transforming more of those brilliant ideas into brilliant apps, we bring you the FastTrack to Android Studio. It starts off quite simple with installation instructions that'd be passed off as pedestrian and then moves onto exploring the User Interface. Once you're comfortable with the UI, you'll plunge straight into developing an app. To be precise, this is a simple RSS reader that you'll be developing. You can find the sample app over at <https://goo.gl/3dYt8D>. Feel free to use the different elements to customise the app to your liking and perhaps make it even better and slap on a few more features. Do drop us an email at editor@digit.in with your creations.

And we don't just stop there, through the course of writing this FastTrack we noticed that Android Studio seems to be missing some element

or the other. Each of our authors felt a certain shortcoming which they found ways of overcoming. What we mean to say is that they all found plugins to help with the development process or to add features that are otherwise not easy to code for a beginner. So we compiled an entire list and dropped it in the booklet. We hope they're as useful to you as they were to us.

And lastly, GitHub is one of the most, if not the most popular version control system and it felt imperative that we include the same in the Fast-Track. Trust us, using a software version system is incredibly helpful, especially in a collaborative environment.

So what are you waiting for? Jump right in! 



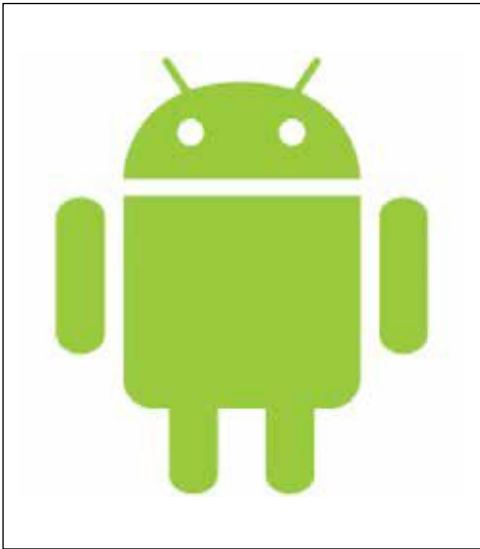
INSTALLATION AND FIRST RUN

Before you begin twiddling your fingers
crunching code, you'll have to set up
Android Studio. Here's how.

For many years, one of the key reasons why the Android platform has performed so well is because of the vast range of development tools that are available for the platform. From beginners trying to make their first app to large-scale companies working on high budget projects, Android has been their primary target market. Android Studio has been freely available since 2013 under the Apache License and is available for Windows, Mac OS X and Linux platforms. Since its inception, it has officially replaced Eclipse Android Development Tools (EADT) as the primary Integrated Development Environment (IDE) for native Android

app development. After a recent upgrade over security concerns, we take a look at the basics of setting up Android Studio on your device and installing it to develop your own Android based applications.

To setup Android Studio, you will need a device with Microsoft® Windows® 8/7/Vista/2003 (32 or 64-bit) or Mac® OS X® 10.8.5 or higher, up to 10.9 (Mavericks) or a GNOME or KDE desktop. Fairly straightforward expectations from a 21st century reader with a Digit subscription, we presume. Before getting to Android Studio, there are a few system prerequisites. You will need Java Development Kit (JDK) installed. In the typical fashion of programmers and developers who have a penchant for a lack of clarity and fancy

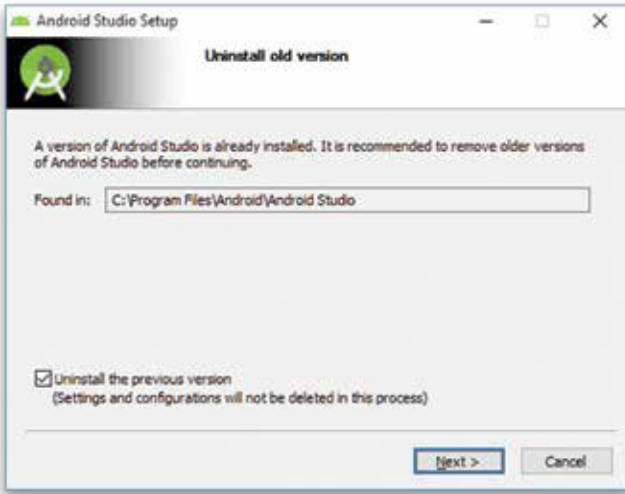


Android Studio: Feed it an Apple

sounding abbreviations, you may also find JDK being called Software Development Kit (SDK) or Java Standard Edition (JSE). The installation files can be found directly on the Oracle website. Select the highest possible version of JSE, but make sure you select the option of picking the entire JDK, not the JRE which is just the runtime files. Download the setup file and the documentation, if you want it, on to your computer. Run the file, pick a target directory

and when it prompts you to pick the features you want to install, select both JRE and Development Tools. Go grab a cup of coffee (Java Arabica to be poetic?) and by the time you return you should be ready to properly get down to installing Android Studio.

Make sure that your JRE is at least version 6 and that the JDK is at least a version 5. These should be alright by default if you install the latest package from the Oracle website. For first time installers you can directly download



Installing Android Studio

the Android Studio setup file from <http://dgit.in/AndroidStudioDL>. The installation software will ask for the path where you have the JDK installed. For most instances, Android Studio is smart enough to recognise the JDK automatically but you may need to do this manually for certain installations. Once this is done, it becomes as simple as installing VLC media or Sticky Notes, just pick a directory with at least 2.7 GB of free space (ideally at least 5 GB, to not have performance issues) and the setup will take it from there. Just so that you know, Android Studio takes up 512 MB of RAM space.

After the installation is done it shows you the window we've shown on the page to the right. Here's where you can start your first Android Project. Click on the button that says 'Start a new Android Studio project'. Let's assume, for this article, that we want to build a smart RSS reader for Dgit magazines that works for all mobile devices that have at least Android Jellybean installed. In the first page, we enter the name of the application and the domain name of the holding company and also enter a local file location from which we develop our project. After entering the application name you need to select the form factors. In our example, we select the option 'Phone and Tablet' and the Minimum SDK that corresponds to Jellybean. At this point, we already get an intuitive sense of

how Android Studio has taken the traditionally complicated field of app development and made it accessible to the amateur developer by making the interactions as graphically intuitive as possible. This new Graphic User Interface (GUI) is one of the key attractions of the new versions. By clicking on mobile/tablet, it also gives you a variety of templates you can use as the basis of your application. Once you have gone through all of these pesky details, it finally opens up the open development tool to begin writing the code for the application.

Android Studio has made the creation of new projects and modules much easier. Click on **File>New Module** to create the actual Graphical User Interface of the app. This creates a new Module with a few folders and other files. Starting with the “manifests” folder which holds the **AndroidManifest.xml** file. This file holds basic information including the name of the app that shows up on the device and all permissions that you define. Android Studio offers independently manoeuvring different UI components and it is smart enough to know which components are completed and doesn’t execute them again to save runtime. This ability of Android Studio is due to the in-built system Gradle. Gradle is a build automation tool that sets out to be easier than traditional XML based project configurators and was made for large



Starting a new project



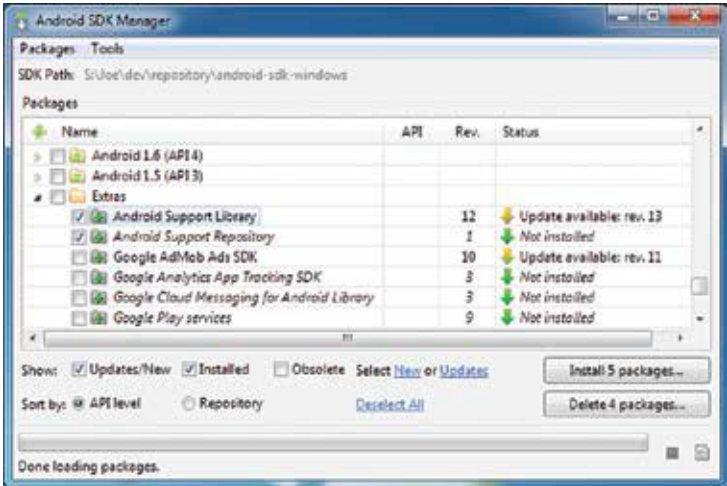
Virtual device for testing your prototype

projects. Gradle provides an easy way to configure app details including build version and SDK version.

To start developing the application, we need a means to test it. To do so, we create what is known as a Virtual Machine. This is a computer run simulation that recreates an Android device on your system. In our example, it'll show a sample phone that has all the characteristics of a machine that has Android Jellybean installed on it. To create such a device, click on the 'Create Virtual Device' button and Android Studio will, by default, set up a device with the settings we had initialised earlier. This will be visible as a mobile/tablet on the right hand side of the user interface. The idea behind this is to try and simulate the experience of the end user with the app under development so that the developer can customise the app as required.

An important feature of the Android Studio setup is the vast directory of libraries that are available. Depending on the need of the user and the demands of the particular app that is being developed as well as the platforms that you want to support, the developer can choose to invoke certain libraries and the built in functions that they have.

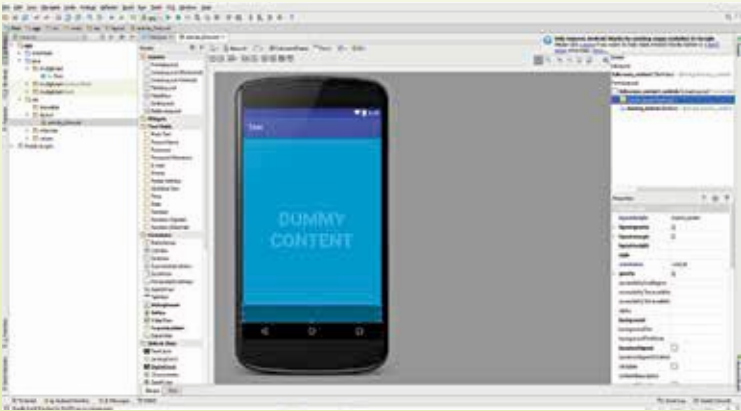
In order to use a Support Library, you must modify your application's project's classpath dependencies within your development environment. You must perform this procedure for each Support Library you want to use. To add a particular library, open the Android SDK manager, scroll to the packages column and select the Android Support Library option. Before installing a support library, consider the features that you want and the platform on which you wish to run your app and pick an appropriate



Library manager

library. Some Support Libraries contain resources beyond compiled code classes, such as images or XML files. For example, the **v7 appcompat** and **v7 gridlayout** libraries include resources. To add support libraries, open the **build.gradle** file and add the support library to the dependencies section. There are plenty of tutorials and sample codes available online on android development blogs and websites which take you on a step-by-step guide of the kinds of libraries, the functionalities they have, the platforms they support and the way you would go about adding them.

Hopefully, this chapter will have acquainted you with the basics and you now have Android Studio set up on your computer. We'll get to the libraries you'll need for your app in a later chapter. For now, you can now go ahead and start exploring the brilliant options that it offers. Read on to find out how to go about designing your first ever Android app. Have fun, and may your app be the next Flappy Bird. **d**



UNDERSTANDING THE USER INTERFACE

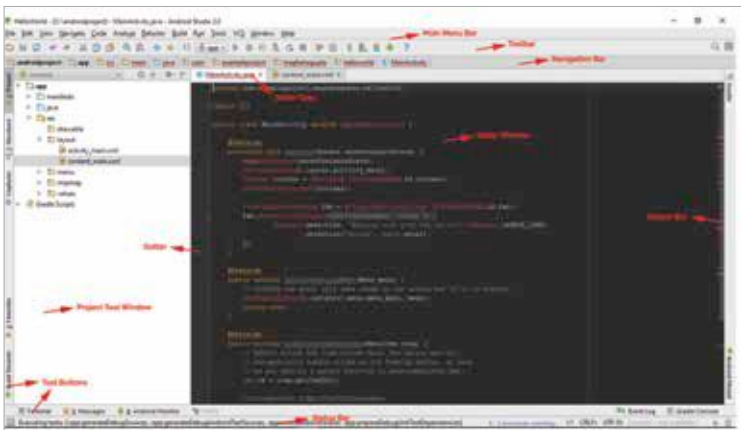
Android Studio has come a long way towards becoming more user friendly. Here's a low down about the different UI elements that you'll be working with.

A superior understanding of your development environment will not only help maximise productivity and workflow speed, but also give you the confidence to take advantage of various facilities and features provided by your IDE. So to give you a better picture of what Android Studio has to offer, we'll start by taking a look at the user interface of an Android developer's favourite IDE.

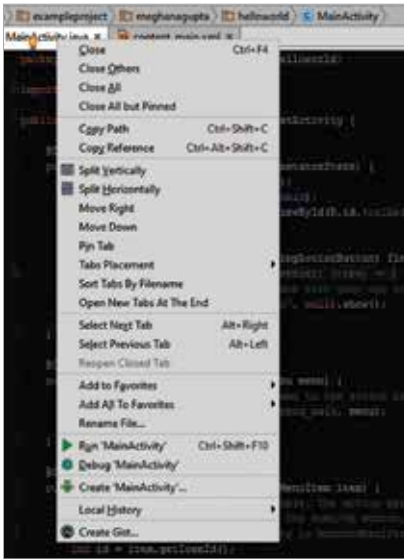
The most visible pane in Android Studio is the Editor window. It is pinned to the top at all times, and is where all code is edited. Above it lies the Editor Tabs bar, containing document tabs, which is used to switch between various documents that need to be edited. Android Studio tries to anticipate which files you're likely to start editing, and then opens them automatically as tabs in the Editor upon completion of the New Project Wizard (encountered when setting up a new Project). A convenient shortcut that can be used to shift between the tabs is Alt+Right-Arrow | Ctrl+Right-Arrow or Alt+Left-Arrow | Ctrl+Left-Arrow. Right-clicking on any of the document tabs opens a new context menu, and any action chosen from this menu applies only to the tab on which we right-clicked.

To the left of the Editor window is the gutter, which is primarily used to convey information about the code. At any time, any code-block in the Editor can be expanded by clicking on the '+' sign enclosed within a square, and collapsed by clicking on the '-' enclosed within the pentagon. Other symbols have different meanings, and can be explored when you add more code to the document.

The marker bar at the right side of the Editor window contains markers pointing to important snippets in the document, which are generally warnings and compile-time errors that are present in the code. The length of the marker bar is representative of the length of the document, and clicking on any one of the markers jumps you to that location in the file.

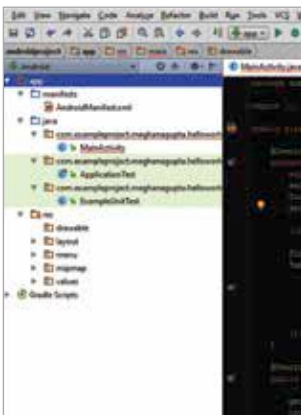


The Android Studio User Interface

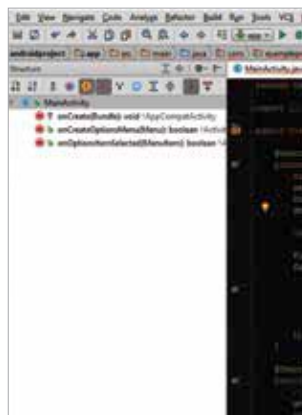


To get an overview of all the packages, directories, and files in your project, we use the Project Tool window. Its ease of use stems from the brevity of the view and the easy-to-understand tree structure in which it is represented. The window has three modes; Project, Packages, and Android. By default, Android Studio will set the mode to Android. Android and Project are the most useful modes, though the Android mode may hide certain directories from you.

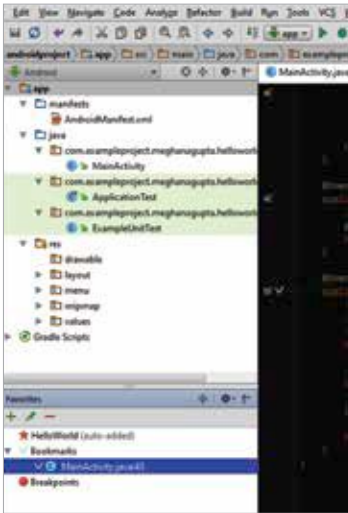
Clicking on the 'Structure' tab present on the left of the Project Tool window switches the view to the Structure Tool window. This panel displays the hierarchy of elements in your file, where elements are fields, methods and inner classes. When the Editor is displaying a Java source file such as `MainActivity.java`, the Structure tool window displays a tree of elements such as fields, methods, and inner classes. This view



Project Tool Window



Structure Tool Window

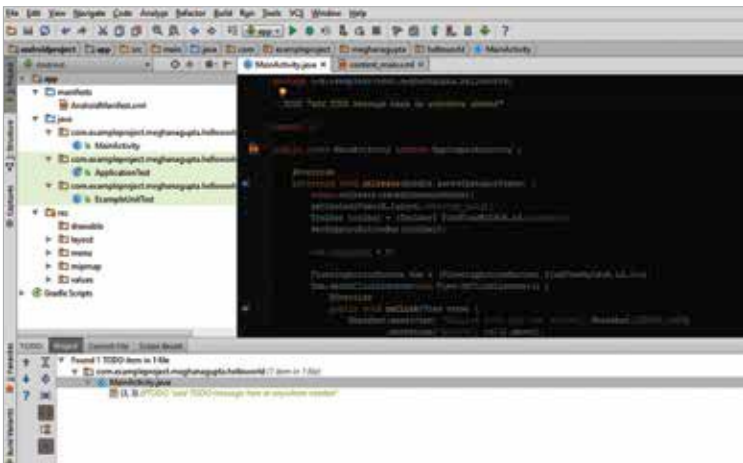


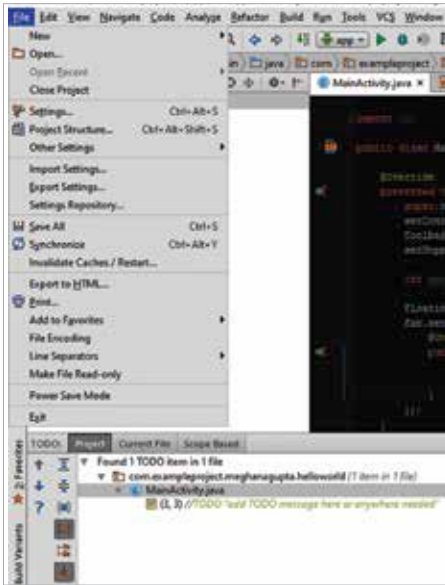
becomes increasingly useful when the complexity and length of code increases, since clicking on any of the methods scrolls the view and cursor to the corresponding element. Again, all views are represented in a nested tree structure.

