For the last 4 months I have been working on my first Android app. It took some work to navigate the Android SDK and learn how it uses some key programming concepts. But I collected a lot of useful resources in the process that I hope can be of help to others as well. In this post I will try to put these resources together and lay out a starting guide for those who are interested in creating their first app.

Here is the tl;dr guide:

- If you are starting from scratch, first get the Android SDK here: http://developer.android.com/sdk/index.html

- If you are really starting from scratch, you should actually learn Java first

- Then consider starting with the following video tutorial: http://www.youtube.com/watch?v=Z149x12sXsw This is a great series that will guide you through the basics of an Android app. I recommend that you not only watch the video but repeat what he is doing simultaneously.

- A great companion book is [The Busy Coder's Guide to Android Development](http://commonsware.com/Android/). If you don't want or can't buy the book, consider looking at its many code examples available for free at: https://github.com/commonsguy/cw-omnibus/

- Learn how to search for solutions before you ask questions

- Learn how to debug your code

- Once you learn the basics, invest some time in understanding some key concepts: configuration changes, usage of threads and broadcast receivers, usage of the support library, and the implementation of Parcelable

- Your friends can be a great source of help and feedback. Ask kindly, do not harass them.

- Unless your app is the next big thing since sliced bread, do not bother too much with announcing it before its ready. There are too many "pre-apps" out there, nobody cares.

If you are now going to read this full guide, thank you for your attention. We will go through detailed information about each of the points above, plus some more.

*
What to Learn First

Although most Android apps are developed in Java (and hence most documentation and help are also available for this language) there are alternatives out there if you don't want to learn it. But if you are a complete beginner, I think it is best to stick with it. At the same time, I don't think that you need to learn it FIRST and only then move to learning to develop for Android. You can do both simultaneously. The only exception is if you never programmed at all before. If that is the case (or if you are interested in some basic information), consider looking at these first:

- [Java Language Fundamentals] (http://en.wikibooks.org/wiki/Java_Programming/Language_Fundamentals)
- [Dealing with Exceptions] (http://en.wikibooks.org/wiki/Java_Programming/Exceptions)
- [Understanding Threads and Runnables] (http://en.wikibooks.org/wiki/Java_Programming/Threads_and_Runnables)
- [Using Interfaces] (http://en.wikibooks.org/wiki/Java_Programming/Interfaces)
- [Using Event Listeners] (http://docs.oracle.com/javase/tutorial/uiswing/events/intro.html)
- [Creating your own Callbacks and Listeners] (http://stackoverflow.com/a/1477229/362298)

There is a lot to go through here but the point is not be overwhelmed by it. It is just important to realise that these concepts exist. Spend perhaps a couple of hours every day on them until you feel comfortable understanding what each of them are for. In parallel, you can continue focusing on the Android part. Of course if you come from another Object Oriented language, most of the concepts above should be familiar to you already. In that case, you really don't need to bother. You will pick up the language syntax as you go.

Understanding the concepts is a good thing because it lets you search for practical ways to achieve something. Suppose you want to run a piece of code that should run in parallel with another. If you know the concepts, you can just search for them adding the "java" or "android" keyword at the end to get plenty of results that are likely to apply to your needs. Almost everything you can think of doing has already been tried and documented by someone else. You just need to know how to find it. Which leads me to my next set of things you may consider to learn first:

- Learn how to search. I cannot stress this enough. Seriously, just Google it. If you are stuck or in doubt, search for your question. And most of all: Search before you ask.
- Learn how to debug your code. There is a surprisingly high number of learners who do not debug their code at all and can't really [understand their code's log](http://stackoverflow.com/questions/6065258/how-to-interpret-logcat). If you are using Eclipse, [check this out] (http://stackoverflow.com/questions/8551818/how-to-debug-android-application-line-by-line-using-eclipse).

Speaking of Eclipse, the IDE most people still use to program for Android, whether you want to use it or not is up to you. Bear in mind the following though:

- The alternative IDE most people talk about, Android Studio (which is based on IntelliJ), is still in an **early access** preview. The official Android page has the [following warning about it](http://developer.android.com/sdk/installing/studio.html): "Android Studio is currently available as an early access preview. Several features are either incomplete or not yet implemented and you may encounter bugs."
- Most tutorials and documentation for beginners are still based on the use of Eclipse. All the resources I talk about here which are connected to an IDE, refer to Eclipse.

