CREATE YOUR FIRST

ANDROID PROJECT USING KOTLIN

IN 15 MINUTES

antonioleiva.com

Create your first Android project using Kotlin in 15 minutes

WELCOME!

I’m so glad you’ve decided to start developing Android Apps using Kotlin, or at least taking a look.

In case you don’t know about me, I’m Antonio Leiva. I’ve been working as software developer for 9 years, 5 of them focused on Android.

For the last two years, I’ve devoted part of my time to study Kotlin and how this language can simplify the process of developing an Android App.

I find it a really thrilling language that opens up a new world of possibilities. I wish I’d be able to share my passion with you.

WHAT WILL YOU FIND IN THIS GUIDE?

In this guide, you’ll learn to create an Android project in less than 15 minutes (once you do it for the first time, make sure it will be much less) and, as a bonus, I’ll show you a final surprise :)

But let’s start. I already made you waste a couple of minutes from those 15 you saved for me.

The screens and menus correspond to a Mac, but you’ll easily find the equivalent ones in Windows or Linux.

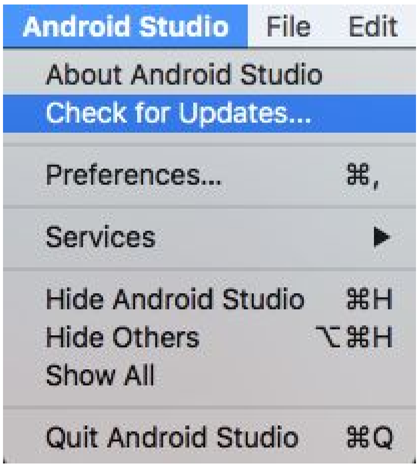
I really hope it helps you, and that it will let you start writing Android Apps in Kotlin from today.

Thanks!
Antonio

Step 1: Install the latest stable version of Android Studio

Es muy posible que esto ya lo tengas. Ten cuidado con las versiones Canary (las que liberan para testeo), porque hay veces que durante unos días Kotlin es inestable en ellas.

If you want to be sure you have the latest one, open Android studio and click *Android Studio --> Check for Updates…*

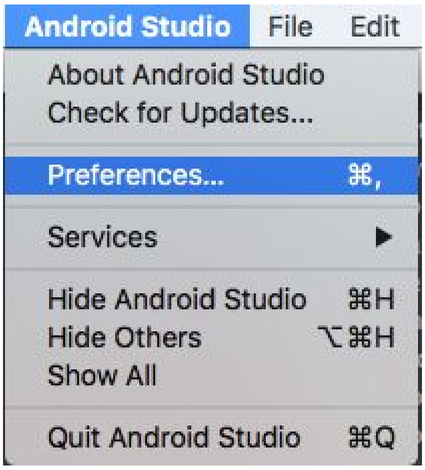


Check whether it shows an Android Studio update. It won’t harm to also update the Android SDK if it’s suggested.