Android Studio provides a way to quickly reference and open files that are frequently used. This makes streamlining the workflow easier when projects get larger and the number of documents become difficult to keep track of. The Favourites Tool window comes into play to help group related files that are useful. To add a file to the list of Favourites, you

have to right-click on the on the relevant tab and choose 'Add to Favourites', and then the name of the list to which you wish to add the document. You can also click on 'Add All to Favourites', and again the name of the favourites list.

Just as the Favourites window allows you to navigate immediately to any particular file or groups of files, bookmarks allow you to navigate immediately to any particular line in a file. Pressing F11 creates or removes





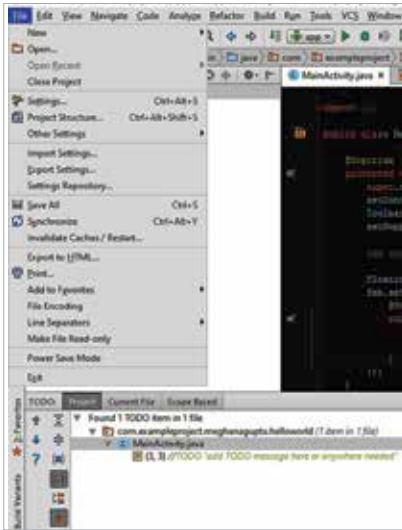
a bookmark at the point where the cursor was last positioned.

To view the list of Favourites, click on the Favourites tab to the left of the Project Tools/Structure Tools window.

The TODO Tool Window is a useful feature for when collaborating within a large group of people. To draw the attention of a fellow collaborator to a particular section or action, simply start a comment to a particular section or action, simply start a comment (comments start with ‘\’) followed by ‘TODO’, and then the comment.

For example, ‘//TODO add method telling time’. Clicking a TODO in the TODO tool window immediately jumps to that TODO in your source code.

At the bottom of the window is the status bar, which provides detailed information about the running status of the documents and the running processes. The leftmost icon is used to toggle margins, and hovering the cursor over it positions a new context menu to toggle other tool windows. The next left portion of the bar provides status messages, provides feedback and displays any information about concurrently running processes. Clicking on this area opens the Event log. The next portion of the bar displays the location of your cursor in the Editor in the ‘line: column’ format. Clicking here opens a dialog box, in which entering a particular line number allows you to navigate directly to that portion. The next area, the line separator area, displays the format of the carriage returns used in your text files. The text format area, which comes after the line separator area, describes the text encoding used for your source files. The default is UTF-8. Allowing you to toggle between read/write and read-only is the file access indicator button. An unlocked icon means that the current file in the Editor has read/write access. A lock icon means that the current file in the Editor is read-only. You can toggle these settings by clicking the indicator’s icon. The



Highlighting Level button activates a dialog box with a slider that allows you to set the level of highlighting you want to see in your code.

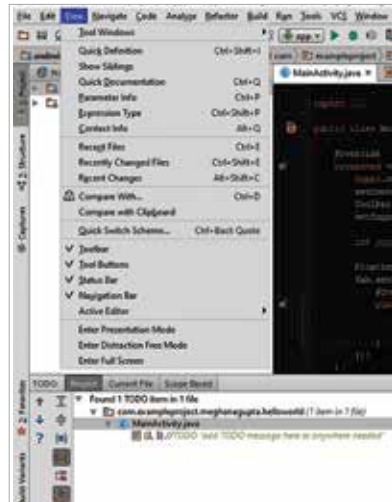
Going back up to the navigation bar, we see that it is useful for navigating your project's assets without having to resort to the Project Tool windows. The bar displays a horizontal chain of folders that are representative of the nested subdirectories and folders, with the chain starting from the root directory.

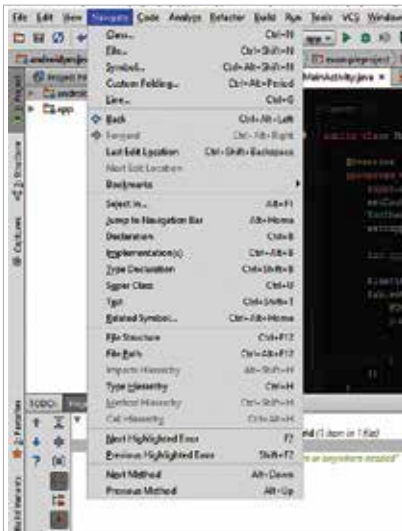
Above the navigation bar is the Toolbar. While most of

the buttons in this toolbar can be accessed from other menus and keyboard shortcuts, the toolbar has been provided nonetheless, for beginners unaware of shortcuts and other ways to access the options. The regular options for copying, pasting, undoing and redoing are provided, along with the 'Save' and 'New Project' buttons. Some new buttons are the 'Run App' and 'Debug App' buttons, and others for the ADK Manager and SDK Manager.

The one menu whose view cannot be toggled off is the Menu Bar. Again, though many people are aware of keyboard shortcuts and other Tool bars to make navigating and avoiding the endless jumble of the Menu bar easier, knowing all the options available for use is never a bad thing.

Starting off with the File menu, the standard array of features is available. Some new options are the 'Synchronize' option, that is





used to reload the dependencies specified in the Gradle build file, and is thus used whenever a change is made in the document. Other options are accessible through keyboard shortcuts and Tool windows visible on the main window. The 'Settings Repository' option allows you to sync your IDE settings with a specific repository and sync them across installs.

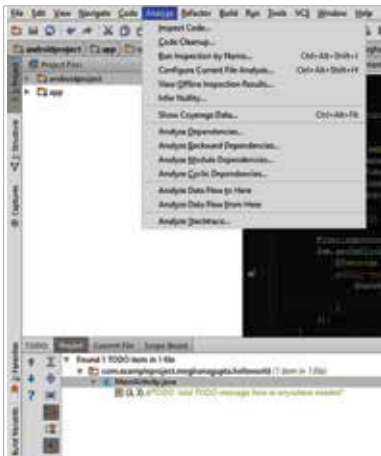
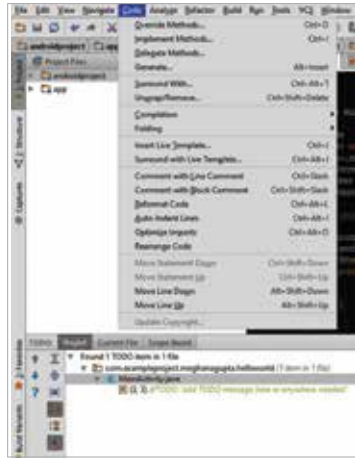
Under the Edit tab, the options available are most of those visible in the Toolbar menu. A few additions are the 'Toggle Case' and 'Fill Paragraph' options. The feature names pretty much give away their function. For users used to the column selection mode in other IDEs, that can be enabled by choosing the 'Column Selection Mode' available in the Edit tab.

Under the View tab, the 'Tool Windows' option can be used to toggle the view for various Tool windows. 'Quick Documentation' is used to view tooltip like documentation for methods, classes and other references within the Android API. Other options are useful for opening recently viewed files ('Recent Files') and recently changed files ('Recently Changed Files'), changing the window layout to other options that are margin-free and Tool window-free and provide only the Editor in the working area ('Enter Distraction Free Mode'), and moving to a full window view ('Enter Full Screen').

The Navigate tab is a more verbose version of the navigation tab visible on the main menu. The regular aside, the option to provide custom collapsing for code blocks is available through the 'Custom Folding' option, and clicking on the 'Type Hierarchy' button opens up a separate button that pre-

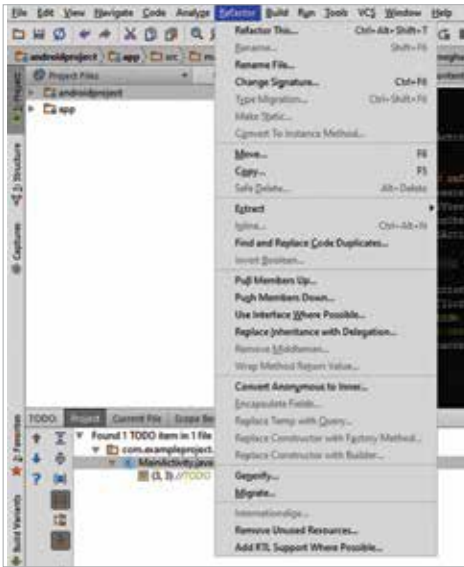
sents the inherited class hierarchy for the method currently pointed to by the cursor (within the Editor window) in a nested tree format. The ‘Next Highlighted Error’ and ‘Previous Highlighted Error’ navigates through the errors in the code, and is useful for large lengths of code. However, simply using the markers available in the Markers tab is much faster and easier.

Selecting the ‘Override Methods’ and ‘Implement Methods’ options in the Code Menu is akin to referencing documentation to decide which methods to override or implement. The options for code completion and folding are also available, and work well to help automate your code generations. Other options like ‘Reformat Code’, ‘Auto-Indent Lines’, ‘Optimise Imports’ and ‘Rearrange Code’ are useful to improve the visual look of the code and help increase readability. An extremely useful feature provided by Android Studio, that helps reduce time wasted writing repetitive boilerplate code, is Live Templates. Live Templates are shortcuts displayed as code-completion options that, when selected, insert a code snippet that you can tab

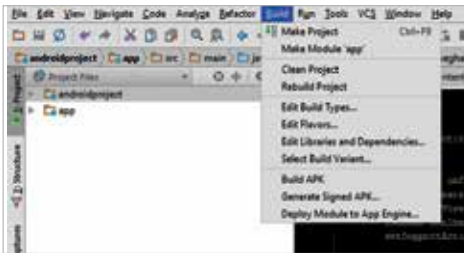


through to specify any required arguments. Pressing Tab while writing code, or using the ‘Insert Live Template’ option available in the Navigation Menu are useful for working with Live Templates.

Many features under the Analyze tab are useful for large-scale production level code. Using the ‘Inspect Code’ feature runs a series of checks through the code using Android Lint, a static code analysis tool that checks your Android project source files for potential bugs and optimization



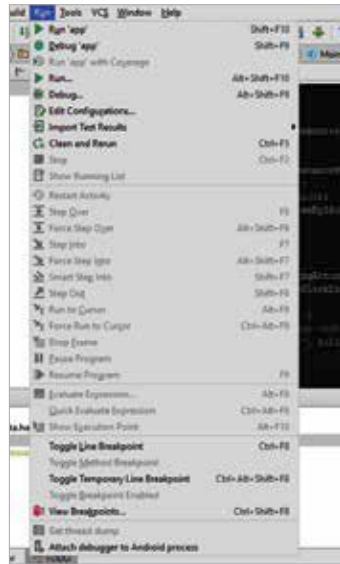
improvements for correctness. Any potential weaknesses are identified and displayed in a window. Running the ‘Code cleanup’ feature provides a similar suggestion as the previous feature. The option to ‘Analyze Dependencies’, ‘Analyze Backward Dependencies’, ‘Analyze Module Dependencies’ and ‘Analyze Cyclic Dependencies’ displays the results of analyzing dependencies, backward and cyclic dependencies.



Another production-level feature that is an essential component of many a professional developers’ workflow is the refactoring process, which is basically restructuring existing computer code without

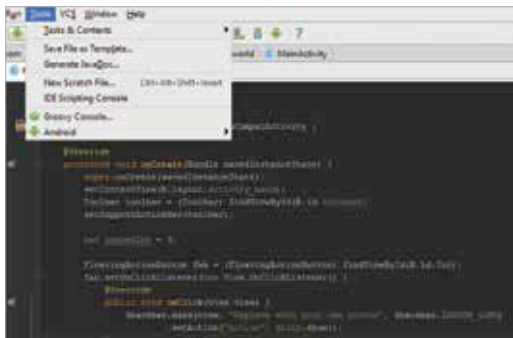
changing its external behaviour. Android Studio has taken some cues from IntelliJ and has provided an option for convenient refactoring, right in the Refactor menu. Refactoring improves nonfunctional attributes of the software. Advantages include improved code readability and reduced complexity; these can improve source-code maintainability. Since inherited classes are known to increase the chances of spaghetti code and cyclic dependencies, an option to replace inheritances with delegations is also provided (use ‘Replace Inheritance with Delegation’). Other options include the ability to eliminate inheritances and use interfaces instead (‘Use Interface Wherever Possible’), and the conversion of anonymous classes to inner classes (‘Convert Anonymous to Inner’).

The Build tab is handy after most of the code has been written and it is time to finally build the app and run it. The most relevant features on this menu are the ‘Make Project’ feature, used when the source files in the entire project that have been modified since the last compilation are compiled, the ‘Make Module’ feature which is used when all the source files that have been modified since the last compilation in the selected module, as well as in all the modules it depends on recursively, have been compiled and finally the ‘Rebuild Project’ feature, where all the source files in the project are recompiled. Then again, all this can be avoided by simply clicking on the ‘Run

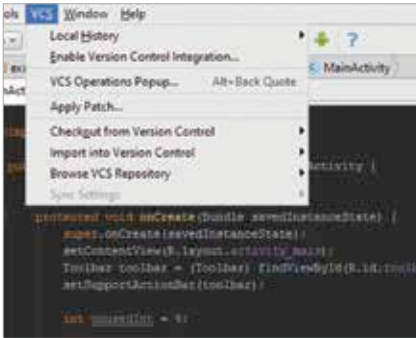


App’ button on the Toolbar (the button is denoted by a green ‘play’ icon) or a similar button in the Run menu. Other features in the menu are not of significant interest for beginning developers and can be slowly explored along the process of creating and building the app.

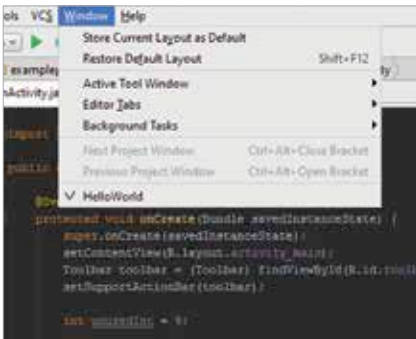
The Run menu is the lifeline of all production quality code. This menu and its corresponding keyboard shortcuts are indispensable during the phase of debugging and optimising the app. To build and run the app, the obvious choice is to click on the ‘Run App’ button, and debugging needs the ‘Debug App’ button to be clicked on. One requirement to use the ‘Run App’ feature



is that there must be a deployment target (an emulator or a connected device), only after which Gradle is used to deploy your app to the target. Other options that appear depressed in the menu (see right) are



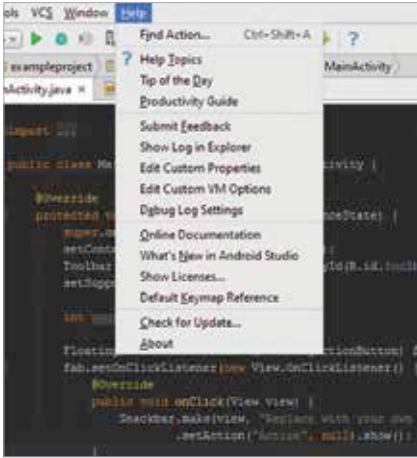
usable only after the process of debugging has been started. Again, the alternatives to the above two features are the 'Run...' and 'Debug...' options, are useful when one wants to significantly reduce the time between updates to their app.



Though the Tools menu seems irrelevant and unnecessary, it is a lifesaver for any developer who has to juggle between multiple window contexts and tasks and would like to seamlessly shift between different window orientations and views. Clicking on the 'Tasks and Contexts' button opens up a list of recently saved window contexts. To create a new context, use the keyboard shortcut **Alt+Shift+T**

and create a new context when the option pops up. Choosing the 'Save File as Template' feature is a good way to create custom boilerplate templates for when you want to jump in directly into a project, without having to write dummy code and outlines from scratch. And finally, the 'IDE Scripting Console' and 'Groovy Console' are used specify commands to the IDE through a console, and a good way to run scripts that automate build and run tasks, and other such workflow tools.

The VCS menu, or Version Control System Menu, is like a local Git for Android Studio. This option allows any code changes and modifications to be worked into your VC workflow, and is useful during large scale collaboration and code publishing. The 'Local History' option allows you to view the last changes made in a document, along with the modification location. Clicking on the 'Enable Version Control Integration' helps start a local VCS such as Git, Mercurial, Google Cloud and Subversion. One thing to remember is, right after enabling a Version Control System, it would make to checkout from



the version control system. Choose the 'Checkout from Version Control' button that appears only after you enable integration, click on the add icon, and type the repository URL. If you are interested in applying a patch to your source code without having to modify it in several places, the obvious choice would be to use the 'Apply Patch' option.

