# How to Install Android Studio IDE & SDK and Get Started with Android Programming

TABLE OF CONTENTS (HIDE)

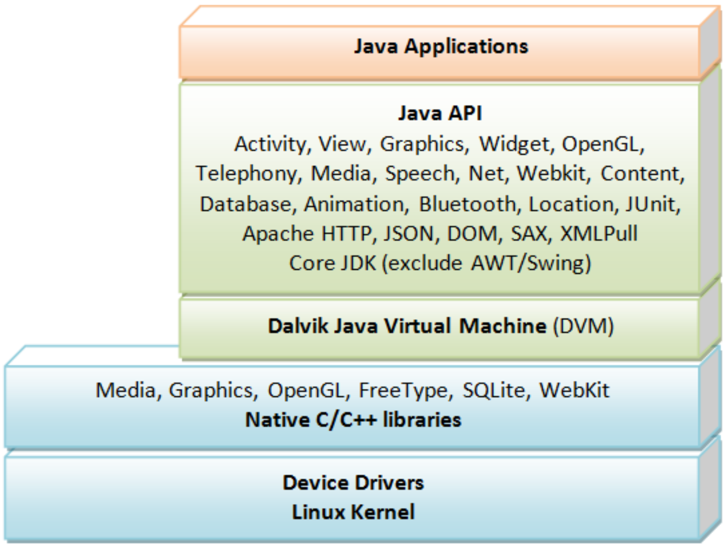
## 1. Introduction

Android is an *Operating System for mobile devices* developed by Google in 2007, which is built upon a Linux kernel. Android competes with Apple's iOS (for iPhone/iPad), Microsoft's Windows Phone, and many other proprietary mobile OSes.

The latest Android supports Phone/Tablet, TV, Wear (watch and glass), Automobile and Internet of things (IoT).

### Android Platform

Android is based on Linux with a set of native core C/C++ libraries. Android applications are written in Java. However, they run on Android's own Java Virtual Machine, called Dalvik Virtual Machine (DVM) (instead of JDK's JVM) which is optimized to operate on the small and mobile devices.


Java Applications
Java API
Activity, View, Graphics, Widget, OpenGL, Telephony, Media, Speech, Net, Webkit, Content, Database, Animation, Bluetooth, Location, JUnit, Apache HTTP, JSON, DOM, SAX, XMLPull
Core JDK (exclude AWT/Swing)
Dalvik Java Virtual Machine (DVM)
Media, Graphics, OpenGL, FreeType, SQLite, WebKit
Native C/C++ libraries
Device Drivers
Linux Kernel

In May 2017, Google announced support for a new Kotlin programming language. As you are familiar with Java, you probably should start in Java (many of the examples out there are written in Java), and then move into Kotlin. Kotlin will not be discussed in this article.

The mother site for Android is https://www.android.com. For programmers and developers, visit https://developer.android.com to download the SDK, Android Training, API Guides and API documentation.

## 2. Installing "Android Studio IDE" and "Android SDK"

Installing Android software is probably the most challenging part of this project. It takes times - from 30 minutes to *n* hours to forever - depending on your luck, your programming knowledge, and your PC. You probably need a fairly decent PC (with 8GB RAM) and 10GB of free disk space to run the Android emulator!!! Running on "actual" Android phone/tablet requires much lesser resources.

### Step 0: Pre-Installation Check List

1. Before installing Android SDK, you need to install Java Development Kit (JDK). Read "How to install JDK". Ensure that your JDK is at or above 1.8. You can check your JDK version with command "`javac -version`" (compiler) and "`java -version`"(runtime).
2. Uninstall older version(s) of "Android Studio" and "Android SDK", if any.
3. The installation and many operations take a LONG time to complete. Do NOT stare at your screen or at the ceiling. Browse through the "Android for Developers"

@ https://developer.android.com.