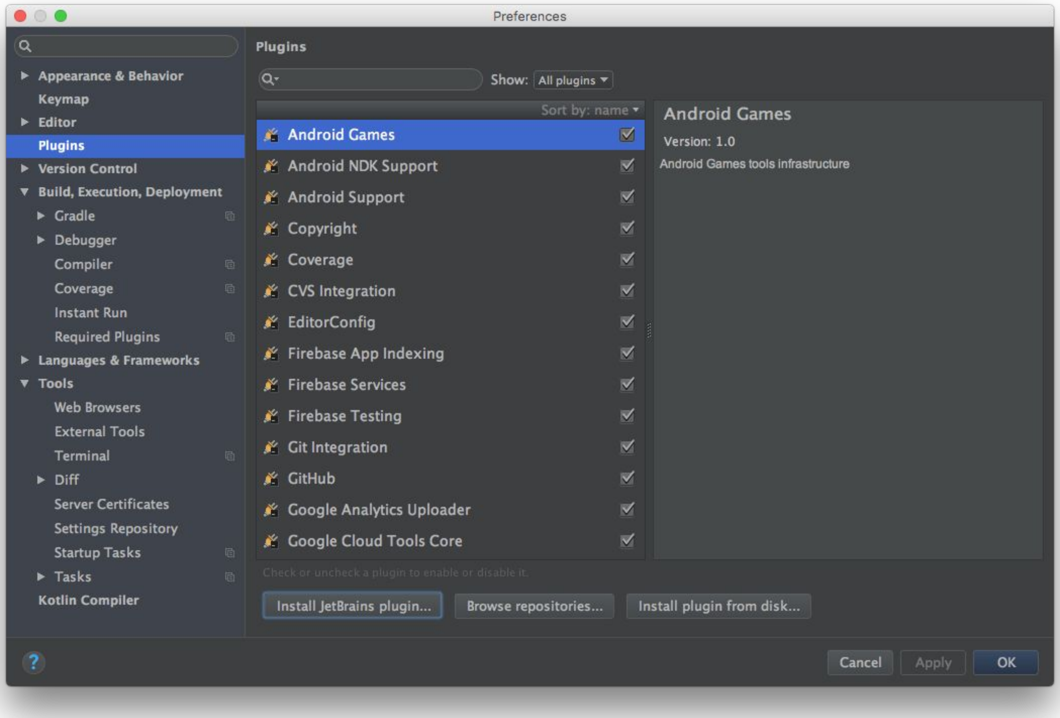
Step 2: Install the Kotlin plugin

Android Studio doesn't include the plugin installed by default, so we need to install it manually. But it's extremely easy.

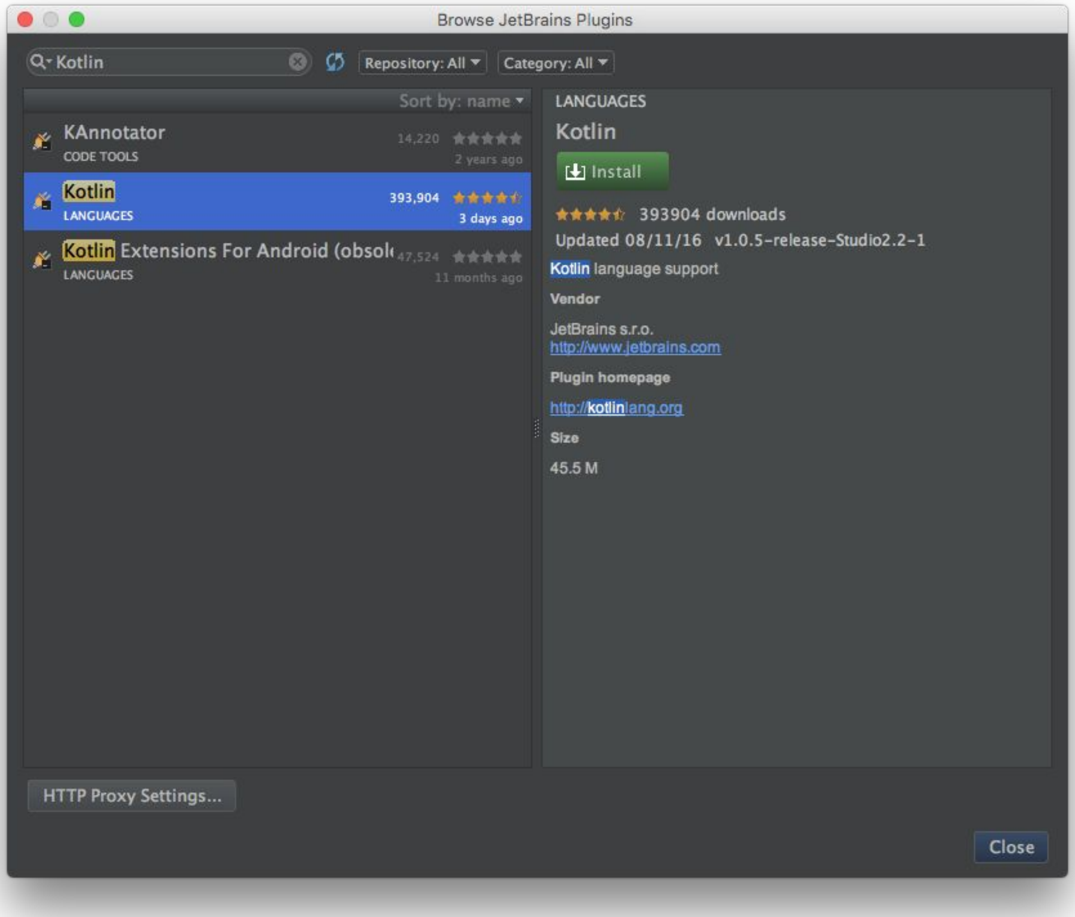
Go to the plugins configuration section in Android Studio. To do it, go to *Android Studio → Preferences…*