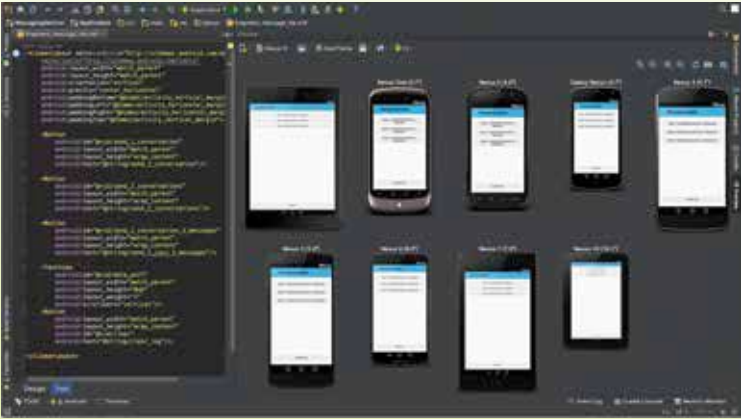
Many of the options under the Window tab are visible on the toolbar or through more

accessible keyboard shortcuts. Nevertheless, if one is lost and is unable to find a way to reopen required Tool windows, going to Active Tool Window | Jump to last Active Tool Window can help restore the last closed set of windows. Like I said, these are generally used through keyboard shortcuts, but having a separate menu for these can make life easier in some situations.

And last but not the least, the standard Help Menu is always present in case you lose your way and need to either reference the Android Studio documentation ('Online Documentation') or would like to view Android Studio logs, in case the software is acting up and you need to find the root cause of the problem. A shortcut for viewing the IntelliJ IDEA help menu is accessible through the toolbar, by clicking on the blue '?' button, which redirects you to the online version of the IntelliJ IDEA Help web page.

One important point to note is that, since Android Studio has been developed on top of the open source implementation of IntelliJ IDEA, many features available in Android Studio are the same. So if you are a seasoned IntelliJ user, shifting to Android studio will take no time at all. For that matter, many commands in Android Studio are very similar to the Eclipse IDE, and so shifting to Android Studio should be a matter of time.

Android Studio is filled to the brim with features that make the workflow of an Android developer much simpler and streamlined, hence, reducing the time wasted on unnecessary activities and increasing productivity. With the release of Android Studio 2.1, it has never been a better time to dive into the software. **d**



EXPLORING THE USER INTERFACE

Now that you've figured out what the different elements of the Android Studio UI, here's how you can go about configuring them to your liking.

Since the first chapter already demonstrated how to install Android Studio, and the second chapter introduced the user interface in the software, it makes sense to put our knowledge to work by developing an application and fiddling around with settings.

The first point of order would be to configure the development environment. After creating a basic boilerplate template, you can either emulate your app on an Android Virtual Device (AVD), or on an Android Phone or Tablet connected to the computer/laptop. For those who would like to try out an AVD, you can start by clicking on the Android Virtual Device

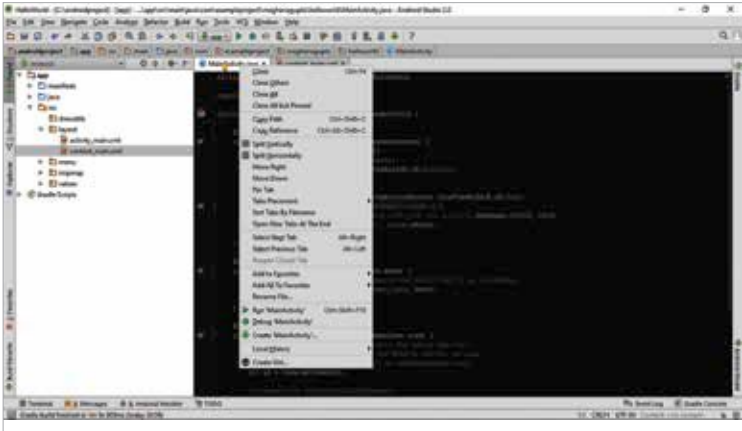


Manager icon (as encircled). After being presented with the ‘Create Virtual Device’ button, click on it and proceed to choose the specifications on the device you wish to emulate the app on. Because emulation means that the physical features of the target device are reproduced, down the RAM and operating system, it is slightly system intensive and may generate a few seconds delay during deployment.

In any case, after choosing the device specifications and then the system image on the panel that follows, you’ve successfully created a virtual device to emulate your app on. To deploy the app, click on the ‘Run’ button.

A reminder – the AVD option for an x86 architecture works only when hyper-virtualisation is enabled on the machine. So in the case you do not have an option to enable hyper-virtualisation, using an AVD is out of the question. A viable option is using a physical Android device to test the app. If your computer does not recognize your Android device when you connect it to your computer via a USB cable, you probably require a USB driver. If your computer initially recognizes your Android device, you should probably forgo installing a different USB driver. For the device to deploy the app, you’ll have to enable the USB Debugging option. After that, click on the ‘Run’ button to deploy the app on the physical device. Again, there’s a catch. If you are developing on Windows and would like to connect a device to test your applications, then you need to install the appropriate USB driver. Fire up a search engine and look for ‘device drivers android studio’ to get the links that provide a pretty decent coverage of web sites for several original equipment manufacturers (OEMs), where you can download the appropriate USB driver for your device.

This is pretty much all you need to set up your development environment and start writing code, deploying the app and debugging it. In the case you feel more comfortable with another visual layout and would like to change the placement of the various window panels, go to View | Tool Windows and select the windows you would like to view. For any tool windows you would like to remove from your view, simply right-click on the window tab and choose ‘Remove from Sidebar’. And like we mentioned in the previous chapter, in case you need to interrupt your work and switch to a new task, go to Tools | Tasks and Contexts and save your current window layout, so you can switch back to it after your work is done. If you would like to enable additional options such as line numbers and themes, go to File > Settings >



Appearance and then select whichever options you need. And if you're still unable to make changes in your window layout, just explore the options in the View menu or just go bonkers with Google and Stackoverflow.

Many users of Android Studio work with the software from behind a firewall or secure network. For them, connecting through a proxy is the only way to connect Studio to the internet while working. Proxies serve as intermediary connection points between HTTP clients and web servers that add security and privacy to internet connections. To support running Android Studio behind a firewall, set the proxy settings for the Android Studio IDE and the SDK Manager. Use the Android Studio IDE HTTP Proxy settings page to set the HTTP proxy settings for Android Studio. The SDK Manager has a separate HTTP Proxy settings page.

To set a proxy for Studio, follow the given steps:

1. From the Menu bar, choose File > Settings > Appearance & Behavior -- System Settings -- HTTP Proxy.
2. Select 'Auto-Detect Proxy Settings' to use an auto-configuration URL to configure the proxy settings or manual to enter each of the settings. Click Apply to enable the proxy settings.

To set the proxy for the SDK Manager:

1. Start the SDK Manager and open the SDK Manager page.
2. On Windows, select Tools > Options from the menu bar.
On Mac and Linux, choose Tools > Options from the system menu bar.

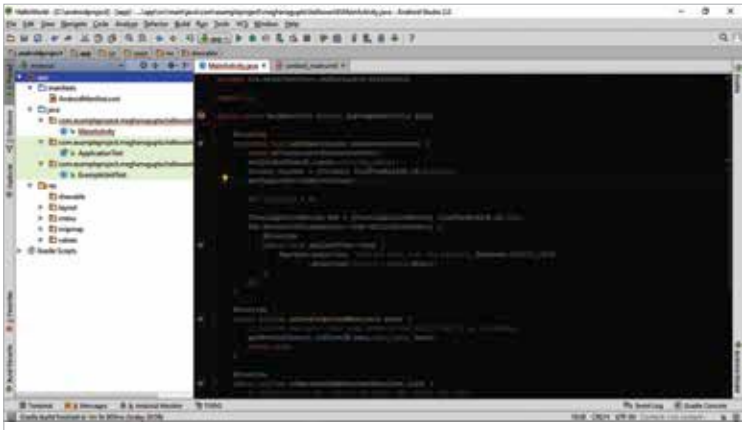
The Android SDK Manager page appears. Enter the settings and click Apply.

Most of what has been talked about up till this point is well and good for developing an app and probably even deploying it. But if you rely heavily on third party vendors, external libraries, SDK tools and other components, getting along with the SDK Manager and learning to play with it will make a world of difference. The first step would be to update old libraries and install missing components. Go to Tools > Android > SDK Manager or click on SDK Manager in the toolbar. You'll be presented with something like this:

If you have been unable to run or deploy your app because of missing packages, check if the following are installed by launching the standalone SDK Manager ('Launch Standalone SDK Manager'):

1. Android SDK Build Tools (Includes tools to build Android apps)
2. Android SDK Platform-tools (Includes various tools required by the Android platform, including the adb tool)
3. Android SDK Tools (Includes essential tools such as the Android Emulator and ProGuard)
4. Android SDK Platform (In the SDK Platforms tab, you must also install at least one version of the Android platform. Each version provides several different packages. To download only those that are required, click the check box next to the version name).

Since these are the bare minimum requirements needed to deploy your app, it would be advisable to update these packages. In the case that you need to use libraries and packages from third party websites, click on SDK



Shortcut	Function
F12	Moves the focus from the editor to the last focused tool window
Alt + Insert	Generate getter and setter methods
Select code fragment + Ctrl + Alt + T	Surround code with try... catch statement
Ctrl + /	Uncomment a single line of code
Ctrl + Shift + /	Uncomment a commented block
Ctrl + Shift + V	Insert recent clipboard contents
Alt + Up, Alt + Down	Move Between Methods
Ctrl + Shift + F12	Hide All Tool Windows
Esc	Return to Editor
Alt+1	ProjectTool Window
Ctrl + Alt + 9	Version Control Window
Shift+F10	Run
Shift + F9	Debug
Alt + 6	Android Monitor Window
Ctrl + Space	Code completion for a class
Ctrl + B	Navigate to the declaration of a class, method or variable
Select element + Enter	View members of the current class
Ctrl + N	Go to Class
Ctrl + Shift + N	Go to File
Ctrl + Alt + Shift + N	Go to Symbol
ALT + Left-Arrow; ALT + Right-Arrow	Navigate open tabs
Ctrl + E	Lookup Recent files

Update Sites tab. You can add other sites that host their own tools, then download the packages from those sites.

If a carrier or device manufacturer has hosted an SDK add-on repository file on their website, follow these steps to add their site to the Android SDK Manager:

1. Click the SDK Update Sites tab
2. Click the Add button (plus sign) at the bottom of the window, then enter the name and URL of the third party site.
3. Click OK.
4. Make sure the check box is checked in the Enabled column.
5. Click Apply or OK.

If you need to search for a particular file within the current directory structure, you can collapse and expand directories in the Project Tool Window to the left of the Editor window and search for your file. In case it is a commonly accessed file, you can go to File > Open Recent and search for your file in the list. And in the case you would like to search for a particular method, class or symbol in all the open files, use Ctrl + N, Ctrl + Shift + N and Ctrl + Alt + Shift + N for a class, file and symbol respectively.

As you've noticed, every major function in Studio has a corresponding keyboard shortcut. We can summarise some of them in a table for quick referencing later on. Since the regular copy and paste commands are universal everywhere, we won't be mentioning those in our list.

Even though most of these commands can easily be found in the various menus in the Menu bar, it's advisable to pick up a few keyboard shortcuts to help increase proficiency while coding.

All in all, Android Studio has a pretty impressive list of functionalities to get you comfortable and started with Android App development. With time, it's easier to gain mastery of the IDE and the multitude of features it provides. **d**



WORKING WITH PROJECTS

Having familiarised yourself with the nuts, bolts, buttons and tabs of Android Studio 2, it's time to start some real work - here's a quick primer on creating and modifying projects, and getting started towards making your own Android app.

CREATING A NEW PROJECT

A project is the complete package that goes towards building your app, since it contains all the files that make up your source code.



Time for the ABC of programming - a quick "Hello World" Android app to get you started.

1. To create a new project, go to File -> New Project, or go to New Project on the Welcome screen, depending on whether or not you have a project open already.
2. Your app starts taking shape here on out. Fill the following details:
 - **Application Name:** Your app's identity on the outside and what the users will know it by.
 - **Company domain:** An attribute that's added to the package name and will be saved for reference in future projects.
 - **Package name:** The project's fully-qualified name; has to necessarily be unique from all other packages installed on the system.
 - **Project location:** The directory where the project is saved.
3. On the next screen, you're asked to select which form factors your app runs on. The default option is of course phones and tablets, but you can include various Android devices: Wear, TV, Auto, and Glass.
 - An important choice here is the minimum SDK - this is the oldest version of Android that your app can run on. Android Studio has a Help

me choose option too, which pulls up a neat cumulative distribution of Android users by version number.

4. Next, you can add Activities to your app. An activity is a response to an action by the user - opening the app, swiping in from the side, touching an element on screen, etc.
5. Your project is now up and running. In the tab on the left (or by pressing Alt+1), you can view your default project files.

Your project files are what make your app, and by default the project files will be shown in Android view, which is a simplified representation of the most commonly used files. All build-related files are grouped under Gradle Scripts, and all other application files come under the app folder. Inside that, resources such as images and layouts are grouped in the res folder regardless of their actual location on the disk.

The Project view is representative of the actual hierarchy on the disk and also includes hidden files. You will see that it is far more detailed than the Android view. Some of the most important subdirectories in this view are:

/app

- build: where build outputs are stored
- libs: where private libraries are stored
- src: where the code as well as the resources (in the res subdirectory) are stored

Configuring your (File ->) Project Structure (or Ctrl+Alt+Shift+S) settings lets you add Google services to your app, such as Google sign-in for a simpler authentication process, Google Analytics to bring you user interaction metrics for your app across categories to understand your target audience, and AdMob to monetise your app through ads. In addition, you can modify the SDK, ADK and NDK locations, and set the versions for Gradle and the Android plugin for it. In addition, the Module section lets you configure various options for your app including Properties, which specifies the SDK version and its build tools; Signing, which specifies your APK signature, and Flavors, where you can give each build its own characteristics, like giving different set of minimum and target Android versions for each flavor.

JUMPING SHIP

Eclipse is a fairly popular choice of an IDE for Java programming, and the Android Development Tools plugin for it extends its capabilities, allowing



Migrating from Eclipse to Android Studio? We've got you covered.

you to create Android projects. However, if you feel the need to migrate from Eclipse to Android Studio 2, we've got you covered.

You'll need to ensure the following before migrating:

1. For Eclipse ADT:

- The AndroidManifest.xml file must be inside the Eclipse ADT root directory. The root directory must also contain either the .project and .classpath files from Eclipse or the res/ and src/ directories.
- Ensure your latest workspace and project updates are saved and included in the import. You can do that by making a build of your project before importing.
- In the project.properties or .classpath files, you will need to comment out the references to Eclipse ADT workspace library files for import. These references can be added in the build.gradle file after the import.
- Make a note of your workspace directory, path variables, and path maps that could be used to specify unresolved relative paths, path variables, and linked resource references. Android Studio allows you to manually specify any unresolved paths during the import process.

2. For Android Studio:

- Find out which third-party plugins are being used for your Eclipse IDE currently and see if there are equivalent Android Studio compatible

plugins that are available through the IntelliJ Android Studio Plugins repository in File -> Settings -> Plugins

- If your Android Studio will be running behind a firewall, set the proxy settings for Android Studio and the SDK Manager. Android Studio requires an internet connection during setup, third party library access, access to remote repositories, Gradle initialization and synchronization, and Android Studio version updates.
 - In File -> Settings -> System Settings, ensure that you have the latest version of Android Studio.
3. When you import your Eclipse project to Android Studio, it will create a new Android project in the latter where the Eclipse workspace becomes a new Android project and each Eclipse project becomes a module inside it. So the files you select for importing will accordingly depend upon the structure of your Eclipse ADT development project.
 - If a workspace has multiple projects, each project folder for each Eclipse ADT should be chosen separately.
 - If there are separate test projects, the test project folder should be selected for import. Android Studio will look for the source project using the dependency chain automatically.
 - For Eclipse ADT projects sharing dependencies in a single workspace, each project should be imported into Android Studio individually. The dependencies will then be maintained automatically by Android Studio.
 4. Now for the actual import process. Start by going to File -> New -> Import Project.
 5. Select the Eclipse ADT folder that contains the AndroidManifest.xml file.
 6. Select the destination folder.
 7. Select the import options, and click Finish.

GOOD ARTISTS COPY...

Picasso said it, so it must be true. When you created a new project, you would've noticed that Android Studio serves up a helpful range of templates. Here's how to add one to a current project:

1. Go to File -> New -> Module.
 2. Enter the Application Name, Company Domain, Package name, and minimum SDK, then click Next.
 3. Pick a template from the options and click Next.
- A quick look at some of the templates on offer:

1. Blank Activity template: you can use this template with three different navigation types if you need to manage multiple pages from this activity.
 - None: A basic, minimal starting point for the app. The action bar contains only a settings menu button.
 - Tabs or Tabs+Swipe: The action bar contains pages or sections that can be navigated using swipe gestures.
 - Swipe Views + Title Strip: The action bar shows a scrollable list of pages/sections that can be navigated with swipes.
 - Dropdown: Adds a dropdown menu to the action bar which lists the pages to be navigated.
2. Full Screen Activity template: The activity runs full-screen by default and switches to a view with the notification bar and the main soft keys when the screen is tapped. The full-screen behaviour is realised using a SystemUiHider implementation.
3. Master Detail Flow template: In this template, the layout that the app uses is adaptive to the device using the app. A simple example of this is the Gmail app which behaves differently on tablets and phones. Selecting from the “master list” leads you to a page having details about the list item. On a tablet, the master list (say, your list of emails) and the details (the contents of your selected email) would be displayed simultaneously, while smaller devices would either reflow it differently or have the two display on separate pages entirely.

There are even specific activities that you may want to add to your app, and Android Studio provides templates for some frequently-used ones as well. To add an activity template, go to the project folder of the app where you want the activity, and right click it. Select New -> Other... and select Android -> Android Activity, then Next. Two of the handiest ones are login and settings..

