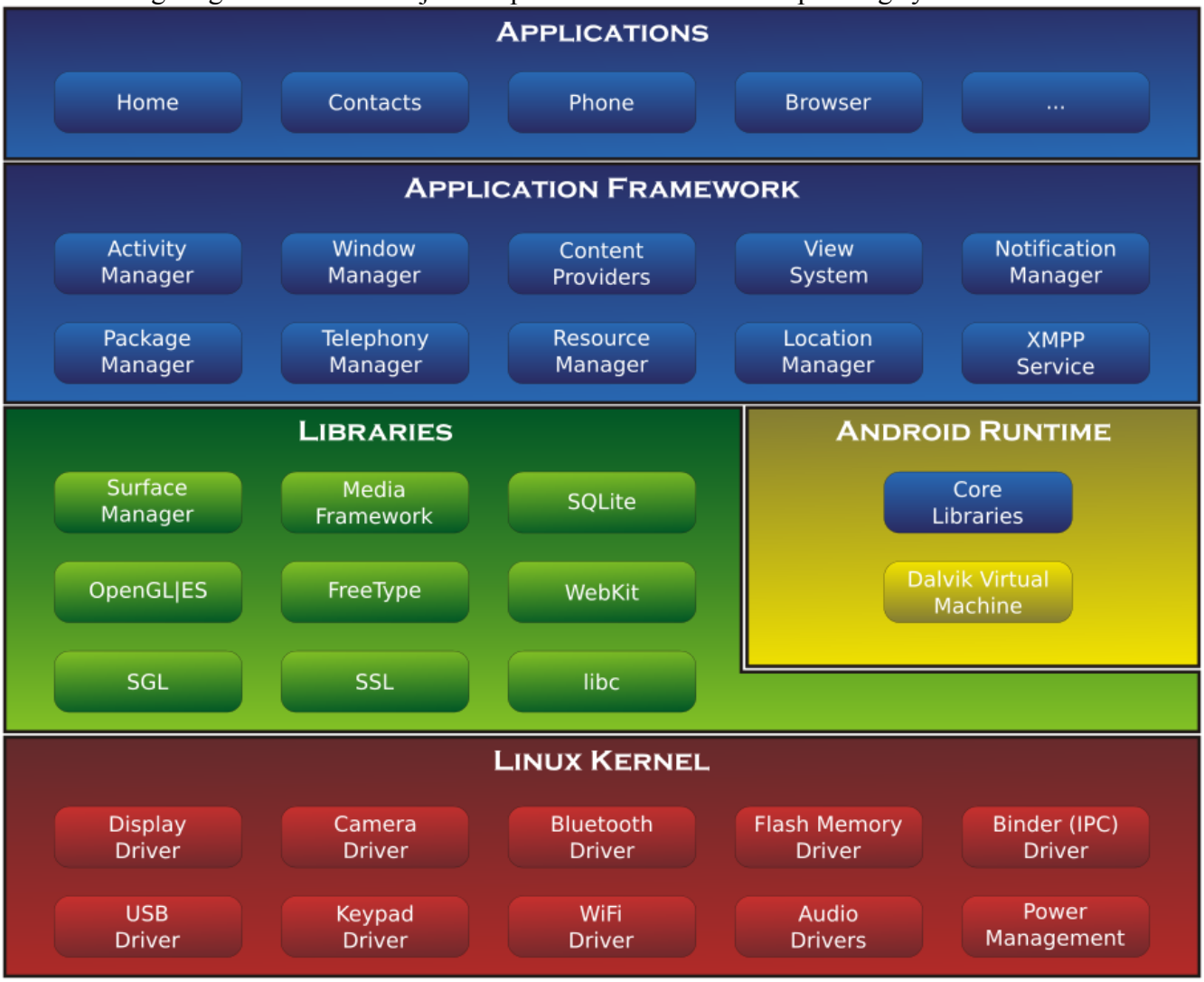
Android

What is Android?

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

The following diagram show the major components of the Android operating system:



How do we get started?