Then search *Plugins* in the side menu and click on it. At the bottom of the screen, click on *Install Jetbrains plugin…*



In the next screen, write “Kotlin” in the browser:



KAnnotator
CODE TOOLS
14,220
2 years ago

Kotlin
LANGUAGES
393,904
3 days ago

Kotlin Extensions For Android (obsol
LANGUAGES
47,524
11 months ago

LANGUAGES
Kotlin
393904 downloads
Updated 08/11/16 v1.0.5-release-Studio2.2-1
Kotlin language support
Vendor
JetBrains s.r.o.
http://www.jetbrains.com
Plugin homepage
http://kotlinlang.org
Size
45.5 M

Choose the right plugin (it’s just called “Kotlin”), and click *Install*. It will ask to restart Android Studio.

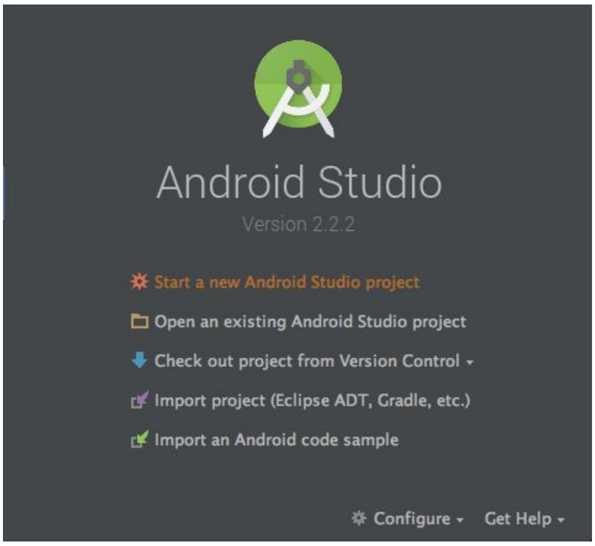
¡Done!

Step 3: Create a new Android Studio project

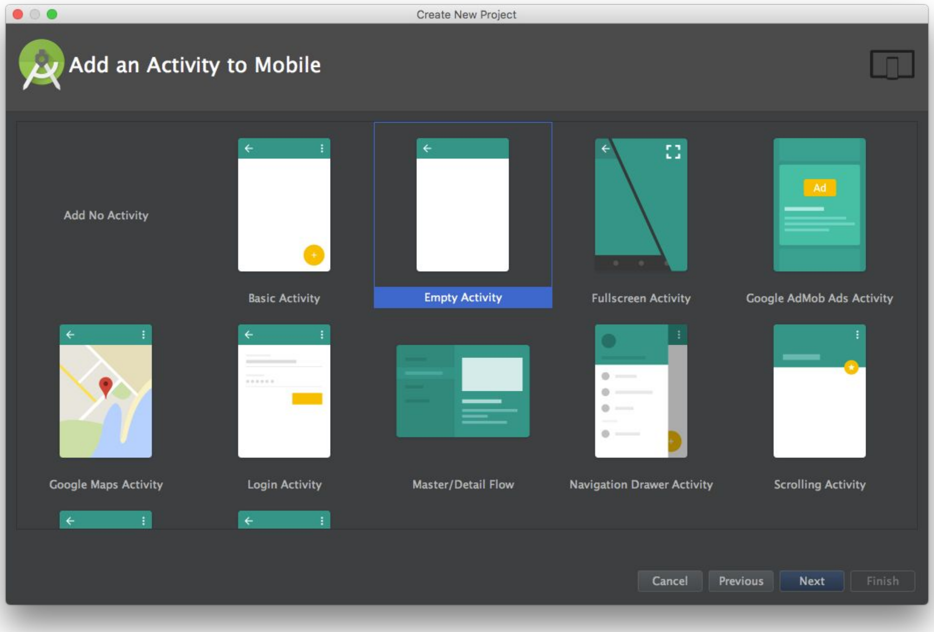
A regular project, it doesn't need anything different. Choose *Start a new Android Studio project*.