I personally chose Eclipse that comes bundled with the [Android Developer Tools (ADT)](http://developer.android.com/sdk/index.html) and it is working fine for me most of the time. When it doesn't, I have to either just clean the code or on the extreme case restart it. But that is rare.

*

Android Resources for Beginners

With the basics in mind, it is time to start coding your first Android app. To begin, download and install the [Android Developer Tools] (http://developer.android.com/sdk/index.html). Then consider the following tutorials and resources:

- [The Android official training guides](http://developer.android.com/training/index.html) are a good place to start. The [Building Your First App](http://developer.android.com/training/basics/firstapp/index.html) lesson is very easy to follow and already gives you a good understanding of some key concepts of the Android SDK.

- [The Android Development Tutorial](http://www.youtube.com/watch?v=Z149x12sXsw) by Derek Banas is great for those of you who prefer video lessons. It has 25 video lessons in total ranging from 10 to 30 minutes each. I used the first lessons to get a feel of the development process, specially to understand layouts.

- [Common tasks](http://developer.android.com/guide/faq/commontasks.html) are a useful list of typical things you can do in your app and how to develop them.

- [Android Guides](https://github.com/thecodepath/android_guides/wiki) give you even more explained code recipes and examples on how to build most common things in an Android app. I wish this was published when I started learning. It would have certainly helped.

- [Tasker](https://play.google.com/store/apps/details?id=net.dinglisch.android.taskerm&hl=en) is a great app available on the Google Play Store. You probably have heard of it. This app won't teach you anything about how to develop Android apps but it will show you what an Android app can do with your phone. I learned a lot from it. Whenever I realised I wanted to do something specific, I knew I could do it because I had done it before on Tasker.

-
[DevAppsDirect](https://play.google.com/store/apps/details?id=com.inappsquared.devappsdirect&hl=en) is another great app you can get from the Play Store. While it also won't teach you how to develop an app, it will show you what is available out there. The app maintains a list of open source libraries you can use in your project for a variety of purposes. Knowing what you can reuse will save you a lot of time in the future.

In addition, my favourite book is by far [The Busy Coder's Guide to Android Development](http://commonsware.com/Android/). The book is comprehensive with its over 2,400 pages but starts with the basics, explaining the concepts. It also has those "do-it-yourself" tutorials to help you retain what you are learning. The book is a bit expensive but it comes with a one-year subscription to keep it updated during the period.

All code examples are free and can be found here: https://github.com/commonsguy/cw-omnibus/ Even if you don't buy the book, consider browsing through some of the examples there to learn how other programmers do things. Finally, Mark Murphy, the author of the book, is helpful whether you contact him by e-mail or on the [Stackoverflow website](http://stackoverflow.com/). Check out [his profile](http://stackoverflow.com/users/115145/commonsware). He is in the all time top-10 ranking list there.

*
Concepts to Keep in Mind and Focus on

Now that you have learned the basics and are capable of building some apps, consider focusing your attention on some specific features of the Android SDK. The ones listed below are some of the important ones that represent key building blocks of most Android apps but, at the same time, are often misused or ignored.

*
Dealing with configuration changes

The most common of these is caused by orientation changes. Whenever there is an orientation change, your activity needs to be destroyed and recreated to address the changes in layout. This means you need to handle this recreation process yourself, making sure your app doesn't crash. A lot of beginners (myself included) consider simply disabling these changes but this is consider a bad practice. Besides, even if you do disable orientation changes, there are other things that can cause configuration changes that you need to handle anyway.

Here are two good posts discussing this further: http://stackoverflow.com/a/582585/362298 and http://stackoverflow.com/questions/1111980/how-to-handle-screen-orientation-change-when-progress-dialog-and-background-thre

To force yourself to catch problems sooner than later, consider the following tip from a previous [post here on Reddit](http://www.reddit.com/r/androiddev/comments/19143c/is_your_app_stateful_test_your_might/). You can enable a developer setting to not keep activities, so that they always get destroyed and recreated.