4. We need to install two HUGE packages:
    a. Android Studio (IDE) (about 1.6 GB), which is an Integrated Development Environment (IDE) based on IntelliJ (a popular Java IDE); and
    b. Android SDK (Software Development Kit) (about 5 GB) for developing and running Android apps.

### Step 1: Install "Android Studio IDE"

**Reference**: "Install Android Studio" @ https://developer.android.com/studio/install.

#### (For Windows)

1. Check that environment variable `JAVA_HOME` is set to the JDK installation directory via command "`set JAVA_HOME`". Otherwise, Follow the steps HERE.
2. Check the system requirements for Android Studio/SDK @ https://developer.android.com/studio. Search for "System Requirements", e.g., For Windows 10, 8GB of RAM, 8GB of disk space, and 1280x800 minimum screen resolution. Take note that you should have enough space on C drive. Insufficient space on C drive will take you many days to install.
3. Goto "Android Studio" under "Android Developers" @ https://developer.android.com/studio ⇒ Click "Download Android Studio" (Android Studio Electric Eel | 2022.1.1 for Windows 64-bit (988.6MB)) to download the executable installer "`android-studio-2022.1.1.21-windows.exe`".
4. Run the downloaded installer ⇒ You may watch a short video @ https://developer.android.com/studio/install.
    a. In "Choose Components", select "Android Studio" and "Android Virtual Device" (i.e., Android Emulator) (space required: 3.2GB).
    b. In "Configuration Settings Install Location", accept the default "`C:\Program Files\Android\Android Studio`".
    c. In "Choose Start Menu Folder", accept the default ⇒ Install.
    d. Start Android Studio ⇒ Finish. Continue to Step 2.

by default, the "Android Studio IDE" will be installed in "`C:\Program Files\Android\Android Studio`", and the "Android SDK" in "`c:\Users\`*`username`*`\AppData\Local\Android\Sdk`".

**Notes (SKIP)**: You can also use the ZIP version: Download the Windows 64-bit ZIP version (about 1.5GB) ⇒ UNZIP into a folder of your choice ⇒ Run "`bin\studio64.exe`" to launch the Android Studio ⇒ It will enter the "setup" for the first launch ⇒ "Do not Import Settings" ⇒ In "Welcome", click "Next" ⇒ In "Install Type", choose "Custom" (so that you can see what is going on) ⇒ In "Select Default JDK Location", use default ⇒ In "Select UI Theme", choose one that you like ⇒ In "SDK Components Setup", select "Android Virtual Device (1.05GB) ⇒ Take note of the "Android SDK Location" with default of "`C:\Users\username\AppData\Local\Android\Sdk`" ⇒ In "Emulator Settings", use default ⇒ In "Verify Settings", check the settings and choose "Finish" ⇒ In "Download Components", click "Details" and check that nothing fails ⇒ Wait ⇒ Wait ⇒ Wait.

#### (For macOS)

1. Check the system requirements @ https://developer.android.com/studio#Requirements e.g., macOS 10.14 or higher, 8GB of RAM, 8GB of disk space, and 1280x800 minimum screen resolution.
2. Goto "Android Developer" under "Android Developers" @ https://developer.android.com/index.html ⇒ Click "Download Options" ⇒ For **Intel** processor, choose "Mac (64-bit) Android-studio-2021.x.x.xx-mac.dmg (928MiB)"; for **Apple M1 ARM** processor, choose "Mac (64-bit, ARM) Android-studio-2021.x.x.x-mac_arm.dmg (925MiB)".
3. Launch the downloaded "`.dmg`" installation file ⇒ You may watch a short video @ https://developer.android.com/studio/install.
4. Drag and drop Android Studio into the "Applications" folder. Continue to Step 2.

The "Android SDK" will be installed in "`~/Library/Android/sdk`", where ~ denotes your home directory.

Note: If you see a warning that says "the package is damaged and should be moved to the trash", goto "System Preferences" ⇒ Security & Privacy ⇒ under "Allow applications downloaded from" ⇒ select "Anywhere". Then run again.