Follow the regular process. Choose a name and a path for the App, and press *Next*.

In the screen *Target Android Devices* choose only *Phone and Tablet*, and the minimum version you prefer. I'll be using min-API 15.


Android Studio
Version 2.2.2
Start a new Android Studio project
Open an existing Android Studio project
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Import an Android code sample
Configure
Get Help

In the Activity creation screen, choose *Empty Activity*:



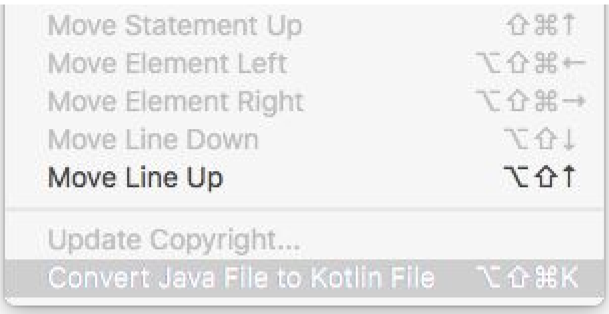
You can use the default values in the last screen of the wizard. Press *Finish*.

Step 4: Convert the MainActivity to Kotlin

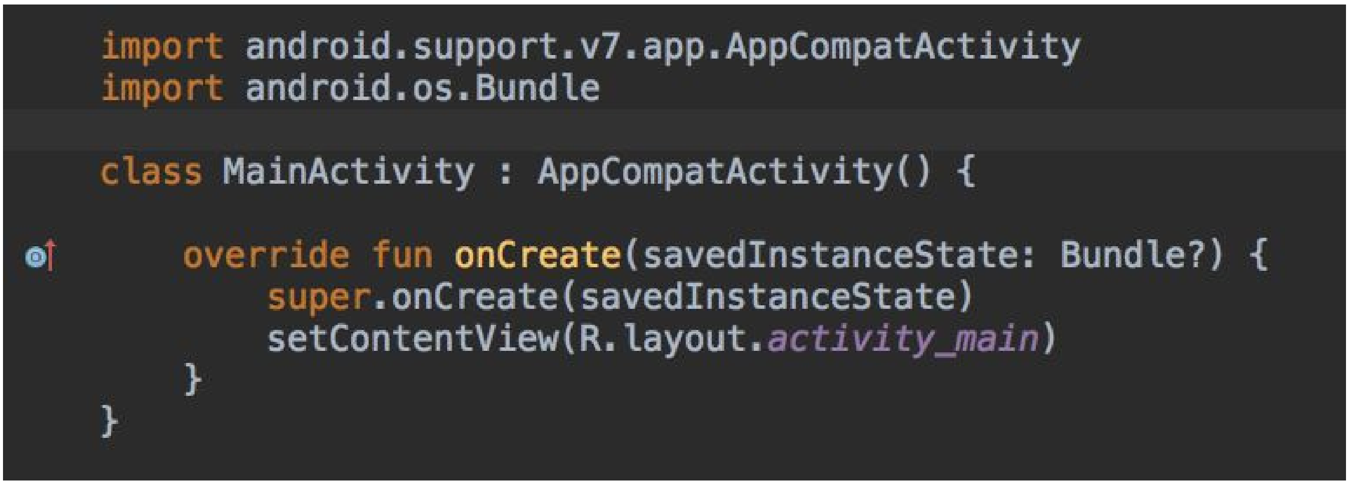
The simplest way to prepare a project to start using Kotlin is having a Kotlin file inside of it.