4. Login: creates a Google-recommended interface for asking for login information. An implementation called AsyncTask handles network operations independently from the main interface and also provides a progress bar for network operations.
5. Settings: leverages the PreferenceActivity class and provides sample implementations of different data types for settings, such as binary (yes/no type), text values, time-dependent behaviours, etc.

...GREAT ARTISTS STEAL

It's fairly obvious that making even a bare-bones, minimal functional app is an extremely time- and labour-intensive process where your attention



You don't always have to make your own app from scratch. Learn from apps others have written and shared, and you'll grow faster.

is divided between aesthetics, functionality and marrying the two so that the end product isn't a total pain in the back to use.

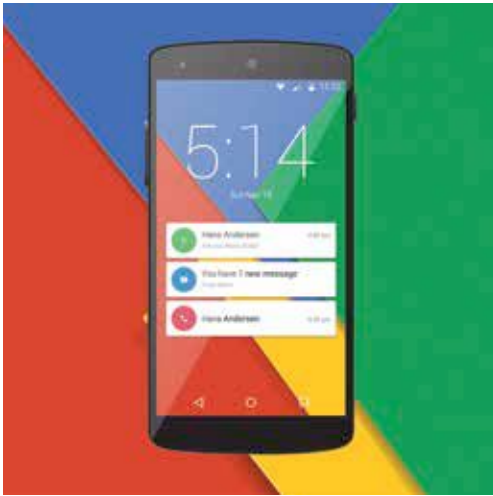
If you'd like some inspiration from others who have built entire apps, there are several templates of apps that you can use to learn how to make a set of relevant functionalities work together. Envato, an online digital marketplace that sells digital media such as videos and code, and even lets you hire designers and developers, offers app templates for you to learn from and base your own app on.

1. **Restaurant Finder:** A full android app compatible with both Android Studio and Eclipse, it features map search, reviews, social media integration, rating and booking restaurants, and user profiles, just to name a few. It also includes an administrator panel in PHP that allows adding and removing restaurants and users and a very informative dashboard with various metrics like number of reviews by each user.
2. **Your Radio App:** A mobile internet radio streaming app with features similar to the ones above. Additionally, material design templates, push notifications and Google Analytics are also offered. A PHP MySQL backend included with the app helps interact with the server side.
3. **Inventory Management App:** Lets you track inventory, offers a choice of list view or report view, and also lets you send push notifications

for low stocks. Product keys available are: name, detail, ID, category, company, price, quantity, location and photo. The data and/or reports can be exported as an Excel file. The app uses Material Design and has AdMob integration as well.

MATERIAL DESIGN: THE (ANDROID) WORLD IS FLAT

With the release of Android 5.0 Lollipop, Google introduced a completely new design paradigm called Material Design. It draws heavily from the principles of flat design and minimalism, and a look at their blog explaining Material Design will show you just how immensely detailed the homework for something that looks so simple and clean has had to be. This philosophy uses real-world material, paper in particular, as the metaphor for all design elements and their interactions on-screen. The guidelines for this fluid design paradigm are fairly simple and in many instances far



Android's Material Design paradigm completely changed the entire Android experience. Leverage it to make your app smoother to use and more attractive.

more intuitive and exciting in their response compared to skeuomorphic design elements of the past.

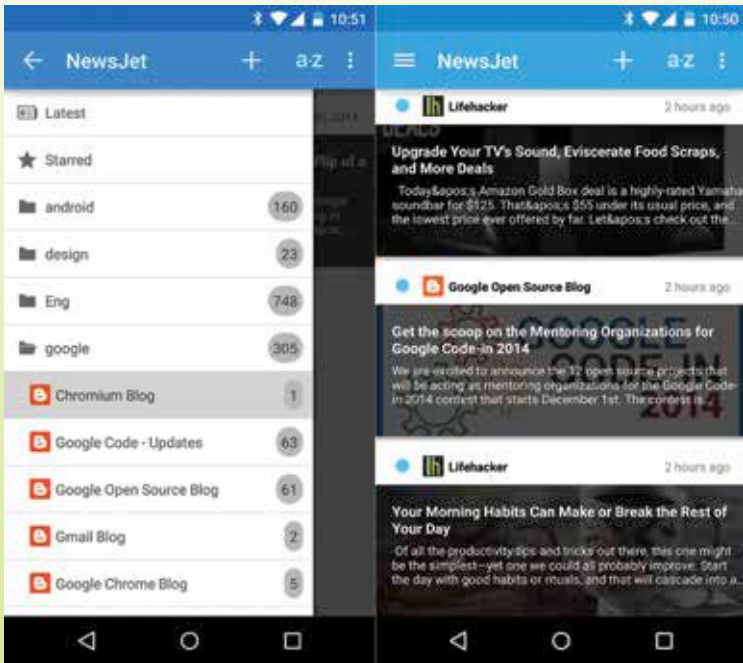
Fortunately, the steps to including a Material Design template in Android Studio for your app are equally simple. For example, if you want to use a navigation drawer accessible with a swipe in from the left, and want to

make it Material, you can search for Android Studio Material template and download the latest version.

1. Next, extract the folder, and copy the folder in the root, called `MaterialNavigationDrawerActivity` and paste it into the `plugins/android/lib/templates/activities` folder inside the Android Studio location on your disk.
2. Restart Android Studio if needed, and open a project.
3. The library will only work for a minimum API of level 7, so choose your minimum SDK appropriately.
4. The Material Drawer will be available as a template activity. Choose it and configure the parameters as needed. You can set the drawer to appear above or below the toolbar, and the navigation pane can be made just like the Google Material Navigation bar or a basic list.
5. Click Finish, and after building the project, you will be able to run it and see the new activity in action.

This was just one example of how adding an element of Material Design to your app works. Keep in mind that most of these templates will have a higher minimum API level, so make sure you're not accidentally skipping on some core utilities of your app just for a little extra fluidity in the UI. **d**

CHAPTER #05



DEVELOPING YOUR OWN APP

Hopefully, you've got the hang of Android Studio by now. So let's make your very first app using the software.

Android Studio is designed particularly for Android development to accelerate the android application development process and make it simpler. In this chapter, we will be harnessing the power of Android Studio to build an Android Application from scratch, in real time. We will be building a RSS Feed Reader android application, which will be pulling RSS Feeds from a pre-defined source in our code and displaying it in a List.

Before diving in directly towards building the app, it would be better to understand some common terminologies used in the Android application development process. You can find the apk here - <https://goo.gl/3dYr8D>

1.Android Basics

Activity

In Android, an activity represents just a single screen with a user interface. Activity is basically a Java code that supports UI. Activity class is an inbuilt class in Android and every application which supports UI must inherit it to create windows. A more-than-basic app would have more than one activity and these activities, though independent of each other, work together to form a cohesive user experience.

Intent

Intent is a type of asynchronous message that allows application components to request resources from other Android components. The basic principle of Intents is to allow you to interact with components contributed by other applications as well as components from the same applications. For example, you can fire up the system browser through your app using intents.

Intents are of two types - Implicit and Explicit. In implicit intent, the component is not specified and the android system by itself decides the best suitable component for the intent call whereas in explicit intent the component is defined.

Gradle

In Android Studio, the building of Android projects is handled by the Gradle build system. When you create a new project in Android studio, the Gradle build scripts are created automatically. The Gradle runtime is pre-configured with Android Studio, so no additional installation is required.

The traditional Eclipse build system has certain limitations to support some complex scenarios. The Gradle build system handles complex scenarios like Multi-distribution and Multi-APK while creating Android applications.

Once you got familiar with the common Android terminologies, you will learn to configure Android Studio to build your new Android app.

2. Setting Up Android Studio for your new project

We assume that, by now, you have Android Studio installed and the Android SDK already configured with the Android Studio. Now, it is time to get your hands dirty with some serious coding and building your first ever Android Application.

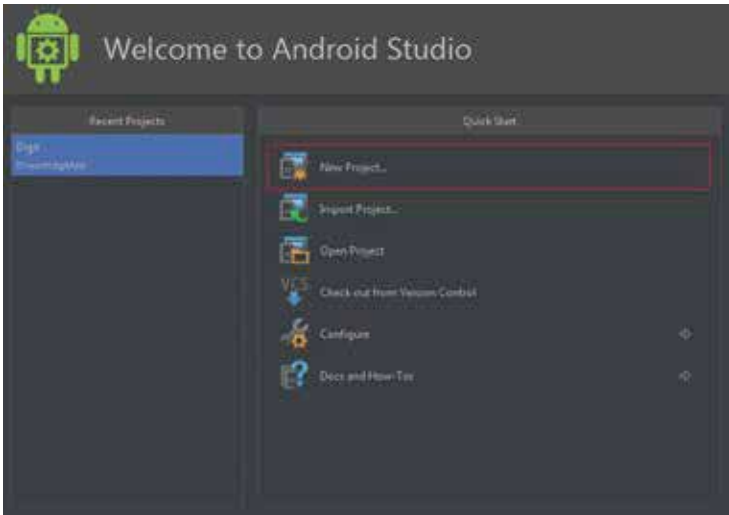
STEP 1 : Fire up Android Studio. You will be presented with an awesome loading screen with all the phones and tablets in the background.

STEP 2 : Once Android Studio is done loading, you shall be greeted with a Welcome Screen with a whole lot of options like creating a new project



Welcome to Android Studio!

or importing / opening an existing project. Click on New Project from the
STEP 3 : Android Studio shall present you a screen as shown in the following image. You will have to specify the Project Name and other details

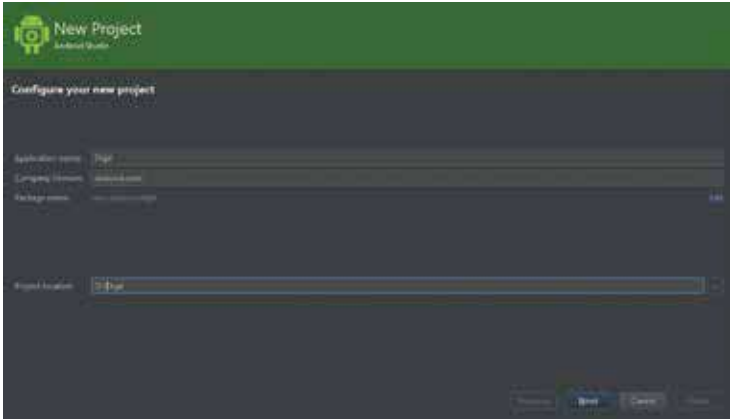


for Android Studio to create the new Android project. You can feel free to use different values for all the fields, as per your requirement.

- Application Name is the name of the app that appears to its users. For this project, we are specifying the app name as “Digit”.
- Company domain is a qualifier that is appended to the package name along with the Application Name.
- Package name is the fully qualified unique name for the project and based on the similar naming convention as the Java programming language). The package name is ought to be unique across all the other packages on the Android system.
- Project location is the pointer to the directory that holds the source code and Android Studio files.

Fill up the required details and click Next.

STEP 4 : On the next screen you will have to define the form factors on which your app will run on. Click on the checkbox for Phone and Tablet. Remember to set the Minimum SDK . This sets the minimum Android version that should be running on the user device to run your app. The minimum SDK value basically defines the SDK capabilities you want and the devices you want to support. We recommend using API version 14(Android



4.0 Ice Cream Sandwich). Generally, each app has a different requirement and you may need to select a different version, depending on the situation.

Select the required details and click Next.

STEP 5 : On the Add an activity to Mobile screen, select Blank Activity and click Next.

STEP 6 : On this screen, you will have to define values like Activity Name, Layout Name and Title. By default Android Studio sets these values to MyActivity, activity_my and MyActivity respectively.



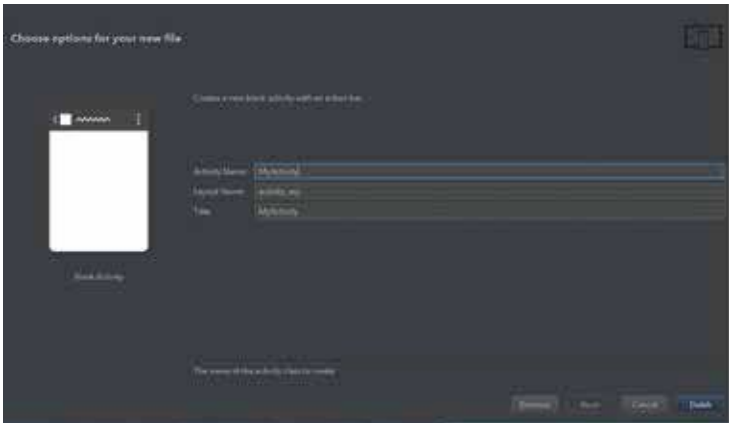
- **Activity Name :** Activity Name is the name of the class that will contain our code. Precisely, we will have a Java Class that is an activity called whatever we name it here.
- **Layout Name :** Android UI Layouts are generally defined in a separate XML text file, not with our Java Code. The Layout Name is what this file will be called.
- **Title :** This is different from Activity Name field and will be used by Android on the device's screen as the name of the App.

As a first android project, we can go ahead with the default values and click Finish.

Android Studio will start building your project.

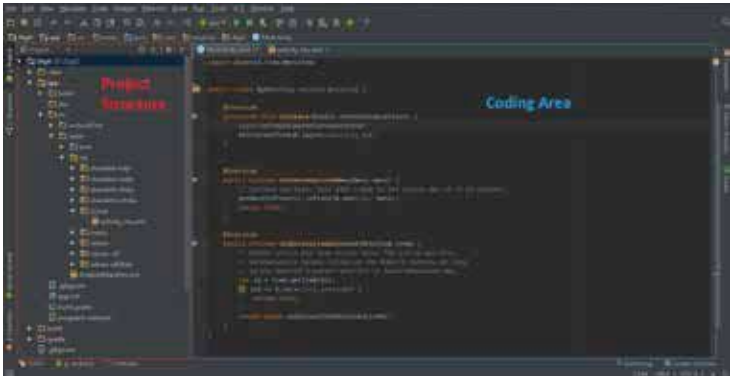
3. Building Your First Android App

Android Studio would take this a few moments to do a few background operations and create the project. After Android Studio is done building



your project, you could see a workspace screen as shown below. You would notice that the project is almost empty, but it has pretty much everything already set up to launch it on an Android device or emulator.

We shall now begin with building our RSS Reader Android Application. As per the image below, you can see that we have divided the screen into two parts – Project Structure and Coding Area. The project structure basically



displays the complete file structure of the App and the coding area is where the magic happens. In the coding area, you may see that there are two files already open for us – **MyActivity.java** and **activity_my.xml**. These are the names which we have set in Step 6 while setting up Android Studio for our project. Android Studio, by default, provides us with a boilerplate by adding some mandatory code into these files, to ease the process of development.

The default boilerplate provided by Android Studio looks like below:

MyActivity.java

```
package com.arsaviva.digit;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.view.Menu;
```

```
import android.view.MenuItem;
```

```
public class MyActivity extends Activity {
```

```
    @Override
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_my);
}
```

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.my, menu);
    return true;
}
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();
    if (id == R.id.action_settings) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}
}
```

activity_my.xml

```
<RelativeLayout xmlns:android="http://schemas.
android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MyActivity">
```

```
<TextView
```



```

android:text="@string/hello_world"
android:layout_width="wrap_content"
android:layout_height="wrap_content" />

```

```
</RelativeLayout>
```

Before getting directly into building your RSS Feed Reader Android Application, it is best to understand a few elements which we will be using in our App.

ListActivity

ListActivity is a subset of the **Activity** class that displays items in the form of a list, by binding to a data source like an **Cursor** or **Array**. **ListActivity** holds the **ListView** object that can be bound to different data sources, and exposes event handlers when the user selects an item. In Android applications, **ListViews** are used to create views with lists of data. Lists are populated with data from data sources with simple arrays or with cursors.

In our RSS Reader App, we will be using **ListActivity**, instead of using an **Activity** to extend our main class from. The text to be populated in the list is available in the string array **listItems**. Similar to other activities the **onCreate()** method is also used and the first thing that needs to be done is call **super.onCreate**. Next set the **activity_my** file as the layout to display with **setContentView(R.layout.activity_my)**.

After all of this is done, the last thing you need to do is to populate the list. Now, in order to populate the list you need to create an **ArrayAdapter** and hand it over to **setListAdapter()**. The **ArrayAdapters** manages the **ListView** objects in an internal array. The **ArrayAdapter** constructor has to be passed 3 parameters, - the current class instance (**this**), the display layout to use for each list item, and the array of items respectively.

AsyncTask

The Android framework is strictly against performing long background tasks like network operations on the main UI thread and recommends performing such operations on a separate thread. To meet with this requirement, Android provides us with an abstract class - **AsyncTask**. **AsyncTask** lets us use the UI thread properly. It class allows us to perform background /long operations and show its result on the UI thread without manipulating threads and/or handlers.

Android is based on a single thread model and whenever a new android application is launched, a new thread is created. In our RSS Feed Reader

App, we will perform some network operations to pull the feeds from the Internet. As soon as the main activity loads a request would be made to the server and response will be awaited. Now, because Android is based on a single thread model, our screen would basically freeze till the time response is awaited. Hence, performing long running operations on the UI thread must be avoided. This also takes into account file and network access.

To get over such situation, you will have to create a new thread and implement run method to perform such network operations, so that the UI stays responsive. But the Android UI toolkit is not thread safe, so if we ever need to make some change to the UI on the basis of the result of the background operation performed, then this approach will lead to some serious issues. To resolve these issues very good pattern which is enveloped into **AsyncTask**. An **AsyncTask** is defined by a computational code that runs on some background thread and whose result can be easily published on the UI thread.

AsyncTask generally has four basic steps:

1. doInBackground: The main part of the code which is to perform long running operation is declared in this method. The **onClick** method when executed, calls **execute** method which accepts parameters and calls the **doInBackground** method automatically, with the parameters passed.

2. onPostExecute: After the **doInBackground** method has completed processing, this method is being called. The result from the **doInBackground** is passed to the **onPostExecute** method.

3. onPreExecute: This method is called must before the **doInBackground** method is invoked.

4. onProgressUpdate: This method can be invoked by calling **publishProgress**, whenever required, from **doInBackground** call this method.

XmlPullParser

XML stands for Extensible Markup Language and is a reputed data exchange format. An XML document typically consists of elements - every element has a start tag, content and an end tag. An XML document will have exactly one root element and it is case-sensitive. An XML file is considered valid if

that is well-formed and should contain a link to an XML schema and which later validates it. In the Android Framework, it is recommended to use the `XmlPullParser` for parsing and writing XML. This `XmlPullParser` is not available in the Java Framework but is very much similar to the `Stax` parser.

Its high time that we now dive into building our project. You shall first start with modifying your Android Manifest file. In order to retrieve RSS Feeds from over the Internet, you need to declare the Internet permission in your Manifest file as follows –

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

Now, your manifest file would look as shown below. You would notice that you will have to declare each activity of your app in the manifest file. `MyActivity` is by default declared in the manifest file and we will have to do it manually for the rest of the activities.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.arsaviva.digit"
    >
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MyActivity"
            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>

    <!-- Internet Permissions -->
    <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

Next we will move to working on the layout file. Fire up the `activity_my.xml` file. First of all, we shall change the layout type. We shall replace the default `RelativeLayout` to `LinearLayout` because in this app we will be organizing our entire view into a single column in the form of List. Inside the `LinearLayout` we will declare a `ListView` element and provide it with the id `@android:id/list`. Once we are done with this, we are ready to begin with coding the main logic of the app in the Java file.

activity_my.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.
android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <ListView
        android:id="@android:id/list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" >
    </ListView>

</LinearLayout>
```