### Step 2: Installing Android SDK

This step takes a long time as you need to download about 3GB of zip data, and expand to 5 GB of disk data, even for the minimum configuration.

Note: You can actually copy the SDK from another computer with the same OS.

#### (For Windows and macOS)

1. Launch Android Studio ⇒ It will run the "setup" wizard for the first launch.
    a. Choose "do not import previous settings".
    b. In "Welcome", choose "next".
    c. In "Install Type", choose "Standard" (default).
    d. In "Select UI Theme", use the default or choose one that you like.
    e. In "Verify Settings", take note of the SDK directory (by default @ `c:\Users\`*`username`*`\AppData\Local\Android\Sdk` for Windows, "`~/Library/Android/sdk`" for macOS):

```
Setup Type: Standard
SDK Folder: C:\Users\xxxxxxxx\AppData\Local\Android\Sdk
JDK Location: C:\Program Files\Android\Android Studio\jbr (Note: Gradle may be using JAVA_HOME when invoked from command line. M
Total Download Size: 443 MB

SDK Components to Download:
Android Emulator                          318 MB
Android SDK Build-Tools 33.0.2           53.1 MB
Android SDK Platform 33-ext5                      64.5 MB
Android SDK Platform-Tools                        5.85 MB
Intel x86 Emulator Accelerator (HAXM installer)    500 KB
SDK Patch Applier v4                              1.74 MB
```

f. In "License Agreement", Accept ALL licenses (You need to choose the licenses from the left panel one by one) ⇒ Finish ⇒ Wait ⇒ Wait ⇒ Wait ⇒ Wait ⇒ Wait ⇒ Wait.

2. (For Windows) Use "File Explorer" to check the SDK installed directory. Take note that the "`AppData`" is a hidden directory. You need to choose "View" ⇒ Uncheck "Hidden Items" to see this directory.
   (For macOS) Use "Finder" to check the SDK installed directory.
3. You can also use "Android Studio" to check the SDK packages installed by selecting "More Actions" (or "Configure" in older versions, or "Tools") ⇒ "SDK Manager" ⇒ "Android SDK" (sidebar):
   - Under "SDK Platforms" tab:
     - Android API 33
   - Under "SDK Tools" tab:
     - Android SDK Build Tools 34-rcx
     - Android Emulator
     - Android SDK Platform-Tools (34.x.y)
     - Intel x86 Emulator Accelerator (HAXM installer)

   Note: If some of these items (such as Android SDK Platform) were not installed, don't worry, it will be installed automatically when you start a new project.

## 3. Write your First Android App

Android apps are written in Java (or Kotin), and use XML extensively. I shall assume that you have basic knowledge of Java and XML.

Take note that Android emulator is slow - VERY VERY VERY SLOW!!! Be Patient!!! Using a physical phone is much faster.

### 3.1 Hello-World

#### Step 0: Read

Goto "Android Guides" @ https://developer.android.com/guide/index.html. Read "Build your first app".

#### Step 1: Create a New Android Project

1. Launch "Android Studio".
2. Choose "New Project".
3. Under "Template" ⇒ select "Phone and Tablet" tab ⇒ select "Empty Activity" (NOT the first item "No Activity") ⇒ Next.
4. In "Create a new empty activity" ⇒ Set "Name" to "Hello Android" (this will be the "Title" in your phone's app menu) ⇒ Take note that "Package name" and "Save Location" will be updated automatically ⇒ In "Language", select "Java" ⇒ Leave the "Minimum API Level" and the rest to default ⇒ Finish.
5. Be patient! It could take a few minutes to set up your first app (It will also install the missing SDK if any). Watch the "progress bar" at the bottom status bar and Zzzzzzzzz...... If you see "2 processes running", you can click on it to see the details. Once the progress bar indicates completion, **a working hello-world app is created by default**. (It took more than 5 minutes to complete!!!)