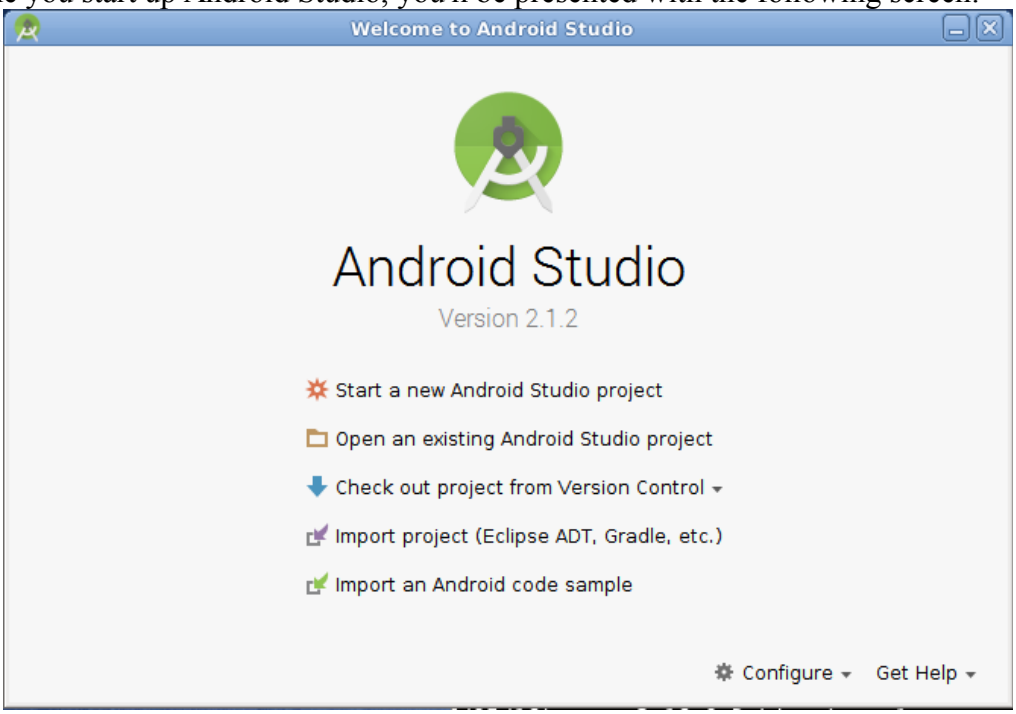
The first thing you need to do is to download the Android Studio IDE by going to:

https://developer.android.com/studio/index.html

Android Studio is available for Windows, Mac and Linux, once you download the installer, run it and follow the setup wizard to install Android Studio and the necessary SDK tools. When you first start out, it's recommended that you simply select all of the default options and let it install and setup everything for you.

Getting started with Android Studio:

The first time you start up Android Studio, you'll be presented with the following screen:



It's recommended that you first read the Android Studio User Guide which will explain the basic workflow and layout of the Android Studio IDE. There are a large number of step-by-step tutorials online that will walk you through the creation of some basic Android apps to get you started. Now is also a good time to start reading up on Java, since all of the Android programming you'll be doing will be in Java:

- Android Studio User Guide: https://developer.android.com/studio/intro/index.html
  Writing your first app, sample code, UI and layout editor, building and running apps, etc...

- Android Developer Training: https://developer.android.com/training/index.html
  Google's training classes and guides for getting started with app development.

- A few other online resources for Android tutorials and sample projects:
  - Tutorials Point: https://www.tutorialspoint.com/android/
  - Vogella: http://www.vogella.com/tutorials/android.html
  - Tutorials for Beginners: https://www.sitepoint.com/12-android-tutorials-beginners/
  - Javapoint: https://www.javatpoint.com/android-tutorial
  - Java tutorial: http://www.androidauthority.com/java-tutorial-beginners-582147/

Creating your first app:

Once you've opened Android Studio, you'll be sitting at the Welcome screen, click on **Start a new Android Studio project**, and enter the following information while clicking **Next** on each screen:

Configure your new project:

Application Name: **My First Application**

Company Domain: **apps.cool**

Click **Next.**

Select the form factors your app will run on:

[X] Phone and Tablet

Minimum SDK: **API 15**

We are creating a simple app at this point, but if you wanted to also created an Android Wear, TV, Auto or Glass app, you could enable those platforms as well on this screen. Click **Next**.