So let’s convert the *MainActivity*. The Kotlin plugin includes an action to do it for ourselves. Remember it because it will help you during your learning.

Select the *MainActivity.java* file in the left project tree. In the *Code* menu, choose the last option: *Convert Java File to Kotlin File*



After a couple of seconds, this will be the result:



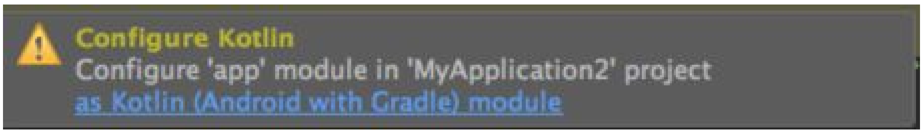
```
import android.support.v7.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

In this guide I’m not covering the differences between languages in deep (I’ll send you some extra things to help you with this).

But you can see that, even having some differences, the code is perfectly understandable if you already know Java.

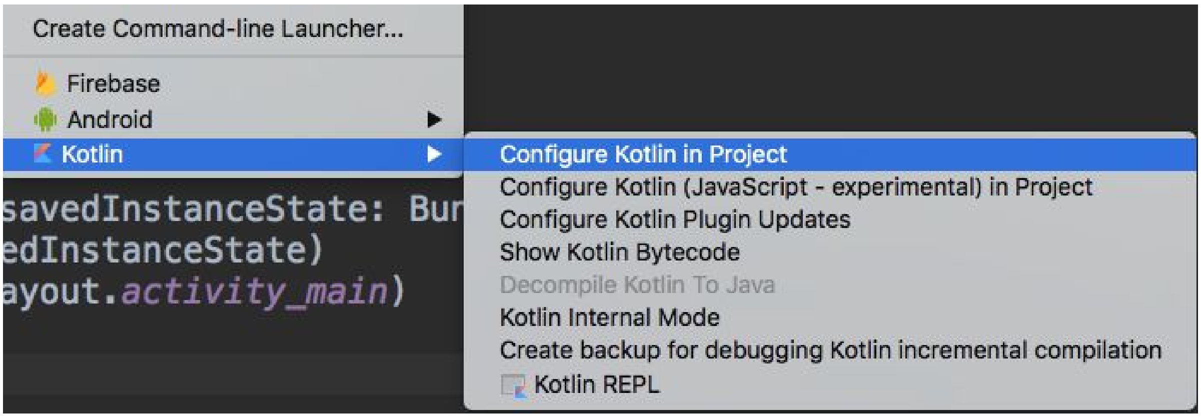
Step 5: Configure Kotlin in your project

It’s possible that you already see an alert like this one:



Configure Kotlin
Configure 'app' module in 'MyApplication2' project
as Kotlin (Android with Gradle) module

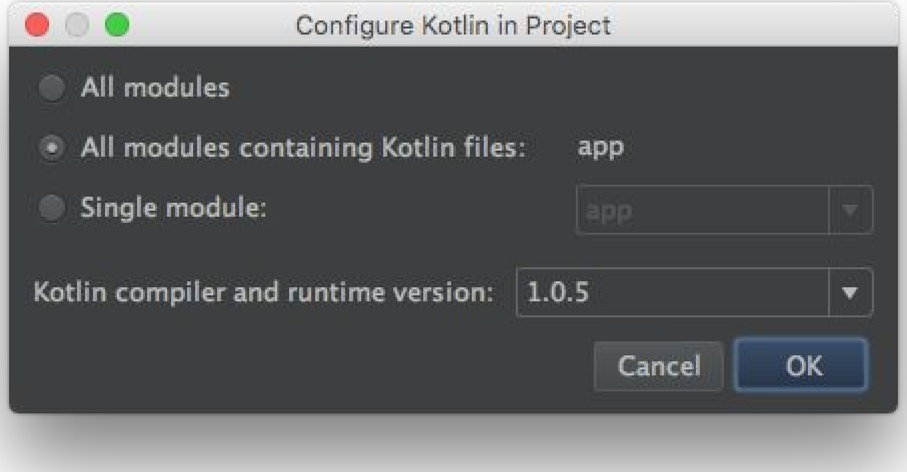
You can click on it. But if not, don’t worry. In the *Tools* menu, you’ll find a submenu called *Kotlin*. Choose *Configure Kotlin in project* there:



Create Command-line Launcher...
Firebase
Android
Kotlin
Configure Kotlin in Project
Configure Kotlin (JavaScript - experimental) in Project
Configure Kotlin Plugin Updates
Show Kotlin Bytecode
Decompile Kotlin To Java
Kotlin Internal Mode
Create backup for debugging Kotlin incremental compilation
Kotlin REPL