#### Step 2: Setup Emulator (aka Android Virtual Device or AVD)

**Reference**: "Create and manage virtual devices" @ https://developer.android.com/studio/run/managing-avds

Note: The older versions of Android Studio called the Android Emulator "Android Virtual Device" or "AVD".

To run your Android app under an emulator, you need to first create an Android Virtual Devices (AVD). An AVD emulates a specific mobile device (e.g., your zPhone 88 or Taimi Green).

1. In Android studio, select menu "Tools" ⇒ "Device Manager" (or "AVD Manager" in earlier versions).
2. Under "Virtual", click "Create Device".
3. In "Select Hardware" "Choose a device definition" ⇒ In "Category", choose "Phone" ⇒ In "Name", choose "Pixel XL" (or any phone that you like) ⇒ Next.
4. In "System Image" "Select a System Image" ⇒ Choose the latest API 33 "Tiramisu" (or one that is already installed), click the "Download" icon ⇒ Finish, and wait...
5. In "Android Virtual Device (AVD)" "Verify Configuration" ⇒ Finish.
6. If you see "VT-x is disabled in BIOS": Check your BIOS setting to ensure that "Virtualization Technology" is enabled. Shutdown and re-boot your PC to enter the BIOS setup. This is machine dependent. Google "*Your-PC-brand-and-model* enter BIOS setup". For example, for my HP computer ⇒ Boot ⇒ "ESC" to enter BIOS setup ⇒ Advanced ⇒ System Options ⇒ Check "Virtualization Technology (VTx)" ⇒ Save ⇒ Exit.

#### Step 3: Run the Android App on Emulator

1. In Android Studio, select the "Run" menu ⇒ "Run app" ⇒ Wait.... (In older versions, you may need to go "Available Virtual Devices" and select the AVD created earlier).
2. You MAY BE prompted to install Intel HAXM (Hardware Accelerated Execution Manager). Follow the instruction to install HAXM.
3. Be patient! It may take a few MINUTES to fire up the app on the emulator. The status shows "Gradle build running". Once build completes, the emulator will be

launched. You first see a blank screen ⇒ wait ⇒ Google logo ⇒ wait ⇒ G logo ⇒ wait and wait and wait ⇒ "Android" ⇒ Home screen ⇒ "Hello, world!" message. **If you have problem running on the emulator, I suggest you try to run on an actual Android Phone**. Goto next step.

4. In the "Emulator" pane, select "Settings" ⇒ "View Mode" ⇒ "Float".
5. Push the "HOME" button of the emulator ⇒ In the app menu, search for "Hello Android" app that we have just installed.
6. **DO NOT CLOSE THE EMULATOR**, as it really takes a long time to start. You could always re-run the app (or run a new app) on the SAME emulator. Try re-run the "Hello Android" app by selecting "Run" menu ⇒ "Run app".

**Common Errors**:

1. Google the "error message"!
2. If everything fails, select "File" menu ⇒ "Invalidate Caches / Restart..." ⇒ and wait ...
3. Cannot find "**AVD manager**" in the "**Tool**" menu: You have missing packages. You should have a window with Gradle alert with a link which you can click and you will see a window with a prompt to download missing packages. When all stuff downloaded the "ADV manager" should be enabled.
4. If you get an error message "**Fail to find target with hash string 'android-26'**". Either click the link "Install missing platform(s) and sync project" to install API-26 (another GB download!); or Under "Gradle Scripts" ⇒ Open "build.gradle (Module: app)" ⇒ Change "`compileSdkVersion`" and "`targetSdkVersion`" from 26 to 27 (we have installed API-27) and "`com.android.support:appcompat-v7:26.x.x`" to "`27.0.0`".
5. If you get an error message "Re-download dependencies and sync project (requires network)", click the link to download. If the message appears again:
   - (Windows) Goto "`C:\Users\`*`username`*`\.gradle\wrapper\dists`" and delete "`gradle-x.x-all`". Take note the "`.gradle`" is a hidden directory and you need to enable viewing of hidden directory.
   - (Macs) Goto "`~\.gradle\wrapper\dists`" and delete "`gradle-x.x-all`". Take note the "`.gradle`" is a hidden directory and you need to enable viewing of hidden directory.