In the `MyActivity.java` file, you shall notice that Android Studio by default imports the required packages and defines the package name and the mandatory methods. Apart from the default packages, you will also need to import some additional packages which you will require to develop the app. Append the following lines of code to your activity to import the additional packages –

```
import android.app.ListActivity;
import android.content.Intent;
import android.net.Uri;
import android.os.AsyncTask;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
```

```
import android.widget.ListView;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import org.xmlpull.v1.XmlPullParserFactory;
import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
```

Once you have added the above piece of code to your activity, have a close look at **MyActivity** class declaration. You shall notice that by default the activity extends the **Activity** class. Because, you will be using **Listview** in this activity, you will have to make sure that the activity extends the **ListActivity** class instead of the **Activity** class.

```
public class MyActivity extends ListActivity
```

Now, you define two global variables of list datatype – headlines and links. The headlines variable will hold all the headlines of the RSS Feeds and the links variable would hold all the links to the respective content.

```
List headlines;
```

```
List links;
```

The next major goal is to code the **onCreate** method. Inside the **onCreate** method, at the very first step you will have to inflate the **activity_my** layout. Because, our app is heavily dependent on long running background operation to fetch RSS feeds over the network, we will stick to **AsyncTask** to do so. We instantiate the **AsyncTask** within the **onCreate** method so that the app starts the **AsyncTask** process as soon as it is loaded.

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_my);
    AsyncCall task = new AsyncCall();
    task.execute();
}
```

For this project, you wouldn't be bringing any modifications in the `onOptionsItemSelected` method and the `onOptionsItemSelected` method. In the next step, you shall instantiate the `AsyncCall` class which would inherit the `AsyncTask` class. As described earlier, the `AsyncTask` would have four methods – `doInBackground`, `onPostExecute`, `onPreExecute` and `onProgressUpdate`. These methods are used for their original purpose. All the heavy lifting network operation are done by the `doInBackground` method and later the data from the network is bind to the UI through the `onPostExecute` method.

```
private class AsyncCall extends AsyncTask<Void, Void, Void> {
    private static final String TAG = "FeedTag" ;
    @Override
    protected Void doInBackground(Void... params) {
        Log.i(TAG, "doInBackground");
        getFeeds();
        return null;
    }
}
```

```
@Override
protected void onPostExecute(Void result) {
    Log.i(TAG, "onPostExecute");
    // Binding data
    ArrayAdapter adapter = new ArrayAdapter(MyActivity,
this,android.R.layout.simple_list_item_1, headlines);
    setListAdapter(adapter);
}
```

```
@Override
protected void onPreExecute()
{
    Log.i(TAG, "onPreExecute");
}
```

```
@Override
protected void onProgressUpdate(Void... values)
{
    Log.i(TAG, "onProgressUpdate");
}
}
```

You would have noticed that inside the `doInBackground` method you have called for the `getFeeds` method. This is the method where you will be pulling all the RSS Feed data from the network. So, it's high time you got into doing some network operations. Firstly, you will instantiate the global variables `headlines` and `links` as `ArrayLists`, which you declared earlier. Next, you will define the URL source from which you are going to pull the RSS Feeds. Here, we are using Digit's RSS Feeds. You may feel free to try with any other source. Now, you shall fetch the XML content from an `InputStream`. We will be defining a method `getInputStream` to do so, later in the chapter.

Once we have the XML content, we shall parse the XML content. We will be looking for the "`<title>`" tag that appears inside the "`<item>`" tag. However, we must also take in consideration that the name of the RSS Feed is also enclosed in a "`<title>`" tag.

Each and Every RSS Feed begins in the following format -

```
<channel>
<title>Feed_Name</title>
...."
```

We will have to skip the "`<title>`" tag which falls under the "`<channel>`" tag, and only consider those "`<title>`" tags which falls under "`<item>`". To achieve our requirement we make use of a Boolean variable `insideItem`. We shall now fetch the starting tags and the ending tags and later based on the status of the previously declared boolean variable, fetch the headlines and the links. These headlines and links are stored in the global variables `headlines` and `links`. This process is continuously repeated till the end of the document is reached.

```
public void getFeeds()
{
    headlines = new ArrayList();
    links = new ArrayList();

    try {
        URL url = new URL("http://feeds.feed-
burner.com/digit/latest-from-digit");
        XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
        factory.setNamespaceAware(false);
        XmlPullParser xpp = factory.newPullParser();
        xpp.setInput(getInputStream(url, "UTF_8");
```

```

        boolean insideItem = false;

        // Returns the type of current event: START_TAG, END_TAG, etc..
        int eventType = xpp.getEventType();
        while (eventType != XmlPullParser.END_DOCUMENT) {
            if (eventType == XmlPullParser.START_TAG) {

                if (xpp.getName().equalsIgnoreCase("item")) {
                    insideItem = true;
                } else if (xpp.getName().equalsIgnoreCase("title")) {
                    if (insideItem)
                        headlines.add(xpp.nextText()); //extract the headline
                } else if (xpp.getName().equalsIgnoreCase("link")) {
                    if (insideItem)
                        links.add(xpp.nextText()); //extract the link of article
                }
            } else if (eventType == XmlPullParser.END_TAG
&& xpp.getName().equalsIgnoreCase("item")){
                insideItem=false;
            }

            eventType = xpp.next(); //move to next element
        }

    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (XmlPullParserException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

```

The next step in the project is to work out with two more simple methods – `getInputStream` and `onListItemClick`. We pass the source URL defined in the `getFeeds` method to the `getInputStream` method and this method returns us with the well-formed XML, which we later parse to display within the app. The XML data being pulled from the RSS Feed source need to be bound

to a view. We display only the headlines which has been parsed through the XML in a `listview`. But, only the headlines is not convincing enough. The users want to read more on the topic, if the headlines is appealing enough. Thus, we define the method `onListItemClick` such that if the user taps on a certain list item / headlines, we pass the respective link of the article, which we have parsed through the XML, as an Intent to the system browser and it opens up article for the user.

```
public InputStream getInputStream(URL url) {
    try {
        return url.openConnection().getInputStream();
    } catch (IOException e) {
        return null;
    }
}

protected void onListItemClick(ListView l, View v, int position, long id) {
    Uri uri = Uri.parse((String) links.get(position));
    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    startActivity(intent);
}
```

Once you are done coding the above, your `MyActivity.java` should read exactly as shown below -

MyActivity.java

```
package com.arsaviva.digit;
import android.app.ListActivity;
import android.content.Intent;
import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
```

```

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import org.xmlpull.v1.XmlPullParserFactory;

import java.io.IOException;
import java.io.InputStream;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;

public class MyActivity extends ListActivity {

    List headlines;
    List links;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);
        AsyncCall task = new AsyncCall();
        task.execute();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.my, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(Menu.Item item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

```

```

    if (id == R.id.action_settings) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}

private class AsyncCall extends AsyncTask<Void, Void, Void> {
    private static final String TAG = "FeedTag";

    @Override
    protected Void doInBackground(Void... params) {
        Log.i(TAG, "doInBackground");
        getFeeds();
        return null;
    }

    @Override
    protected void onPostExecute(Void result) {
        Log.i(TAG, "onPostExecute");
        // Binding data
        ArrayAdapter adapter = new ArrayAdapter(MyActivity.this, android.R.layout.simple_list_item_1, headlines);
        setListAdapter(adapter);
    }

    @Override
    protected void onPreExecute()
    {
        Log.i(TAG, "onPreExecute");
    }

    @Override
    protected void onProgressUpdate(Void... values)
    {
        Log.i(TAG, "onProgressUpdate");
    }
}
}

```

```

public void getFeeds()
{
    headlines = new ArrayList();
    links = new ArrayList();

    try {
        URL url = new URL("http://feeds.feed-
burner.com/digit/latest-from-digit");

        XmlPullParserFactory factory = Xml-
PullParserFactory.newInstance();
        factory.setNamespaceAware(false);
        XmlPullParser xpp = factory.newPullParser();
        xpp.setInput(getInputStream(url), "UTF_8");
        boolean insideItem = false;
        int eventType = xpp.getEventType();
        while (eventType != XmlPullParser.END_DOCUMENT) {
            if (eventType == XmlPullParser.START_TAG) {

                if (xpp.getName().equalsIgnoreCase("item")) {
                    insideItem = true;
                } else if (xpp.getName().equalsIgnoreCase("title")) {
                    if (insideItem)
                        headlines.add(xpp.nextText());
                } else if (xpp.getName().equalsIgnoreCase("link")) {
                    if (insideItem)
                        links.add(xpp.nextText());
                }
            } else if (eventType == XmlPullParser.END_TAG
&& xpp.getName().equalsIgnoreCase("item")){
                insideItem = false;
            }

            eventType = xpp.next();
        }
    } catch (MalformedURLException e) {

```

```

        e.printStackTrace();
    } catch (XmlPullParserException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

}

public InputStream getInputStream(URL url) {
    try {
        return url.openConnection().getInputStream();
    } catch (IOException e) {
        return null;
    }
}

protected void onItemClick(ListView l, View v, int position, long id) {
    Uri uri = Uri.parse(links.get(position));
    Intent intent = new Intent(Intent.ACTION_VIEW, uri);
    startActivity(intent);
}
}
}

```

4. Building the Signed APK

Once you have finished developing an Android application and you have tested it on a wide range of Android devices, you are ready to submit your awesome app to the Google Play App Store. However, before submitting your app, you will have to package the app for release and sign it with a private key. To create a signed APK for your Android app, you will have to follow the below steps –

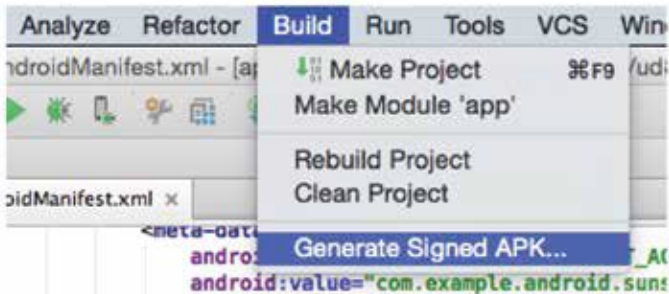
1. Click on **Build** > **Generate Signed APK** in the menu
2. Click on **Create New** to create a new **keystore** file

Creating a keystore

A keystore acts as your signature. In case, you run into some issues with your app, none but only the holder of keystore can update the application. Don't forget to save your keystore in a safe place on your computer or back it up somewhere, if required. You will need to use the keystore to sign your APK, to be able to put your app on the Play Store and later release updates to it. Don't share this file with anyone.

3. Fill in the fields to create your keystore and key as appropriate.
4. Generate your signed APK

In this step, specify the destination folder for you APK and make sure



that the build type is set to release. Click on Finish and you can find the signed APK in the specified destination folder. You can now publish your app on Google Play Store.

5.Android Data Binding Libraries

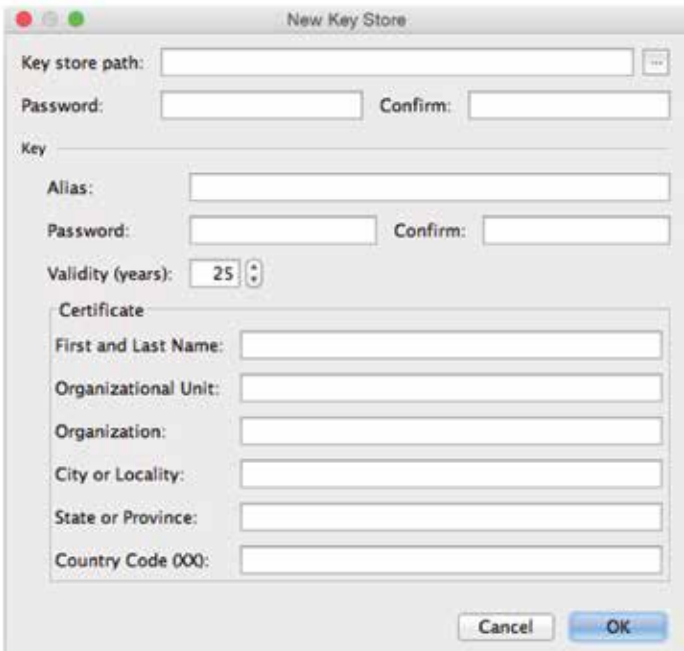
Data binding is used to create a link between the App UI and the data model that holds the information you want to show. The UI widgets like `EditText`, `TextView` etc. are bound to the data stored in java class. Every time there is some change in the data, the UI widget bound to it gets updated automatically. Thus, you don't need to worry about updating the UI by yourself.

You will have to add certain dependencies to you `classpath` in the top level `build.gradle` file, to use Data Binding Library in your Android project dependencies [

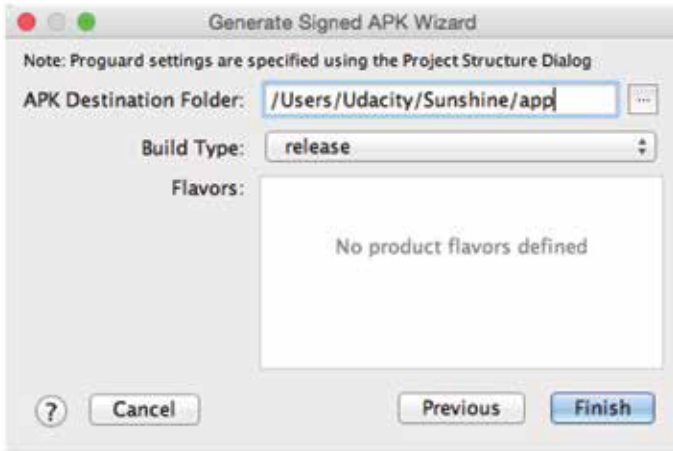
```
classpath "com.android.tools.build:gradle:1.3.0-beta4"
```

```
classpath "com.android.databinding:dataBinder:1.0-rc1"
```

```
]
```



4. Generate your signed APK



The Data Binding plugin must then be applied to the app module build.gradle file after the application plugin.

```
apply plugin: 'com.android.application'
```

```
apply plugin: 'com.android.databinding'
```

Your app will now have all the necessary dependencies to incorporate data binding.

The basic usage model is to use the Data Binding Library to bind strings from the model object to the view. This is generally the most convenient way to use binding in the app. To use data binding in your layout, you will have to wrap your elements in a layout element.

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" tools:context="com.arsaviva.databindingdemo.fragment.SimpleBindingFragment">
    <data>
        <variable
            name="user" type="com.arsaviva.databindingdemo.model.SimpleBindingUser"/>
    </data>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
```



```

        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin">
        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@{user.name}"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:textSize="20sp"
            tools:text="Test Data"/>
    </LinearLayout>
</layout>

```

The model object has to be specified inside the data element. This shall append a setter to the “user” object on the binding class which is generated automatically. The binding classes are generated automatically based on the layout, hence they are named according to the name of the layout file. You can bind the Values to attributes by specifying an expression generic to the Data Binding Library. The expressions are Java statements with some aiding operators to make things simpler.

In order to access the binding within an activity, you will have to inflate the layout using method `setContentView()` of the `DataBindingUtil`. The `setContentView()` method would return a `ViewDataBinding` instance generic to the inflated layout .

```

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    FragmentSimpleBindingBinding binding = DataBindingUtil.
    inflate(inflater, R.layout.fragment_simple_binding, container, false);

    user = new SimpleBindingUser();
    user.setFirstName(getString(R.string.app_name));
    binding.setUser(user);

    return binding.getRoot();
}

```

Here, the `onCreateView`, which is obtained from the `ViewBinding` instance returns the root view of the inflated layout. The `setUser` method

is the only method we have to call on the binding class, and the respective instance would be bound to the UI.

6. Getting started with Google Play Services

Google Play Services is a platform that gives you, the developers, the freedom to integrate the newest and the most powerful features such as Maps, Google+ etc. within your own App with automatic platform updates distributed as an APK through the Google Play store. You can access the interfaces to the individual Google services and also obtain authorisation from users to gain access to these services with their credentials through the Google Play services client library.