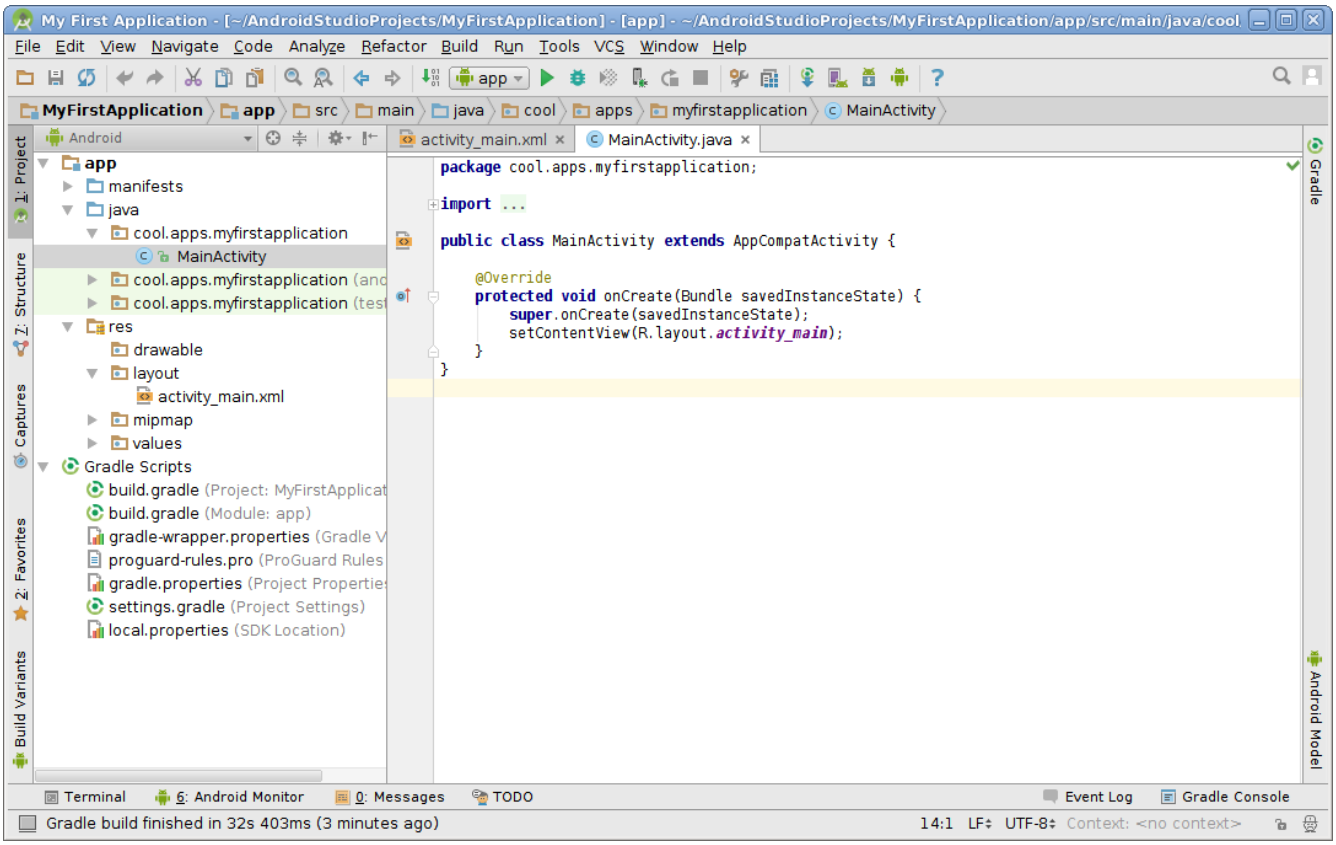
Add an Activity to Mobile:

Select **Empty Activity** and click **Next**.