   Restart Android Studio.
   This error is due to poor network condition, resulted in corrupted download.
6. If you encountered error "**Emulator: ERROR: x86 emulation currently requires hardware acceleration**", Read "https://stackoverflow.com/questions/29136173/emulator-error-x86-emulation-currently-requires-hardware-acceleration", (6), and (7).
7. If you encountered error "**HAX is not working...**" ⇒ Open "SDK Manager" ⇒ SDK Tools ⇒ Check if "Intel x86 Emulator Accelerator (HAXM Installer)" is installed ⇒ Goto SDK Location (by default, "`C:\Users\`*`your-username`*`\AppData\Local\Android\Sdk\extras\intel\Hardware_Accelerated_Execution_Manager`" for Windows or "`~/Library/Android/sdk/extras/intel/Hardware_Accelerated_Execution_Manager`" for macOS) ⇒ run "`intelhaxm-android.exe`" to install HAXM ⇒ Be patient! The installer may take a while to launch ⇒ Follow the screen instructions to complete the setup.
   Take note that: (a) In Windows, the "`AppData`" directory is hidden. You need to unhide via "File Explorer" ⇒ "View" menu ⇒ Uncheck "Hidden Items. (b) In macOS, the "`Library`" folder is hidden. You can unhide via "Finder" ⇒ Go ⇒ Home ⇒ Settings ⇒ Show View Option.
   If the problem persists, remove and then re-install.
8. If you encountered error "**Intel virtualization technology (VT-x) is not turned on**" ⇒ Check your BIOS setting to ensure that "Intel virtualization technology" is enabled. Shutdown and re-boot your PC to enter the BIOS setup. This is machine dependent. Google "*Your-PC-brand-and-model* enter BIOS setup".
   If "Intel virtualization technology" is already enabled, this error is probably caused by your antivirus software. Disable your antivirus for this session and rerun.
   See http://stackoverflow.com/questions/21635504/error-during-installing-haxm-vt-x-not-working.
9. If you have problem creating AVD via "AVD Manager" (On macOS having error "Studio quit unexpectedly"), open the AVD manager via command line as follows:

```
// For macOS
cd ~/Library/Android/sdk/tools
./android avd

// For Windows
cd C:\Users\your-username\AppData\Local\Android\sdk\tools
android avd
```

10. If the emulator flashes and disappears, with error message "error while loading state for instance 0x0 of device 'goldfish_pipe'", installed a newer version of HAXM @ "C:\Users\*username*\AppData\Local\Android\Sdk\extras\intel\Hardware_Accelerated_Execution_Manager\intelhaxm-android.exe".

## Step 4: Run the App on Real Android Phones

**Reference**: "Run on Real Device" @ https://developer.android.com/training/basics/firstapp/running-app.html#RealDevice.

To run the Android app on a REAL device (Android Phone or Tablet):

1. Connect the real device to your computer. Make sure that you have the "USB Driver" for your device installed on your computer. If not, goto https://developer.android.com/studio/run/oem-usb.html to Install OEM USB Drivers. If you device is not certified there, good luck! It took me many hours to find a compatible driver for my cheap un-brand Tablet.
2. Enable "USB Debugging" mode on your *real device*:
   (On Android 4.2/5.0 and newer) Need to enable "Developer options" via "Settings" ⇒ About Phone ⇒ Software Information (if there is one) ⇒ Tap "Build number" seven (7) times until "Developer Mode" is displayed. Return to the "Settings" menu to find "Developer options" ⇒ Open "Developer options" ⇒ Enable "USB debugging".
   (On Android 4.0) From "Settings" ⇒ "Developer options" ⇒ Check "USB Debugging".
   (On Android 3.2 and older) From "Settings" ⇒ "Applications" ⇒ "Development" ⇒ Check "USB Debugging".
   This allows Android SDK to transfer data between your computer and your device. Also enable "Unknown source" from "Applications". This allows applications from unknown sources to be installed on the device.
3. You shall see the message "USB Debugging Connected" (or "Allow USB debugging") when you plugs the USB cable into your computer.
4. You can find your device under "Tools" ⇒ "Device Manager" ⇒ "Physical". (There is a new option called "Pair with wifi in Andriod Studio 2021. To check!)