Firstly, you need to setup the project with Google Play services SDK to develop an app using the Google Play services APIs. This can be done through the Android SDK Manager which you have configured in the previous chapters.

To test an app when using the Google Play services SDK it is recommendable to use an Android device that runs Android 2.3 or higher and includes Google Play Store.

To use Google Play services APIs available to your app:

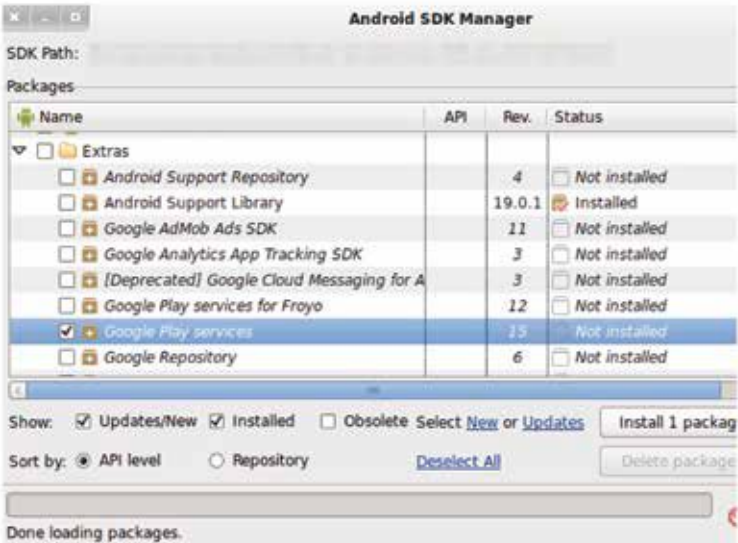
1. Open the **build.gradle** file located inside the application module directory.
2. Add a new build rule under the dependencies section to integrate the latest version of play-services.

```
apply plugin: 'com.android.application'
...
dependencies [
// To compile the entire package of APIs into your app
compile 'com.google.android.gms:play-services:8.4.0'
// To compile the selective package of APIs into your app
compile 'com.google.android.gms:play-services-plus:8.4.0'
compile 'com.google.android.gms:play-services-maps:8.4.0'
]
```

You will have to update this version number each time Google Play services is being updated.

3. Save the build.gradle and Sync Project with Gradle Files in the toolbar.

The next step would be to access the Google APIs through your application. To access the different Google APIs included in the Google Play services



library (Google Sign-In, Maps, or Drive), you will have to create an instance of Google API Client. The Google API Client provides a common interface to access all the Google Play services. Moreover, it also manages the network connection between the user's device and all the Google services. Google Api Client instance is created in your activity's onCreate() method. The GoogleApiClient.Builder is responsible for providing you the methods to allow you to specify the Google APIs to be used. Below is an example of a GoogleApiClient instance that connects with the Google+ service:

```
mGoogleApiClient = new GoogleApiClient.Builder(mContext)
    .addConnectionCallbacks(this)
    .addOnConnectionFailedListener(this)
    .addApi(Plus.API)
    .addScope(Plus.SCOPE_PLUS_LOGIN)
    .build();
```

To learn more about accessing the Google API Services, visit the link below:

https://developers.google.com/android/guides/api-client#start_an_automatically_managed_connection

7. Adding Fingerprint Authentication for your apps

To develop an Android Application for Fingerprint authentication, you need to have a device containing a touch sensor, which is configured to secure the device and enrol at least one fingerprint.

To use the, Fingerprint authentication in your app, you will have to request the `USE_FINGERPRINT` permission within the project manifest file. Within the Android Studio Project tool fire up the **AndroidManifest.xml** file to add the fingerprint request permission as shown below:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.arsaviva.fingerprint">
```

```
    <uses-permission
```

```
        android:name="android.permission.USE_FINGERPRINT" />
```

Fingerprint authentication depends on two system services - Keyguard-Manager and the FingerprintManager. Inside the onCreate method in your Activity, you will have to define the references to these two services as follows:

```
package com.arsaviva.fingerprint;
```

```
import android.support.v7.app.AppCompatActivity;
```

```
import android.os.Bundle;
```

```
import android.app.KeyguardManager;
```

```
import android.hardware.fingerprint.FingerprintManager;
```

```
public class FingerprintActivity extends AppCompatActivity {
```

```
    private FingerprintManager fingerprintManager;
```

```
    private KeyguardManager keyguardManager;
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_fingerprint_demo);
```

```
        keyguardManager =
```

```
            (KeyguardManager) getSystemService(KEYGUARD_SERVICE);
```

```

    fingerprintManager =
        (FingerprintManager)
getSystemService(FINGERPRINT_SERVICE);
}
}
}

```

8. Developing VR Apps for Google Cardboard

At I/O 2015, Google recently presented to the world with its newest version of the Cardboard viewer, which is supposed to work with all high-end, modern smart phones. Cardboard Apps displays stereo 3D rendered scenes on your phone. Cardboard lets the user develop and view experiences which are very realistic, getting completely immersed in an interactive world of virtual reality (VR). Functionality wise, Cardboard is very much similar in function to VR systems, like Oculus Gear VR.

To develop VR Android Apps for Cardboard, you will have to download the Cardboard Android SDK client libraries. You will have to clone the **cardboard-java** repository from GitHub through the following command:

```
git clone https://github.com/googlesamples/cardboard-java.git
```

You can find the SDK libraries in the libraries directory of the downloaded repository as .aar files. Google also provides a few sample apps along with the SDK. These samples can be found in the samples directory as gradle projects already configured for Android Studio.

To develop your own applications with the Cardboard SDK for Android, you will have to first configure your project to adapt to VR Apps, in one of the two ways, as shown below –

8.1 Using Android Studio

1. All the required libraries can be found in the libraries directory of the repository you just downloaded for GitHub. In order to figure out the different .AARs, you would require, feel free to analyze the build.gradle files of the various sample apps in the samples directory of the downloaded repository. Example :

```

dependencies {
    compile project(':libraries-common')
    compile project(':libraries-audio')
    compile project(':libraries-core')
}

```

The above code indicates that this App would require common, audio and core libraries.

- Now you will have to create a new separate module for each of the dependency library. Click on **File** -> **New** -> **New Module**. Select **Import .JAR/.AAR Package**. Find each of the .AARs, one at a time, and import them.
- The next step is to add each of these newly imported modules as a dependency to the main app. This can be done by clicking **File** -> **Project Structure** -> **Modules**. Now, you would notice your App's module name on the left side's section list and certain dependencies on the right side's tab list. Click on **Modules** -> **YOUR APP's MODULE NAME** -> **Dependencies** -> '+' -> **Module Dependency**.

After you repeat these steps for each of the required libraries, you are ready to build your own Cardboard App for Android.

8.2 Directly using Gradle

In the previous method, you used Android Studio to generate new modules and generate the .gradle files. You can also, alternatively, include the libraries in your application's module directly by editing the module's **build.gradle** file and adding some piece of code. Example -

```
dependencies [
    compile(name:'common', ext:'aar')
    compile(name:'audio', ext:'aar')
    compile(name:'core', ext:'aar')
]

repositories[
    flatDir[
        dirs 'libs'
    ]
]
```

The above piece of code basically instructs the Gradle to look in the libs subdirectory of each module for the imported libraries. Now, you will have

to create libs subdirectory inside each of the module's directory and copy the respective libraries into them.

8.4 Manifest file

If you are to develop a Cardboard App for Android, you will have to declare certain permissions the application must have, in the Android Manifest file, in order to access the API and interact with the cardboard.

```
<manifest ...
  <uses-permission android:name="android.permission.NFC" />
  <uses-permission android:name="android.permiss-
sion.READ_EXTERNAL_STORAGE" />
  <uses-sdk android:minSdkVersion="16" android:targetSdkVersion="23"/>
  <uses-feature android:glEsVersion="0x00020000"
android:required="true" />
  <application
  ...
  <activity
    android:name=".MainActivity"
    android:screenOrientation="landscape">
    ...
    <intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER" />
      <category android:name="com.google.
intent.category.CARDBOARD" />
    </intent-filter>
  </activity>
</application>
</manifest>
```

Note the following:

- `<uses-sdk android:minSdkVersion="16"/>` declares that the user's device must be running API Level 16 (Jellybean) or higher. It is mandatory for devices to run on Jellybean to be able to run Cardboard Apps.
- `<uses-sdk android:targetSdkVersion="23"/>` declares that the particular app is targeted for devices with API Level 23 (Marshmallow).

- `android.permission.NFC` declares that the particular app would require access to the Cardboard's NFC tag.
- `android.permission.READ_EXTERNAL_STORAGE` declares that the app would like to pair the user's phone to the VR viewer.
- The intent-filter and specifically `com.google.intent.category.CARDBOARD` declares that the current activity is compatible with VR viewers.
- `<uses-feature android:glEsVersion="0x00020000" android:required="true" />` declares that the device must be able to support OpenGL ES 2.0 to run this app.
- `android:screenOrientation="landscape"` declares that the activity's screen orientation must be "landscape." It is mandatory for VR Apps to render on fullscreen and landscape and thus have landscape orientation.
- `android:configChanges="orientation|keyboardHidden"` declaration is not mandatory, but recommended.

8.5 Extend CardboardActivity

`CardboardActivity` defines the base for coding a cardboard app. It is the base activity that lets an app integrate with Cardboard devices. It exposes events to interact with Cardboards and handles many of the details commonly required when creating an activity for VR rendering. The `CardboardActivity` is based on sticky immersive mode i.e. it makes the content take up the whole screen, hiding the system UI.

Define a CardboardView

All the UI elements in the Android app are built using views. The Android Cardboard SDK provides the developers with its own view, `CardboardView`, that can be used for VR rendering. Check the example below to find out how `CardboardView` is defined in the activity layout xml file :

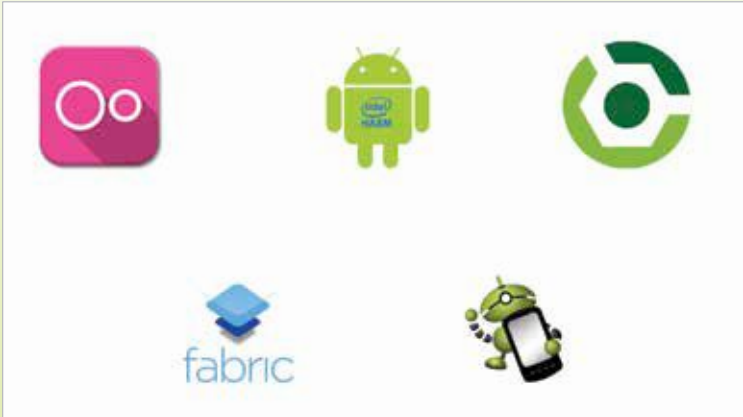
```
<com.google.vrtoolkit.cardboard.CardboardView
    android:id="@+id/cardboard_view
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" />
```

Now, in the main activity class initialize the `CardboardView` in the `onCreate()` method:

Conclusion

The market share of Android users in the smartphone segment is unpar-

alleled. Moreover, the power and feature which Android provides to its developers is also remarkable. The Android platform is growing day and night, by leaps and bounds and with it is growing Android Studio. Android Studio, if not already, is definitely the one single stop for Android Development IDE. Android Studio 2.2, which is scheduled to be released late in May 2016 is supposed to bring in a whole lot of new features like Firebase Integration, APK Analyzer and much more. Eclipse is undoubtedly a great IDE, but it is very generic and thus not the perfect platform for Android Development. Have you started using Android Studio yet? **d**



ESSENTIAL PLUGINS, LIBRARIES, TOOLS AND RESOURCES

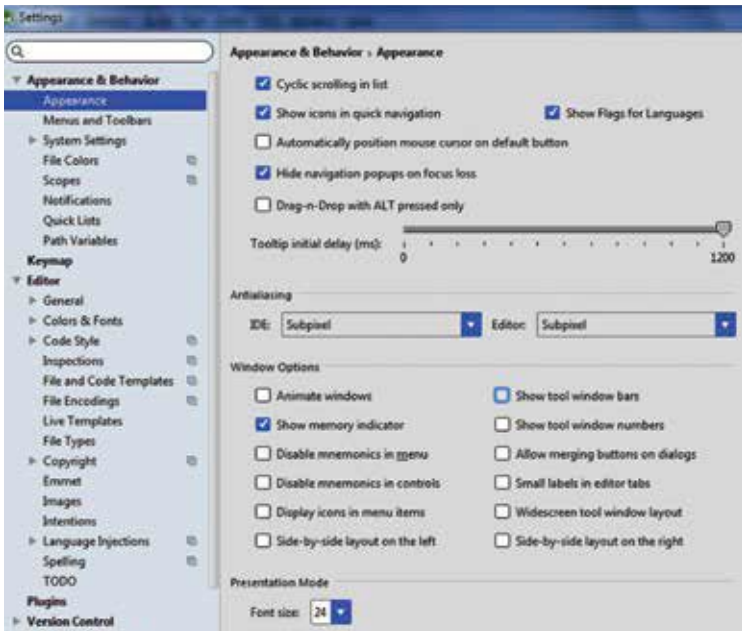
There's a plugin for everything and why shouldn't there be? After all they make your job easier. Here are some tools that you shouldn't miss out on.

Android Studio is a great tool to quickly build and deploy mobile apps for the Android platform. However, you need to tweak the tool in such a way that it runs faster. To improve the performance of Android Studio and also to enhance the quality of your mobile apps, you can make use of extensions such as plugins, third-party libraries and tools. Moreover, you can also enhance the performance programmatically. In this chapter, we will analyse the various ways by which you can tweak the performance of Android Studio including information about plugins, libraries and tools. Towards the end of this chapter, you will also find list of useful resources in the form of videos, podcasts and books.

Enhancing the performance of Android Studio

Memory Size

Increasing the memory size of Android Studio helps the tool to run faster. This can be done by opening the `/bin/studio.vmoptions` file and changing



Enable memory indicator to check how Android Studio is performing

the content from Xms128m and Xmx800m to Xms256m and Xmx1024m. In this case, we have increased the initial memory allocation pool. You can also provide heap values like Xms128m, Xmx750m and Xmx4096m, which is the maximum.

After modifying the parameters, you should save the file and restart Android Studio. You can also activate “Show memory indicator” from the appearance section by navigating to **File > Settings** so that you can monitor the memory consumption.

Improving Gradle Performance

You can improve Gradle performance by creating **gradle.properties** file in **C:\Users\\.gradle** and adding the following statements

```
org.gradle.daemon=true
org.gradle.parallel=true
org.gradle.configureondemand=true
```

In case your working environment is Linux, you need to save the file inside **/home/<username>/.gradle/** and **/Users/<username>/.gradle/** for Mac users.

Moreover, you can open **build.gradle** file and change jcenter to maven-central. You can also open **setting.gradle** located in the root folder and add the following statement