Customize the Activity:

Leave the default options and click **Finish**.

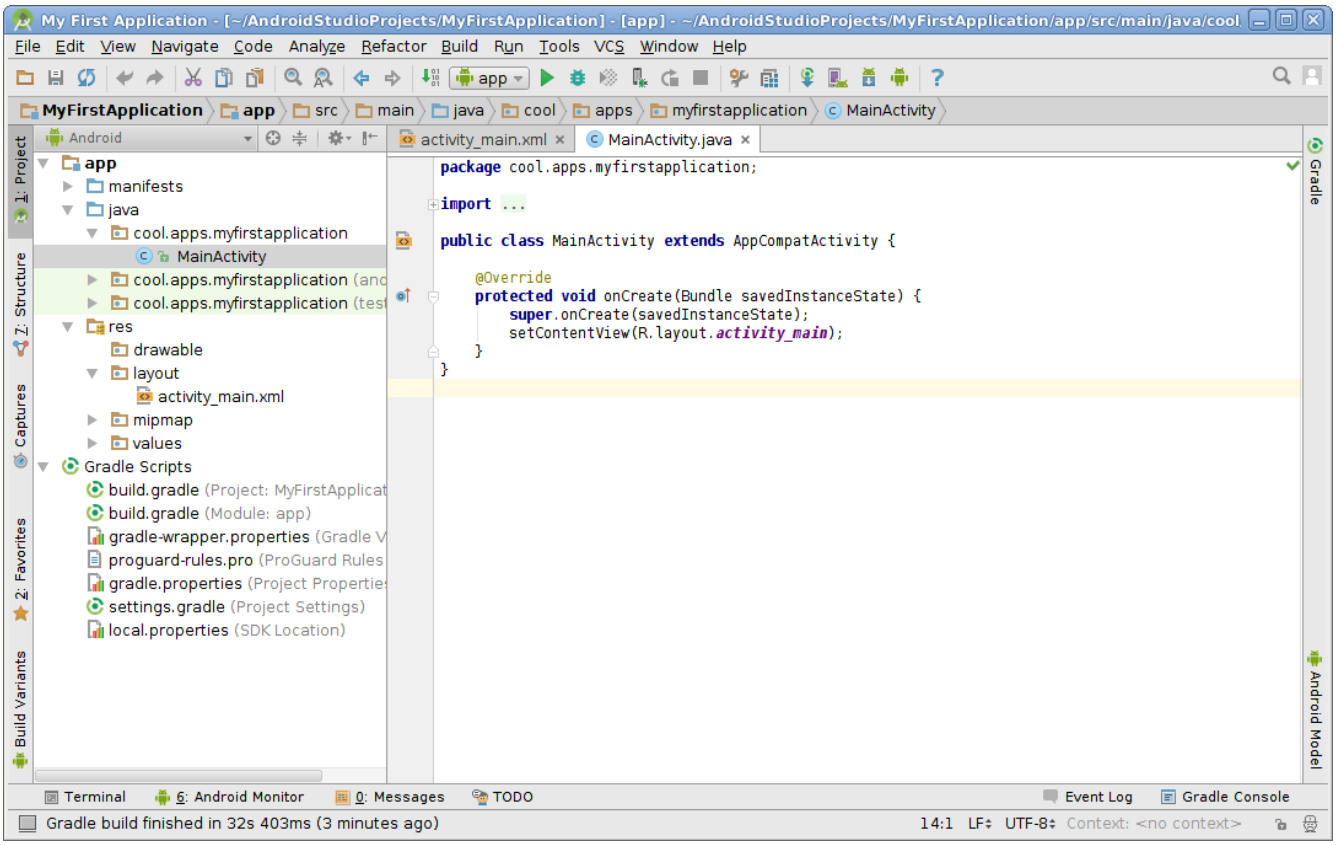
Android Studio will now create the necessary project structure and you will now be sitting at the main screen in Android Studio:



The two main areas that we want to focus on, are under:

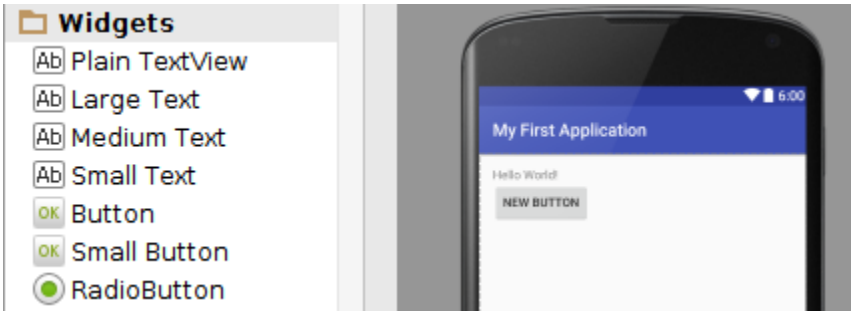
app > java > cool.apps.myfirstapplication > MainActivity.java

app > res > layout > activity_main.xml

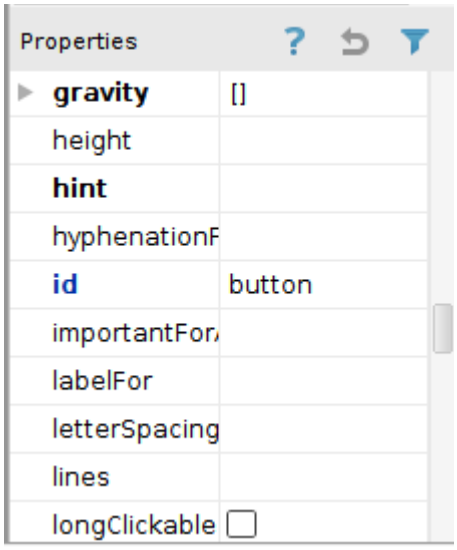


The *activity_main.xml* is the layout file for the main activity of your Android app. You can think of an "activity" as being a graphic window where you create your user interface. The *MainActivity.java* is the file that will contain your source code to handle events that occur on the activity. So let's first look at the *activity_main.xml* file, when you open it, you'll see two tabs at the bottom, one labeled **Design** and the other **Text**. The **Text** view allows you to see the actual text that makes up the XML file for the layout view, however since we're just starting out, I'd recommend you stick with the **Design** view which will allow you to graphically manipulate the widgets (buttons, text boxes, images, etc) in the layout.

By default you should see just a *TextView* widget in your application that has the text "Hello World", we want to also add a *Button*, so drag a *Button* from the **Widgets** area and place it under the *TextView* that is already on your activity, so that it looks like this:



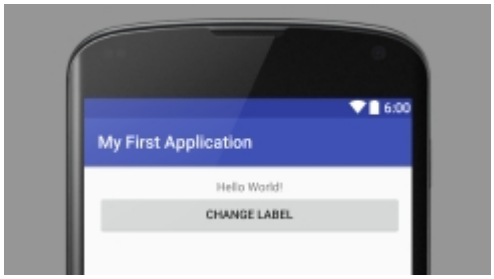
You can modify the properties of each widget, but first selecting the widget on the screen, and then in the **Properties** area, scrolling through the list of properties until you find what you are looking for:



The two main properties that you need to be aware of are the *id* and *text* properties. The *id* property is basically the variable name for the widget which we'll use to refer to that widget in our code. For our *TextView* widget, it's *id* is "textView" and for our *Button* it's *id* is "button". If we wanted to, we could easily change those names to something else, but for now we'll leave them as their default names. The *text* property allows you to change the text that is displayed on the widget. For our *TextView* it's *text* property is set to "Hello World!", and our *Button* has it's *text* property set to "New Button". We want to change that to say "Change Label". You'll notice that the text in the button on the screen is in all capital letters, this is by design, but if you really don't like it, you can modify the button's TransformationMethod (which we won't talk about here), so that it doesn't automatically capitalize the letters on the button. The next thing we want to do is to change the layout of the widgets so that they fill out the entire width of the activity, so on both the *TextView* and the *Button*, in their properties, set:

layout:width = fill_parent
gravity = center

Your activity's layout should now look like this:



Now we can start coding our activity, so switch over to the *MainActivity.java*. The first thing we need to do is to create two variables (lines 7, 8) that we'll use to reference (lines 15, 16) the two graphical widgets that are on our activity:

activity_main.xml × MainActivity.java ×

```
1  package cool.apps.myfirstapplication;
2
3  import ...
5
6  public class MainActivity extends AppCompatActivity {
7      private TextView textView;
8      private Button button;
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_main);
14         //Get references to our widgets
15         textView = (TextView)findViewById(R.id.textView);
16         button = (Button)findViewById(R.id.button);
17
18     }
19 }
20
```

One thing you should notice at this point is that the Android Studio has highlighted the *TextView* and *Button* data types in red indicating that it doesn't know what they are. The reason why Android Studio doesn't know what they are is because we haven't imported the proper classes into our code yet. This can easily be fixed by placing your mouse first over top of the *TextView* and pressing Alt-Enter on your keyboard, and then doing the same thing for the *Button*, this will cause Android Studio to pull in the two class files (android.widget.TextView and android.widget.Button). You will notice that they are no longer highlighted in red:

```
    private TextView textView;
    private Button button;
```

Now we need to create a function that will get called when our button is clicked. Add the following *onButtonClick* function to your code:

activity_main.xml × MainActivity.java ×

```
1  package cool.apps.myfirstapplication;
2
3  import ...
8
9  public class MainActivity extends AppCompatActivity {
10     private TextView textView;
11     private Button button;
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_main);
17         //Get references to our widgets
18         textView = (TextView)findViewById(R.id.textView);
19         button = (Button)findViewById(R.id.button);
20     }
21
22     public void onButtonClick(View v) {
23         textView.setText("Thank you!");
24         button.setText("This Button has been clicked");
25     }
26 }
```

When you add that function, *View* will be highlighted red, you'll have to click on it, and then press Alt-Enter in order to get the correct class imported.

Now we need to connect that *onButtonClick* function to our actual button, so we need to switch back over to the *activity_main.xml* and select the *Button* and in the properties area, scroll down to the *onClick* property and click the little down arrow which will bring up a list of function names, select the *onButtonClick* function. Now anytime someone clicks that button, the *onButtonClick* function will get triggered.

We are done coding, now let's see how we run our app in an emulator...

Setting up an emulator:

The Android Emulator allows you to create a virtual Android device on your development computer. This allows you to immediately test your Android apps on your computer without having to use a physical Android device.

In Android Studio, go to the **Tools → Android → AVD Manager** and click the "+ Create Virtual Device..." button. You can choose whichever hardware device you like and then click **Next**. Choose whichever system image you want for your Android virtual device and click **Next**. You can then give your Android virtual device a name and change any of it's settings if you like, and then click **Finish**.

You can now click the little green arrow play button to start up your AVD, at which point you'll then see an emulator window open showing you an Android device booting up and then you'll be presented with the main Android desktop.

From here you can navigate the Android virtual device just like it was a real Android device.

Running your app in the emulator:

Now that we have an emulator running, we can now run our app that we just created inside of our emulator. Back in Android Studio, click the green arrow button at the top center of the screen:

You will then be presented with a dialog box asking you which target device you'd like to use. If you have any physical Android devices connected and if they are in “USB debugging” mode, then you can either select one of those devices, or any of your virtual devices from the list and simply click “OK”. Your app will be installed on the device and will immediately open. You can now make any changes to the code that you want in Android Studio, and simply click the green arrow button to install the updated version of your app on your Android device.

That's it...I highly recommend looking at the URLs on the second page of this document for more tutorials and sample projects.