5. From Android Studio, select "Run" menu ⇒ "Run app" ⇒ Your device shall be listed under "Choose a running device" ⇒ Select the device ⇒ OK ⇒ "Hello-world" will appear on your phone.
6. In your phone's app menu, look for the "Hello Android" app that we have just installed.

### Deleting a Project

To delete a project, select "File" ⇒ "Close Project" ⇒ On the "Recent Projects" ⇒ Hover over the project ⇒ Press "Delete" key on the project to remove the project from Android Studio ⇒ You can then delete the project directory from the file system.

## 3.2 Hello-world "by Coding"

There are two ways to create User Interface (UI) on Android:

1. Write Java codes.
2. Layout via XML descriptions and let the system generates the Java code for you.

Let's begin with writing Java codes (because I have taught you Java). We shall continue from the "Hello Android" project created earlier.

### `MainActivity.java`

Expand the "`app`" node (by clicking on the triangle). Expand the "`java`" node. Expand the "`com.example.helloandroid`" package node. Open the "`MainActivity.java`" (which actually has already been opened). REPLACE the `onCreate()` method as follows and add the import statement. Do not touch the rest of the codes, if any.

```
package ......;

import ......;
import android.widget.TextView;   // Add this line

public class MainActivity extends ...... {

    // REPLACE the ENTIRE onCreate() method as follows:
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView textView = new TextView(this);   // Construct a TextView UI component
        textView.setText("Hello, from my Java code!"); // Set the text message for TextView
        setContentView(textView);  // this Activity sets its content to the TextView
    }

    // Do not touch the rest of the codes, if any
    ......
    ......
}
```

Run the application ("Run" ⇒ "Run app"). You shall see the message "Hello, from my Java code!" displayed. This round, you should not have to wait too long!

### Dissecting the "`MainActivity.java`" - Application, Activity & View

An Android application could have one or more `Activity`.

An `Activity`, which usually has a screen, is a single, focused thing that the user can *interact* with the application (hence called *activity*). The `MainActivity` extends the `android.app.Activity` class (or `android.app.AppCompatActivity` in the later version), and overrides the `onCreate()` method. The `onCreate()` is a *call-back* method, which is called back by the Android system when the activity is launched.

A `View` is a UI component (or widget, or control). We construct a `TextView` (which is a subclass `View` for showing a text message), and set its text. We then set the content-view of the `MainActivity` screen to this `TextView`.

### Android Application Descriptor File - "`AndroidManifest.xml`"

Each Android application has a *manifest file* named `AndroidManifest.xml` under "app" ⇒ "manifests". It *describes* the Android app.

For example, our "`Hello Android`" application, with an activity called `MainActivity`, has the following manifest (generated automatically by the Android SDK when the project was built):

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Hello Android"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

- The `<manifest>` element specifies the package name - NOT shown in the latest API 33?!
- The `<manifest>` contains one `<application>` element.
- The `<application>` element specifies the icon, label (the app's title) and theme of this application. It contains one or more `<activity>` elements.
- This application has one activity. The `<activity>` element declares its program name ("`MainActivity`" in current package "`.`"). It may contain `<intent-filter>`.
- The `<intent-filter>` declares that this activity is the entry point (`android.intent.action.MAIN`) of the application. This activity is to be added to the application launcher (`android.intent.category.LAUNCHER`).