```
startParameter.offline=true
```

Activating offline work in Android Studio

You can enhance the performance of Android Studio by navigating to **File | Settings** and selecting Gradle option from the left hand side navigation panel. You need to check mark offline work option. You can also use **CTRL+ALT+S** combination to bring up Settings dialog.

Alternatively, you can select Compiler option and providing **--offline** in the command-line options box

Legacy Java for Mac

If you are using El Capitan and Android Studio runs slow, then you need to update your device to legacy Java by downloading Java for OS X 2015-001. You can also install this build if you run Yosemite, Mavericks, Mountain Lion and Lion. You can download legacy Java from <http://dgit.in/legacyjavamac>

Ignoring Thumbs.db

Sometimes, Windows automatically create **Thumbs.db** file. You can avoid this by providing **Thumbs.db** into “Ignore files and folders” option by navigating to **File > Settings > Editor > File Types**.

Creating keymaps & Scratch files

Android Studio enables you to create custom keymaps with support for high density Retina display. Moreover, you can make use of scratch files to quickly run and debug code.

Live Templates

As the name indicates, live templates can be used to automatically insert small chunks of code in Android Studio. To create a live template, you just need to provide the template abbreviations and press the Tab key.

Plugins to enhance Android Studio

HAXM

Intel Hardware Accelerated Execution Manager (HAXM) offers accelerated performance for Android SDK emulators on systems with Intel processors. The plugin make use of Intel Virtualization Technology to enable Android applications to run faster on simulated virtualization environments. To install HAXM, open the Android SDK manager and select the option captioned “Intel x86 Emulator Accelerator (HAXM installer)”. You have only downloaded the plugin. In order to actually install the plugin, you need to locate the folder manually and run the “intelhaxm-android” executable file.

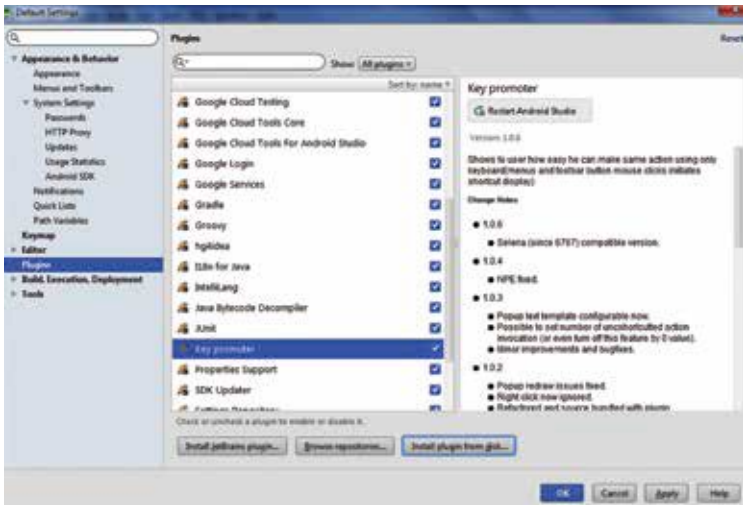
<http://dgit.in/intel-haxm>

Genymotion

Genymotion enables you run inside VirtualBox and comes with a wide range of sensors and features with which you can interact with a virtual cloud based Android environment and in several devices. Genymotion has an ability to catch bugs and integrates with Jenkins and Gradle. If you make use of Genymotion, you have around 3000 Android configurations at your disposal.

<http://dgit.in/genymotion-andemul>

Select **Tools > Android > SDK Manager** and select **Plugins** option from the navigation panel on the left side to install and manage plugins.



Easily install plugins and extend the use of Android Studio

Robotium Recorder

Robotium Recorder enables you to test native and hybrid Android apps on Android Studio, Eclipse including emulators and Android devices. You will be able to record test cases and user actions in addition to the ability to view various test scenarios in a detailed manner. Moreover, Robotium offers support for binary APKs and automatically detects resource IDs. The main advantage of Robotium is that you can monitor the working of your application as if it runs on your mobile device.

<http://dgit.in/robotium-and>

jimu Mirror

With the help of jimu Mirror, you will be able to test layouts and prototypes instantly on a real device. The plugin displays previews of Android layouts that automatically update as you start coding. Moreover, jimu Mirror enables you to share files, track changes and get instant feedback from your designer.

<http://dgit.in/jimumirror>

Strings-xml-tools

Android strings.xml tools is a free plugin, which is used to specifically manage the string resources in Android projects. The plugin offers basic

operations to enable you to sort entries in Android localization files including addition of missing strings. Even though the purpose of the plugin is limited, **strings-xml-tools** will be useful if you have large number of string resources in your project.

<http://dgit.in/strings-xml>

Android Code Generator

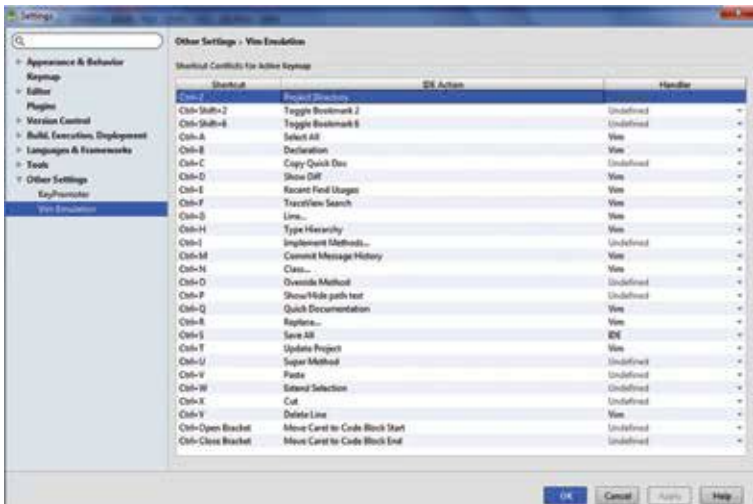
With Android Code Generator, you will be able to parse XML file and retrieve data of all the views that contain **android:id** attribute. In addition to collecting information about view's type and id, the plugin automatically generates activity, fragment and adapter class codes from layout.

<http://dgit.in/android-code-gen>

Android Holo Colors

With the help of Android Holo Colors, you will be able to easily customize your Android application by creating Android layout components from your own choice of colors. The plugin automatically generates all the required assets associated with XML and styles required for the project.

<http://dgit.in/android-holo>



Use keyboard shortcuts and perform source code action

IdeaVIM

IdeaVIM enables you to work with vim key bindings to Android Studio. If you love VIM editor, you will definitely love this plugin. Even though this plugin is not robust as the original vim, you will be able to enjoy coding by using this plugin.

<http://dgit.in/ideavim>

JRebel

JRebel helps you to develop Android applications by monitoring real-time changes in one or multiple devices. You can instantly view the changes as soon as you edit code or resource file without changing the state of the application.

<http://dgit.in/jrebel-and>

AceJump

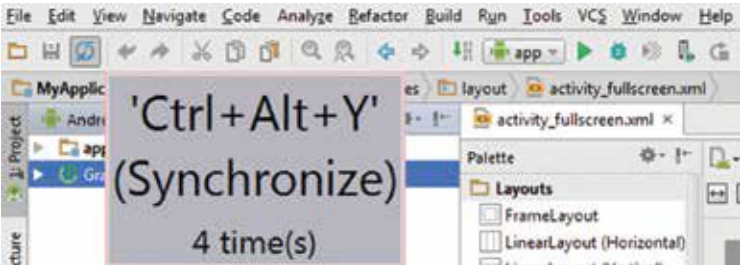
With the help of AceJump, you will be able to easily navigate the cursor point to any desired position inside the editor. You just need to press “ctrl+” and type a character to highlight them on the editor.

<http://dgit.in/acejump>

Key Promoter

Key Promoter automatically senses your toolbar action and reminds you if you click on the toolbar several times with the suitable keyboard shortcut. Even though it takes time to learn, keyboard shortcut is more productive than clicking on the toolbar.

<http://dgit.in/keypromoter>



Make effective use of Keyboard shortcuts with KeyPromoter

Android Material Design Icon Generator

By using Android Material Design Icon Generator plugin, you will be able to bring in material design icons to your Android project with support for vector icons.

<http://dgit.in/material-design-icon>

Android Drawable Importer

Android Drawable Importer enables you to easily develop Android apps for all display sizes and densities. The plugin provides an ability to not only import drawables in various resolutions but also scale a relevant image to a specific resolution

<http://dgit.in/drawable-imp>

Android Plugin for Gradle

Even though Android Studio integrates Gradle wrapper, the Android Plugin for Gradle runs independently. This means you can even build Android apps from the command line or on systems where Android Studio is not available.

<http://dgit.in/and-gradle>

Fabric for Android

Fabric is Twitter's mobile development platform and includes a wide range of tools to help mobile developers to build their apps in minimum amount of time. It includes several tools and plugins to help your Android apps to become more stable, social and profitable. Fabric also automatically inspects your source code and natively injects specific code easily.

<http://dgit.in/fabric-and>

Some of the other popular Android Studio plugins are Parcelable code generator, ButterKnife Zelezny, Otto, Code Glance, Relative Line Numbers, ADB Idea, Anko and Kotlin.

Libraries

Android Annotations

Android Annotations helps you to simplify coding and saves valuable development time. It modifies the Java code in such a way that the source code can be easily readable to any person by dividing them into separate lines with relevant comments.

<http://dgit.in/android-annot>

Picasso

Picasso enables you to easily handle images in your Android application. The plugin automatically handles ImageView recycling, complex image transformations including memory and disk caching.

<http://dgit.in/picasso-lib>

If you think, Picasso lacks customisation possibilities, you can make use of Universal Image Loader (<http://dgit.in/univ-image-loader>).

Sugar ORM

Sugar Object Relational Mapping (ORM) simplifies the process of establishing connectivity and management of SQLite database in Android. The plugin eliminates the need to write SQL queries and statements to work with SQLite database including management of object relationships. You can also consider using ActiveAndroid (www.activeandroid.com), which also does the same work as Sugar.

<http://dgit.in/sugarorm>

JodaTime

JodaTime provides wide range of options to integrate date and time functionality into your Android app. You can build apps with date, without time and vice versa, full date and time with time zone and much more. In short, this library extends the default data class included with Java.

<http://dgit.in/jodatime>

FreemiumLibrary

FreemiumLibrary helps you to build Android app by integrating premium features but your users will have to pay to use them via in-app billing v3 module from within the Google Play Store. The library also implements the AdMob library, which can be used to display ads to the user when they are using the app normally without payment.

<http://dgit.in/freemiumlib>

Gson

Gson is a Java library, which is used to serialize and deserialize Java objects from and into JSON. You will have to make use of this library if you frequently communicate with APIs.

<http://dgit.in/gson-lib>

```

1.  @Fullscreen
2.  @EActivity(R.layout.bookmarks)
3.  @WindowFeature(Window.FEATURE_NO_TITLE)
4.  public class BookmarksToClipboardActivity extends Activity {
5.
6.      BookmarkAdapter adapter;
7.
8.      @ViewById
9.      ListView bookmarkList;
10.
11.     @ViewById
12.     EditText search;
13.
14.     @App
15.     BookmarkApplication application;
16.
17.     @RestService
18.     BookmarkClient restClient;
19.
20.     @AnimationRes
21.     Animation fadeIn;
22.
23.     @SystemService
24.     ClipboardManager clipboardManager;
25.
26.     @AfterViews
27.     void initBookmarkList() {
28.         adapter = new BookmarkAdapter(this);
29.         bookmarkList.setAdapter(adapter);
30.     }
31.
32.     @Click({R.id.updateBookmarksButton1, R.id.updateBookmarksButton2})
33.     void updateBookmarksClicked() {

```

Simplify your Android code for easy readability

Retrofit

Retrofit helps you to cleanly organize API calls in a project by adding annotation to the request method and relative URL. If your Android app handles large chunks of code, you should employ this library.

<http://dgit.in/retrofit-lib>

JDXA

JDXA for Android is an object-oriented and flexible ORM library to simplify and accelerate the development of Android apps. The library not only enhances productivity but also decreases the difficulties associated with coding by avoiding long lines of complicated SQL code.

<http://dgit.in/jdxa-lib>

ButterKnife

ButterKnife is a library which works by adding `@InjectView` to your code for easy readability. In order to work with the library, you need to add the following code in the dependencies section of build.gradle file, which can be located inside `app/src`:

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.jakewharton:butterknife:5.1.2'
}
```

<http://dgit.in/butterknife-lib>

AndroidViewAnimations

AndroidViewAnimations (<http://dgit.in/androidviewanim>) enables you to add animations to your Android app by integrating several effects such as bounce, fade, flip, rotate, slide and zoom. On the other hand, ListViewAnimations (<http://dgit.in/listviewanim>) provides an ability to add animated items in a list.

AboutLibraries

The AboutLibraries library enables you to create a dedicated About section for your Android app by incorporating name, description, developer, license and version including the ability to automatically detect libraries.

<http://dgit.in/about-lib>

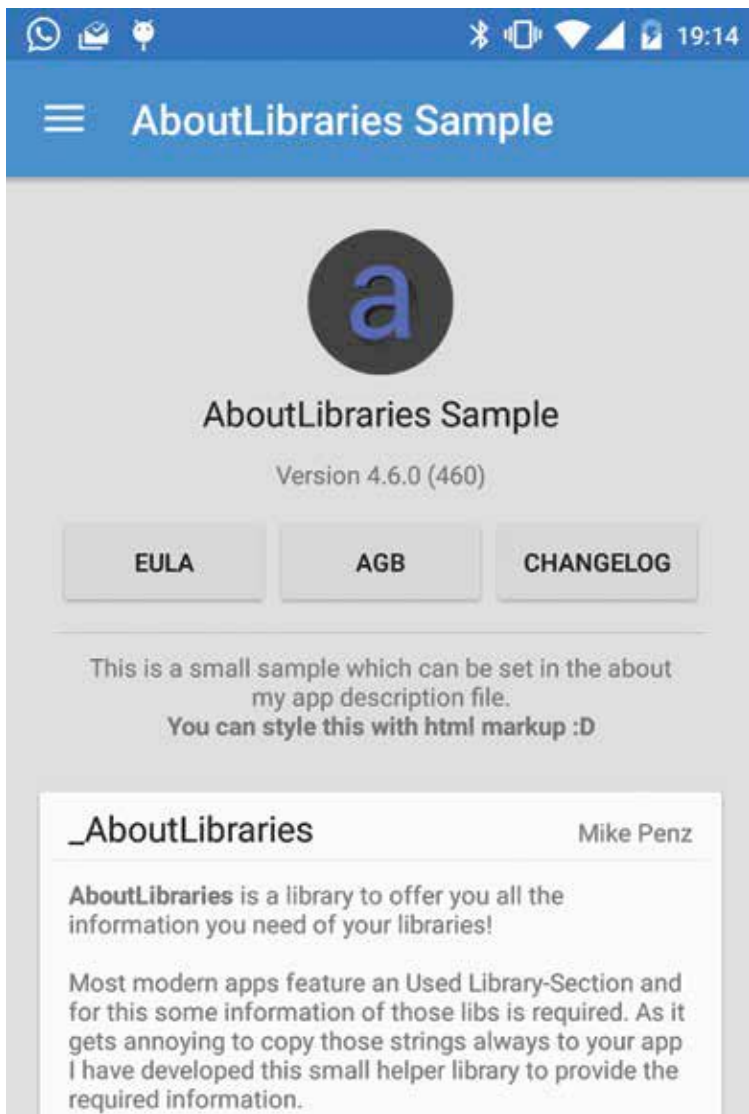
Some of the other notable libraries are IcePick, LeakCanary, Espresso, Robolectric, Dagger 2, RxJava, RoboGuice, Calligraphy, Chronos, Volley, Fresco, PinnedListView, ItemAnimators, GreedoLayout, ArcMenu, JazzyViewPager, ParallaxPager, FadingActionBar, AndroidCharts, AndroidPlot, and CamView.

Tools

Cloud Tools for Android Studio

Cloud Tools for Android Studio helps you to build backend services in Google Cloud Platform and it includes required modules and plugins for Android Studio.

<http://dgit.in/cloud-tools-and>



Easily create About pages with all the required information

Logcat

Logcat helps you to collect and view output from system debugging logs using the logcat command. To view log messages, you need to use the command from an ADB shell.

<http://dgit.in/logcat-tool>

Lint

The Lint tool automatically verifies your Android project source files for potential bugs and suggest improvements for security, performance, usability and others.

<http://dgit.in/lint-tool>

Desktop Head Unit

The Desktop Head Unit provides a facility to emulate an Android Auto head unit to enable you to run and test apps. Compatible with Windows, Mac and Linux, the Desktop Head Unit is considered as a replacement to previously used Android Media Browser and Messaging simulators.

<http://dgit.in/desktop-head>

Android Debug Bridge

The adb is a client-server based command line tool that enables you to easily communicate with an emulator or any Android device. The tool also includes a daemon, which runs on the background of the device or emulator.

<http://dgit.in/android-debug>

Draw 9-patch

The Draw 9-patch is a rich text editor tool which is used to create bitmap images. The created images are automatically resized to accommodate the view contents and display size. To work with the tool, you just need to drag and drop the image into the workspace window.

<http://dgit.in/draw-9>

Image Asset Studio

Image Asset Studio enables you to generate custom icons for your Android app from any existing image. The tool automatically generates a set of icons at the appropriate resolution for various display sizes.

<http://dgit.in/image-asset-studio>

Android Asset Studio

Android Asset Studio is a web based icon generator tool, which helps you to generate icons from existing images, cliparts or text based content. You will be able to create icons for various activities and can download them in a single ZIP file.

<http://dgit.in/android-asset-studio>

Crashlytics

If you build Android apps, you should employ reporting tools to track the app in real time. Crashlytics for Android SDK, part of Fabric for Android, automatically performs a detailed study and analysis of the source code and identifies the core issues affecting the code and mobile device. It also analyses the crashes and provides an automated way to deobfuscate stack traces.

<http://dgit.in/crashlytics-tool>

Some of the other tools used during the development process in Android Studio are bmgr, device monitor, dmtracedump, etltool, hierarchy viewer, hprof-conv and Android monitor.