```
savedInstanceState: Bur
edInstanceState)
ayout.activity_main)
```

A window like this one will open:



Configure Kotlin in Project

○ All modules

● All modules containing Kotlin files: app

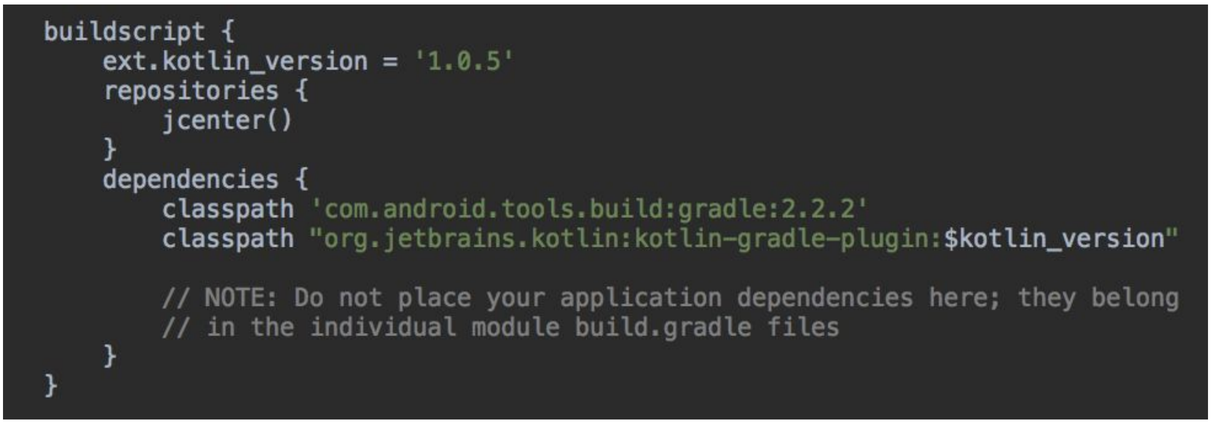
○ Single module: app

Kotlin compiler and runtime version: 1.0.5

The default selection is OK, you don’t need to change anything. Just check that the latest version of the Kotlin runtime is selected. As there’s only one module in the project, the 3 options basically mean the same in our case. Press OK.

This will modify the *build.gradle* in the root folder, and also the one in the *app* module.

If you review the changes, you’ll see that a new dependency was added, to the root *build.gradle*. It corresponds to the Kotlin plugin.



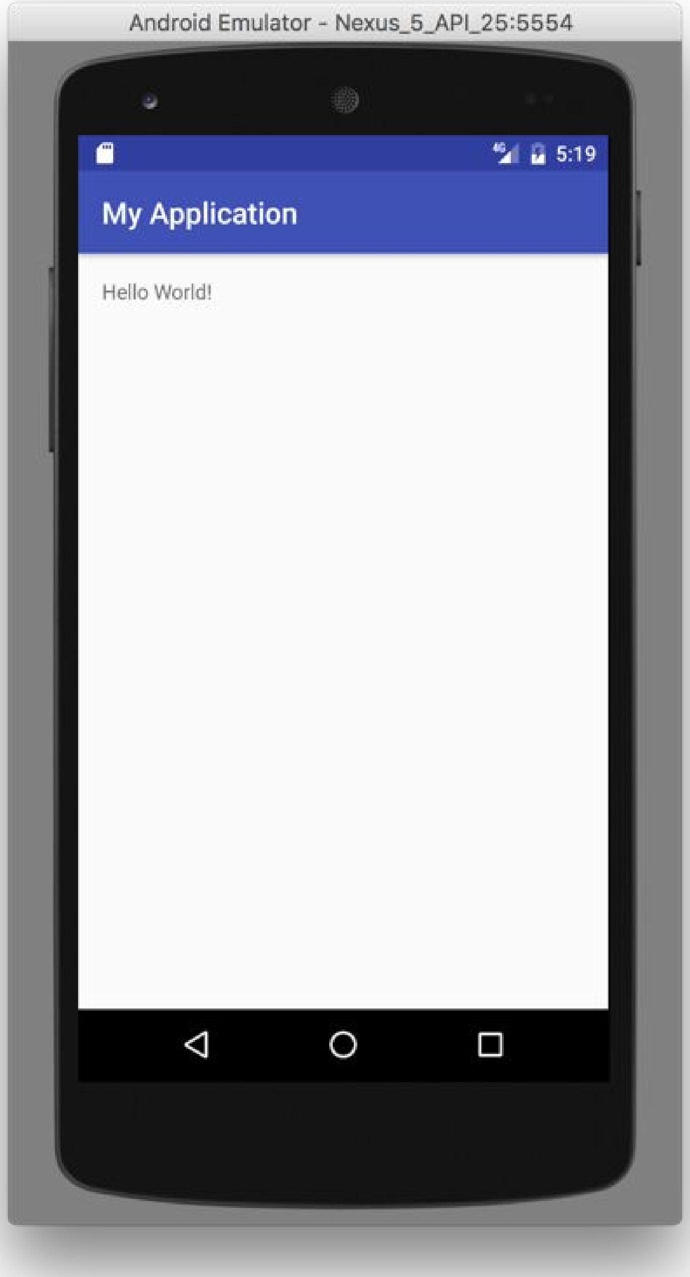
```
buildscript {