And more specifically, here are two examples of dealing with this problem when using a FragmentPagerAdapter, a common use case:

- http://stackoverflow.com/questions/17629463/fragmentpageradapter-how-to-handle-orientation-changes and

- http://stackoverflow.com/questions/18425646/fragmentpageradapter-not-restoring-fragments-upon-orientation-change

In a more abstract sense but still related to the topic, [an old post on avoiding memory leaks](http://www.curious-creature.org/2008/12/18/avoid-memory-leaks-on-android/) is probably worthy of your time as well. This was written by Romain Guy, a Google employee very active in the Android community. It is certainly worth [following him on Google+](https://plus.google.com/+RomainGuy).

*
Parcelables are fun

You can pass objects from one activity to another in a Bundle if your class implements the [Parcelable interface](http://developer.android.com/reference/android/os/Parcelable.html). Parcelables were designed specifically for performance and should almost always be used instead of Serializable. Here is a a good page explaining how to use it: http://shri.blog.kraya.co.uk/2010/04/26/android-parcel-data-to-pass-between-activities-using-parcelable-classes/

You can also have inheritance while using it without adding too much of an overhead to the children like this: http://stackoverflow.com/questions/20018095/parcelable-inheritance-issue-with-getcreator-of-heir-class/20018463#20018463

One thing you MUST always keep in mind is the order with which you write to the parcel and read from it. That order must be consistent. Otherwise you will start getting very crypt error messages which are very hard to debug.

*
Dealing with threads
Remember in the beginning of this guide that one of the recommendations was to read about threads and runnables? Yes, Android is full of features to help you deal with them. This is a very important aspect of Android development because your app has to give snappy responses. So all your heavy work such as database operations and network access need to be done in a separate thread.

For this reason it is important to know when to use a Service, a Thread, an IntentService or an AsyncTask. Learn about to them, check examples, and make sure you use them whenever appropriate. Perhaps the best place to start is this post summarizing your options: http://techtej.blogspot.com.es/2011/03/android-thread-constructspart-4.html You will realise that knowing about callbacks and listeners will be useful here too.

*
Setting up broadcast receivers
Broadcast receivers are a great way to have your asynchronous tasks (refer to the previous topic above) communicate with the main thread or to receive push notifications from your phone. It is a powerful feature to understand and use. Consider reading about it in the [official documentation](http://developer.android.com/reference/android/content/BroadcastReceiver.html). Also, refer again to the list of [common tasks](http://developer.android.com/guide/faq/commontasks.html) to get to know how to use them.

*
Wake locks do not need to keep you awake
Wake locks can be nasty little things. If you rely a lot on wake locks to execute background tasks while the phone is sleeping, you need to manage them carefully. Otherwise you risk having the phone awake consuming battery for no reason, and having your app listed as a battery drainer in the phone stats. To avoid, a tiny but very useful library came in handy to me.

It is explained in detail in book I mentioned above but you can download its source code for free at https://github.com/commonsguy/cwac-wakeful. Instead of implementing an IntentService (see previous topic for the difference between Service and Service) you implement a WakefulIntentService which handles the lock acquisition and release for you. Simple and elegant.

*
Supporting multiple devices and Android versions
In my app I chose to support Android API 9 and above. Why? Well, according to the latest statistics I can see in my Developer Console, that represents almost 97% of all active Android devices out there. I also chose not to release a tablet version for now. The number of Android tablets is still much lower than that of Android phones. Take this year's Q1 sales for a reference.

[28 million Android tablets were sold](http://www.businessinsider.com/android-ahead-of-ios-tablet-market-share-2013-5) as opposed to [156 million Android phones](http://www.gartner.com/newsroom/id/2482816) sold in the same period.

Supporting older Android versions turned out to be easier than I thought initially. I am using the [ActionBarSherlock](http://actionbarsherlock.com/) for providing the same UI experience in terms of ActionBar layout. I know now that the official Android Support Library has a similar library called [ActionBarCompat](http://android-developers.blogspot.pt/2013/08/actionbarcompat-and-io-2013-app-source.html) which is probably worth considering to avoid relying too much on third-party libraries.