Resources

Articles

Android Official Developer Tutorial - <http://dgit.in/android-official>

Android Studio Tutorial - <http://dgit.in/androidstudio-tut>

Android Studio tutorial for beginners - <http://dgit.in/as-beginners>

Getting Started With Android Studio 2 - <http://dgit.in/android-studio-2>

Introduction to Android development with Android Studio - <http://dgit.in/android-studio-intro>

The 12 Best Android Tutorials for First-Time App Developers - <http://dgit.in/12androidtut>

9 Best Free Android Studio Tutorials for Beginners - <http://dgit.in/9androidstudio>

How To Get Started With Android Programming - <http://dgit.in/android-prog>

Get Started in Android Studio - <http://dgit.in/get-android-studio>

Beginner's Android Studio Tonic eBook - <http://dgit.in/android-studio-tonic>

Books

Android Studio Cookbook - <http://dgit.in/as-cookbook>

Android Studio Essentials - <http://dgit.in/as-essentials>

Android Studio Application Development - <http://dgit.in/as-app-dev>

Murach's Android Programming - <http://dgit.in/android-murach>

Expert Android Studio - <http://dgit.in/as-expert>

Learn Android Studio - <http://dgit.in/learn-android-studio>

Android 6 for Programmers - <http://dgit.in/android6-book>

Courses

Exploring Android Studio - <http://dgit.in/as-explore>

Gradle Fundamentals - <http://dgit.in/gradle-funda>

Android Studio Essential Training - <http://dgit.in/as-essential>

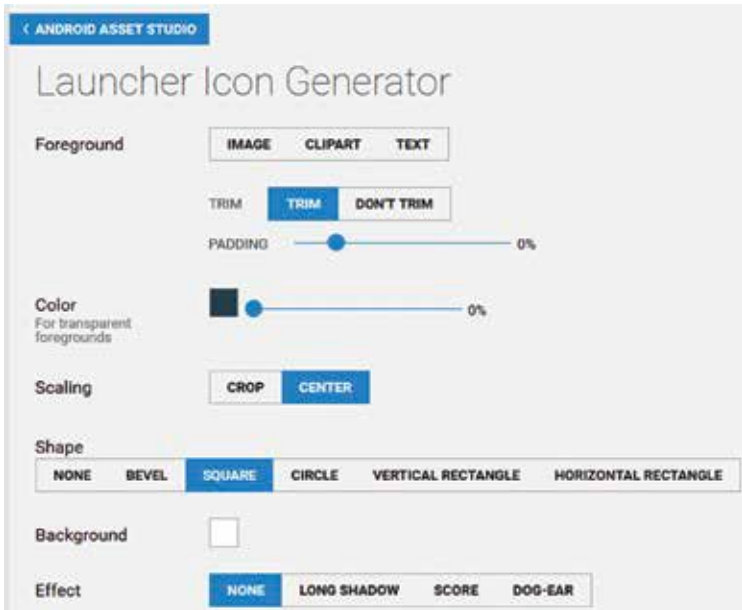
Developing Android Apps (Free) - <http://dgit.in/android-app-develop>

Android App Development (Free) - <http://dgit.in/designcoder>

Podcasts

Fragmented - An Android Developer Podcast - <http://dgit.in/fragmented-pod>

Android Central Podcast - <http://dgit.in/acentral-pod>



Quickly create icons for your Android mobile app development purpose

Android Developers Backstage - <http://dgit.in/android-back>

Videos

Android Studio by Donn Felker - <http://dgit.in/android-donn>

Android Studio Tutorials by Alex Cory - <http://dgit.in/android-alex>

Android and Android Studio: Getting Started - <http://dgit.in/android-started>

Android Studio Tutorial for Beginners - <http://dgit.in/as-tutorial>


What's New in Android Studio 2.1 - <http://dgit.in/android-studio21>

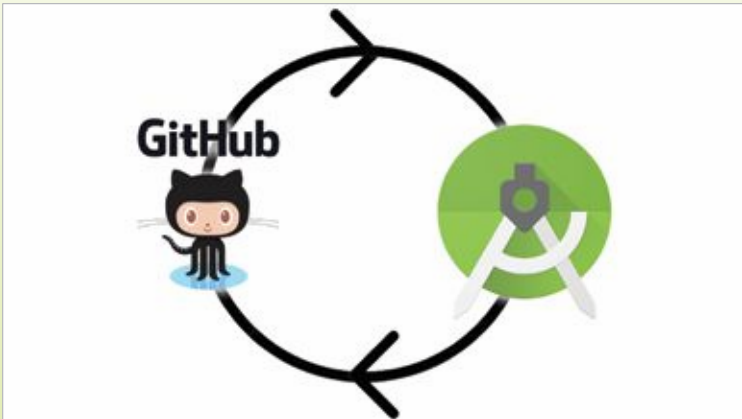
Android Studio App Development Tutorials - <http://dgit.in/astudio-appdev>

Android Tutorials for Beginners - <http://dgit.in/android-tut-beginners>

Android Development for Beginners - <http://dgit.in/android-dev-beginners>

Conclusion

Android Studio is an excellent development environment used for the building Android apps. You can further extend its functionality by using the plugins, libraries and tools covered in this chapter, which will not only simplify the overall development time but also helps you to integrate advanced functionalities into your app. If you would like to learn about Android Studio deeply, you can make use of the articles, books, videos listed under the resources section. That being said, the only way to update your knowledge of Android Studio is to keep track of the programming related websites, and YouTube. 



EXPORTING AND IMPORTING FROM GITHUB

There's no denying that version control plays a huge part in a software's development cycle. Here's how you can use GitHub with Android Studio.

Now that you have developed your application, it's time to share it with the world. To do so, you can use the popular version control system - Git, and a web-based Git-powered repository hosting service - GitHub. In this section, you'll learn the in's and out's of Git and GitHub, and how to export and import repositories.

What is version control?

Version control system (VCS) keeps a track of the developer's code as it progresses. It maintains a special database in the central repository that stores the modifications made to the code with unique time stamps. In case of mistakes, the developer can retrieve the earlier code from the database without losing much progress. Additionally, it organises the project as a tree structure where each developer confines to a leaf, and the central repository is the root of the tree. This makes it easy to track each developer's work and prevents concurrent work from conflicting as each developer's work is synchronised to the central repository after it is completed. There are many systems which offer these features but we'll be choosing Git because of its popularity and ease of use.

What is Git?

Git is an open source version control system which is currently recognised as de facto standard in software firms. It is available in the form of command line as well as a GUI tool for Linux, Windows and Mac OS X operating systems. Although the GUI version is simpler to use, developers prefer the command-line tool. In addition to all the primary VCS features, Git has lot more to offer in terms of functionality, performance, security, and flexibility. In comparison with the alternatives available, Git's raw performance is better. It uses deep learning algorithm that makes committing, branching, merging, etc lot more optimised than its competitors.

What is GitHub?

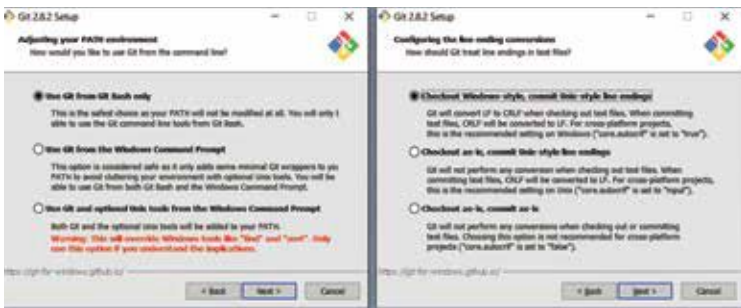
GitHub is a Git-powered, web-based repository hosting service which can be accessed through github.com. It hosts millions of open-source software projects from millions of developers across the world. Git maintains version controlling locally on your PC, however, in contrast, GitHub keeps a global copy of your project on the GitHub servers. Keeping a centralised copy of your project on a public server allows for collaborative development for project contributors from across the world. It also makes sharing your ready-to-execute project easy. The foremost aim of making the code open-source is code reusability.

In this section, we will be guiding you for install and managing Git locally on your PC, creating a personal repository on GitHub, and pushing and pulling projects from it.

Installing Git on your PC

To get Git on your Windows system, go to <http://git-scm.com/downloads> and download the latest version of Git and run the installation wizard. During the installation we recommend keeping the default settings as you're a beginner. Once the installation is done you're ready to use.

To get Git on your Ubuntu system, go to **Terminal** > type “`sudo apt-get install git`” > **Enter**. It will download all the necessary packages and make Git ready for you. If it ends up with an error message, type “`sudo apt-get update`” > **Enter** and later type “`sudo apt-get upgrade`” > **Enter**.



Keep the default settings as shown in the figure

Setting Up a Local Repository

After everything is done you can start off by creating a local repository. You can create any folder on your system as a repository. There can be any number of repositories on your system. We recommend you to make your existing project folder as the repository.

To do that, in the File Explorer, navigate to the Android Studio workspace and open your project folder. **Right-click** > **Git Bash Here**. This will open a command-line interface which is your playground for the Git. The command-line looks familiar to the Windows Command Prompt. In the command-line you'll see your PC name followed by the path of your project directory. To make the current directory as the Git repository type “`git init`” and hit Enter.

You'll see a message “**Initialized empty Git repository in <your path>**” which states your repository is successfully created. You can verify it by seeing a hidden folder named “.git” in your project directory. At anytime you can delete this folder to destroy the repository.

Saving Changes

Your repository is now ready to track files that you add, delete, and modify. To start with type the command **“git status”** that will give you the list of all the untracked files and folders in the directory. Type **“git add .”** which will track all the untracked files. This process is called staging. Once the files are staged the VCS tracks for the modifications. If you modify your Android code and type **“git status”** it will show you the name of the Java file that you modified. Once again you need stage the file. This time since it is a single file you can type **“git add <your filename with extension>”**. Every time you modify a file

```

MHD2054@Studio Projects/android:
$ git status
On branch master
nothing to commit but untracked files present (use "git add" to track)
$ git add settings.gradle
MHD2054@Studio Projects/android:
$ git status
On branch master
nothing to commit
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   settings.gradle
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore
        .idea
        build
        gradle
        gradle.properties
        gradlew
        gradlew.bat
        settings.gradle
MHD2054@Studio Projects/android:

```

Status of your repository before and after staging.
Inspecting a repository

being tracked, git status command will show the name of the file which is modified. You need to stage the file again.

Staging the changes doesn't save the state of your project, committing the changes does. Once you are done modifying the file and want to permanently save the changes with a note and a timestamp, you should use the **“git commit -m 'a personal note' ”** command. This command commits all the files that are in the

staging area with a personal note written in single quotes.

Git offers commands to inspect our repository. These include the one we have already used **“git status”**, and a new command **“git log”**. Since we have already used **“git status”** we know it is used to know the status of our files - whether they're committed or staged.

The **“git log”** command on the other hand is used to display the entire commit history. This shows all the commits that you made before in the repository along with the timestamp and the personal note. The log command makes it handy to get the previous versions of the code if you have messed up something.

Viewing old commits

The log command will list your commit history. If you want to see the code in the previous commit without altering the current modification you can use the command “`git checkout <commit> <filename>`”. The commit in the above command is the unique commit ID that you’ll see above every commit listed in the commit history. The checkout command makes your working directory match the exact state of the old commit. You can do anything you want including compile, modify, rerun, etc.

Undoing changes

If you want to undo the last commit made you can use “`git revert <commit>`”. Where the `<commit>` is the unique commit ID. You can permanently undo your commit by using “`git reset`”. Care must be taken while using this command as you are not in the position to get back the data. The command “`git reset <file>`” will remove the specified file from the staging area, but leave the working directory unchanged. The command “`git reset`” removes all the files from the staging area to match the last commit made. The nuclear weapon here is “`git reset --hard`” which clears the staging area and rolls back the working directory to the last commit made.

```

MINGW64/d/git
varad@AFTOP-000615F4 ~$ git log
commit be467878c80a5261ba5dc9ff631d1997e16c73cf0
Author: varad@hoodnar1 <varad@hoodnar1@gmail.com>
Date: Tue May 17 01:30:33 2016 +0530

    added varad

commit 13ba46c0eb3ccc00e55f6d31992f5fab3e3a137e
Author: varad@hoodnar1 <varad@hoodnar1@gmail.com>
Date: Tue May 17 01:30:03 2016 +0530

    added hello

varad@AFTOP-000615F4 ~$
  
```

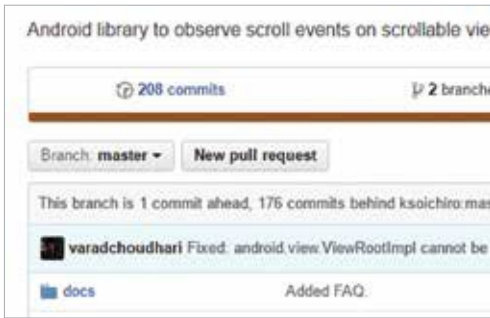
The unique commit located above the author name should be used for “checkout”.

Cloning projects or modules from GitHub

We talked about working on the local directory where all the changes made to the code were hardcoded. You are already aware about GitHub – a public Git repository from where you can get source codes for your project, and also share yours. Cloning a repo is different than forking it. Forking makes you a contributor of the original project, whereas, cloning downloads the project which you can use as a part of your existing project. Cloning an existing project from the GitHub to your local repository is relatively easy.

All you need to do is execute a single command that downloads the project right into your working directory.

In the command-line type “**git clone <clone URL>**”. The clone URL is the direct link to the project repository that you wish to clone. The URL is provided by GitHub for every repository it hosts. Make sure you get the correct clone URL which usually ends with “.git” extension. When that’s done you will see a new folder into your existing repository named after the repository you have cloned. This is the project folder containing all the source code that you need.



The “New pull request” button located above the files list should be used to send a request.

This approach is the command-line way to clone projects. It can be used to clone any project irrespective of its programming language from the GitHub. Since we are working with Android Studio we get a built-in way to do the same. As for the prerequisites, you need to

download and install GitHub for Windows from <https://desktop.github.com/>.

After you are done, launch the Android Studio and close your existing project from **File > Close Project**. This will take you back to the Welcome Screen. In the Welcome Screen go to “**Check out project from Version Control**” > **GitHub**. Enter your GitHub credentials and Login. Enter the clone URL in the Git Repository URL field and click Test. If the test is successful hit the Clone button and the project is cloned right into the directory that you specified.

Forking projects or modules from GitHub

Added advantage of GitHub is you can contribute to anyone’s project by forking them. Forking a project creates a copy into your repository where you can modify the code and push it back. This makes you a contributor to that project. You need a GitHub account to fork projects. Create an account on the GitHub on github.com if you don’t have it. Login to your account and navigate to the project page which you want to fork.

Since you have logged in you'll see a "Fork" button located at the top-right corner, click on the button. This will make your own copy of the project to your account. Verify it by going back to the dashboard, if the project exists, you have successfully forked it. To start the contribution, clone your copy of the repo on your system and modify it.

Follow the version control process of adding and committing the changes as you modify the project. Once you have made your final commit you need to push the project back to the remote GitHub server. This should be done using the command "`git push origin master`". It will ask for your GitHub username and password, enter it and you're done. The changed code is updated into your own copy of the repository. Go to your GitHub dashboard on the website, navigate to repository and check out the new changes.

The changes you made to the code are stored in your copy of the repository. Meaning, the original repository doesn't contain the additions/modifications. For this you need to create a pull request. You'll find a button "New pull request" in the repository page. Click on it, leave a comment, and create a request.

A pull request notifies the owner of the repository about changes made to the project which could be worth merging with the original file. If the

owner approves your pull request, your changes will be readily reflected into the original repository. This is how contributing to an open-source project works on GitHub.



Repository URL link can be directly copied to the clipboard by using the copy button located next to the link.

Sharing and uploading your Projects to GitHub

The app that you just developed using Android Studio is ready to submit to GitHub. This will not only make your source code open-source, but also bring contributors who can identify bugs, fix them, and make your application a better performer than before. Sharing your code is a good practice as it enhances reusability.

To start with this you need to create a new repository on GitHub. For this login to your GitHub account and navigate to Repositories. Since you are new, the list of repositories might be empty. Click on the "New" button located at the top-left which will let you create a new repository.

You'll be asked to enter the name of the repository. It can be anything as long as it doesn't conflict your previous repositories (if any). Add a short description about what the repository is about. Description is optional, however, we would recommend to write one as it gives a short insight about what the repository does.

Then you'll be asked to choose the type of the repository. It can be Public - accessible by everyone, or Private - accessible only with the authorised people. A public repository is always preferred as it is free to host. A private repository is generally used by corporate companies for internal version controlling, hence it is available as a paid option.

The option "Initialize this repository with a README" creates a blank 'read me' file that you need to fill with the 'read me' document for your project. Ignore the other options and click on "Create repository". A blank repository will be created. You need to push your project from your PC to this repository.

To upload your project to GitHub you should have Git initialised into your project folder and all the changes committed. If you have it ready, you are ready to upload. In the Git bash type "**git remote add origin <repository URL>**". The repository URL is provided when you create a new repository. It ends with ".git" extension. If you are unable to find it, navigate to your newly created repository on the website and copy the repository URL. For example,

```
git remote add origin https://github.com/varadcboudbari/Digit.git
```

You'll be asked to enter your GitHub credentials. This will stage your project files to the remote server. However, the project is not yet uploaded. To initiate the upload type "**git push -u origin master**". This pushes your entire local repository to the remote repository on the GitHub server.

This approach is the command-line way to push projects. It can be used to push any project irrespective of its programming language to GitHub. Since we are working with Android Studio we get a built-in way to do the same.

Open your Android Studio project and navigate to VCS menu > **Import into Version Control > Share Project on GitHub**. Enter your GitHub credentials if asked and click on **Login**. If you have checked "Save Password" you'll be asked to enter a new password for Android Studio's password database.

Once you're done, Studio will ask you to enter a repository name, add a description and click on Share. This approach eliminates the need to create



Adding a git ignore list will not stage the files with selected extensions.

a commit message and click “OK”. That’s it. Your project will be pushed to GitHub.

Working with GitHub Repositories

GitHub hosts millions of repositories. From individuals to corporates like Google, Facebook, etc. have their repositories. Google’s account <https://github.com/google> hosts hundreds of repositories for public use. Working with GitHub repositories is simple. In addition to pulling, pushing and contributing these repositories help keeping the quality of code with the ability to track issues. There is a separate section for every repository which is used to keep track of issues and bugs related to the project. The URL <https://github.com/google/google-api-php-client/issues/> will guide you to the list of issues with the current repository. Issue can be raised and fixed by anyone. Each issue is maintained as a topic that makes discussion over it easy. Feature called “Watch” which notifies you about the discussions on that repository. The “Star” button corresponds to liking the repository. **d**

Control Panel > System and Security > System

Search Control Panel

digit.in/review

NEWS AND REVIEWS

Comprehensive news, unbiased reviews and ratings to help you make the right purchase

EENIE MEENIE MYNIE MO.

Control Panel > System and Security > System

Search Control Panel

digit.in/download

DOWNLOADS

Check out the latest software downloads, games for your Windows, Linux, Mac and PDA/mobiles

All this and more in the
world of Technology

VISIT
NOW

 digit.in

Join 800K+ members of the digit community



http://www.facebook.com/thinkdigit

facebook

digit.in

Digit 1,234 likes

Your favourite magazine on your social network. Interact with thousands of fellow Digit readers.

- Wall
- Info
- MEMBER
- Cancel Post
- Schedule
- DUPLICATE
- Cancel T
- Photo
- Video

http://www.facebook.com/IThinkGadgets

facebook

I Think Gadgets 1,234 likes

An active community for those of you who love mobiles, laptops, cameras and other gadgets. Learn and share more on technology.

- Wall
- Info
- Photo
- Video
- Events
- Groups

http://www.facebook.com/GladtobeProgrammer

facebook

Glad to be a Programmer 1,234 likes

If you enjoy writing code, this community is for you. Be a part and find your way through development.

- Wall
- Info
- Photo
- Video
- Pages
- Events

http://www.facebook.com/devworx.in

facebook

devworx 1,234 likes

devworx, a niche community for software developers in India, is supported by 9.9 Media, publishers of Digit

- Wall
- Info
- Friend Activity Log
- Questions
- Photos
- Pages
- Streams
- Home