    ext.kotlin_version = '1.0.5'
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.2.2'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}
```

In the *app* one, there are some other changes:

- An *apply* for the plugin: *kotlin-android*
- A new dependency to the Kotlin library
- A new folder added to *sourceSets* that allows to add Kotlin files to *src/main/kotlin* folder. This can be useful in projects that have mixed Java and Kotlin code, but our Kotlin files can perfectly be created in the *src/main/java* folder.

Step 6: Run the project

You have everything ready! Run the project and you'll be running your first Android App written in Kotlin.



Android Emulator - Nexus_5_API_25:5554

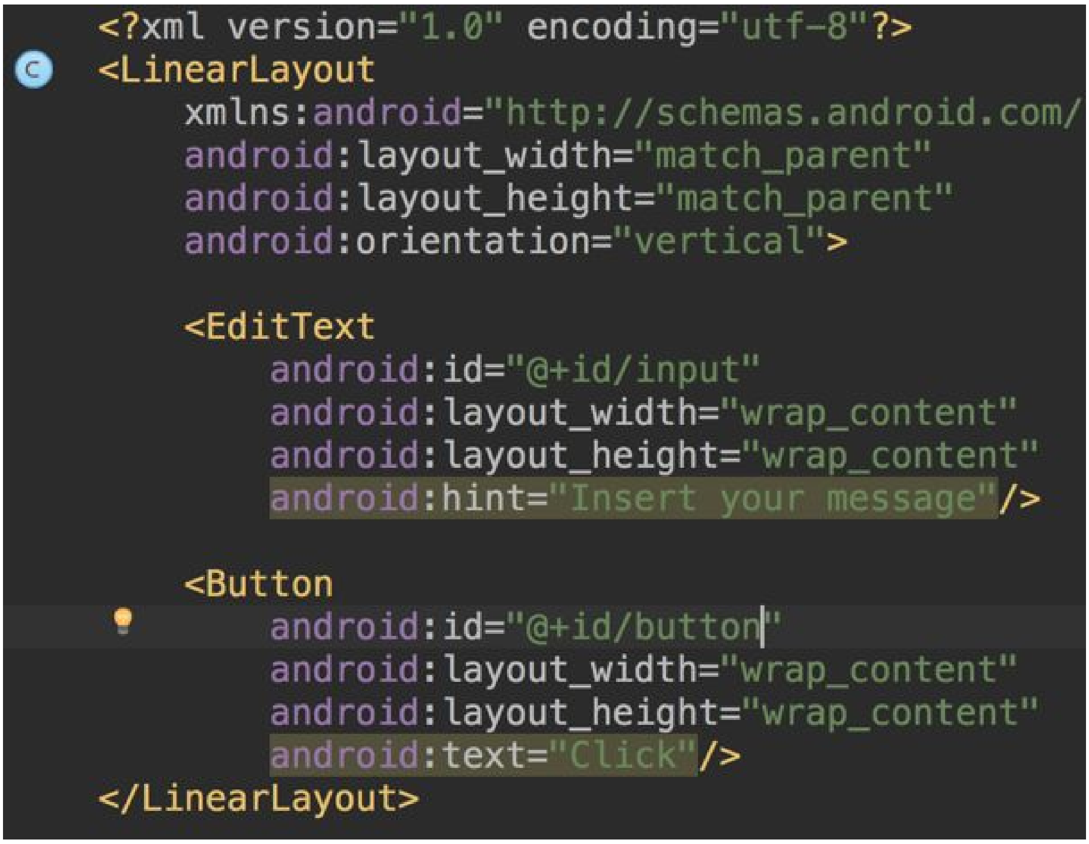
Hello World!

¡Bonus! Step 7: Interacting with the layout