## 3.3 Hello-World using "XML Layout"

Instead of writing Java codes to create the user interface (UI) (as in the above example using a `TextView` component). It is more flexible and therefore recommended to layout your UI components via a *descriptive* XML layout file. In this way, you don't need to hardcode the views, and you can easily modify the look and feel of the application by editing the XML markups. The Java codes can therefore focus on the business logic.

Let's rewrite our hello-world to use XML layout.

### Step 1: Create a New Android App

CLOSE the previous project, via "File" ⇒ "Close Project" (Always CLOSE the previous project before starting a new project).

Choose "Start a new Android Studio project" ⇒ "Phone and Tablet" ⇒ "Empty Activity" ⇒ Next ⇒ Set "Name" to "`Hello Android XML`" ⇒ "Finish".

### Step 2: Define the Layout in XML File "`res\layout\activity_main.xml`"

Expand the "app", "res (resource)", "layout" node. Open the "`activity_main.xml`" (which is actually already opened). Android Studio provides 3 views for this XML file: "Design (or Graphical)", "Code (or XML)", or "Split" - selectable by the icons at the top-right corner of the panel.

Since I am training you as a programmer, select the "Code" view and study the codes:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

The XML declares a `TextView` (text field) that holds a text string "`Hello World!`". The `TextView` component has width and height big enough to hold its content ("`wrap_content`").

### Step 3: Defining String References and Values in XML File "`res\values\string.xml`"

Instead of hardcoding the Hello-World string directly inside the `TextView` (as in the above XML file), we shall use a string reference (or variable) for better flexibility.

Expand `res/values` node. Open `strings.xml`, and ADD the line in red:

```
<resources>
    <string name="app_name">Hello Android XML</string>
    <string name="hello">Hello world from XML!</string>
</resources>
```

This "`string.xml`" defines 2 variables/values:

- A string variable "`app_name`" contains the application's name, that you entered when you created the project.
- A string variable "`hello`" contains the value of "`Hello world from XML!`".

Now, modify the "`activity_main.xml`" to use the string variable "`hello`", in the format "`@string/hello`", as follows:

```
    ......
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        ...... />
    ......
```

### Step 4: Program the Activity in "`MainActivity.java`"

Next, check the "`MainActitivy.java`" (under `app/java/com.example.helloandroidxml`), as follows:

```
package ......;

import ......;

public class MainActivity extends ...... {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);  // Use "res\layout\activity_main.xml" to layout the screen.
    }
}
```

The "`MainActivity`" sets its content-view to "`R.layout.activity_main`", which is mapped to the XML layout file "`res\layout\activity_main.xml`" that we have modified earlier.

### Step 5: Run the App

Run the application. You shall see the new string "`Hello, from XML!`" displayed.

## 4. Uninstall Android Studio IDE and Android SDK

Android is huge. You probably want to remove it after your project sessions.

### (For Windows)

- Run "Control Panel" ⇒ "Programs and Features" ⇒ Uninstall "Android Studio"; or remove the ZIP installed files.
- Delete:
  - `C:\Users\`*`username`*`\AppData\Local\Android\Sdk` (this directory contains the SDK and is huge, about 5GB)
  - `C:\Users\`*`username`*`\.android`
  - `C:\Users\`*`username`*`\.AndroidStudioX.X`
  - `C:\Users\`*`username`*`\.gradle`
  - `C:\Users\`*`username`*`\AndroidStudioProjects` (these are the Android projects written by you)

### (For macOS)

[TODO]

## REFERENCES & RESOURCES

1. Android Developer @ http://developer.android.com/index.html.
2. "Android Guides" @ http://developer.android.com/guide/index.html.
3. Android "API Reference" @ https://developer.android.com/reference/packages.html.

Latest version tested: Android 13 API 33 (Tiramisu), Android Studio 2022
Last modified: March, 2023