Aside from the ActionBarSherlock, though, most things in the UI can done to all API levels using the [Support Library](http://developer.android.com/tools/support-library/index.html). You just have to make sure to use the right sets of classes. For example, instead of using "android.app.Fragment" when creating a new fragment, you use android.support.v4.app.Fragment instead.

*
Choosing a Paid vs. Free version option
There are three main options developers consider when choosing the Free/Paid app route each with its own advantages and drawbacks.

- The first is to have two separated apps. One is free, the other is paid. Do handle your project's code, you need to create now 3 projects: One is a library project that contains most of it. The other two are just to handle the specific package names and feature control for your specific free and paid apps respectively. The obvious disadvantage here is that you will have two apps in the PlayStore, splitting the statistics, reviews and ratings. You also have to handle data migration in case the user starts with the free version then switches to the paid one.

- The second is to have your app as free with the paid features unlocked when the user buys a "pro key" app of yours. You can do this by checking if the "pro key" app is installed and matches the signature of your main app. [Here is a post](http://stackoverflow.com/questions/3062946/how-can-i-use-the-paid-version-of-my-app-as-a-key-to-the-free-version) that explains how to do it. This version is very simple to setup and implement and avoids the "3 projects hack" above. It also reduces the split of reviews since your "pro key" will get less attention. However, you are forcing the user to be stuck with two apps which might be confusing. This can be mitigated by either 1) adding some sort of check and explanation on your "key app" or 2) hiding it from the app drawer.

- Finally, the third solution is to implement "in-app billing" and have your user buy the "pro features" of your app. This is probably the most elegant solution but also the trickier one to

implement. It is also only supported for apps sold over the Google Play store. You can find out more about in-app billing here:
http://developer.android.com/google/play/billing/billing_overview.html

*
Do not be afraid of Android's source code
Once you feel comfortable developing apps, you could also consider exploring bits and pieces of the Android source code. There is a lot to learn from it. I learned a lot from the sources of the Parcelable interface and Parcel class, for example. The code is usually well documented (although a bit out-dated at times) which makes it generally easy to understand.

*
Publishing and Advertising your app
This is [a great
guide](http://www.reddit.com/r/androiddev/comments/1lgbh7/advice_for_all_new_devs_intending_to_publish_to/) here on Reddit explaining how to publish your app on the Play Store.

Regarding advertising, there are tons of books, blog posts, discussions and what not out there trying to tell you what to do to advertise your app. Here is what I am doing.

I did announce my app last month before it was ready. I had one of those "coming soon" templates with some screenshots of it. The goal was mainly to use this page to advertise the app on web sites dedicated to announce the release of new apps. The places I tried include:

- [Betali.st](http://betali.st/)
- [Launching next](http://launchingnext.com/)
- [Pre-apps](http://www.preapps.com/)

There many others you can consider but from my experience with these, I wouldn't bother. There are too many apps being listed in those sites and competition for attention is fierce. So far, I got more attention from my posts on Facebook to my friends. Friends can help you with word-of-mouth and feedback from usage. My friends were great and I'm very thankful to them. My approach was to ask them kindly but without insisting. I didn't want to bother them so I focused my effort on the feedback of those who could help.

I am now paying for small ads on Facebook but that doesn't seem to be working very well. When you start setting up an ad for your page, Facebook gives you an estimate of how many "likes" you will get per day. In may case that estimate was between 5 and 19 likes. I am running the ad for the third day now and so far I got 6 likes that came from the ad. Not exactly a reliable estimate.

In the coming days I will try also Google Ad Words and ads here on Reddit. I will be glad to share the numbers with you once I get the results if you are interested. I don't know yet what to

do once I get this information. I guess it will depend on how much feedback I get from the app's current release. Your suggestions are more than welcome if you have any.

*

Wrapping up

If you read until this far, thank you very much again. I hope I gave you some new and interesting information to work with. At the same time, I'm sure I'm missing tons of other stuff. If there is something you feel like I should include in this guide, please let me know in the comments. I will be glad to improve it.

In case you are wondering, the app I developed is called [The Talker App](https://play.google.com/store/apps/details?id=com.thetalkerapp.main). I will be very happy if you could check it out and give me some feedback to improve it.

Cheers!