I don't want to leave you without showing you some really interesting things about Kotlin, so let's create a layout with an *EditText* and a *Button*.

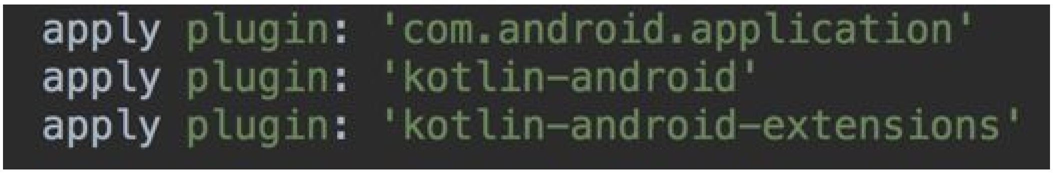
When you press the *Button*, it will create a toast that will show the value written in the *EditText*.

To do this, use an XML like this one:



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

    <EditText
        android:id="@+id/input"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Insert your message"/>

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click"/>
</LinearLayout>
```

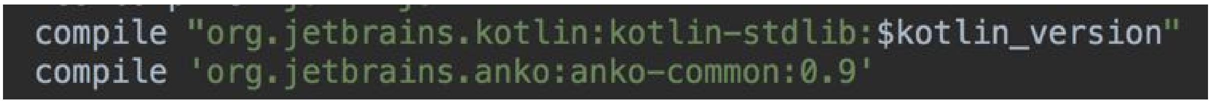
Now you'll evidence something amazing. Apply this new plugin inside the *build.gradle* in the *app* module:



```groovy
apply plugin: 'com.android.application'
apply plugin: 'kotlin-android'
apply plugin: 'kotlin-android-extensions'
```

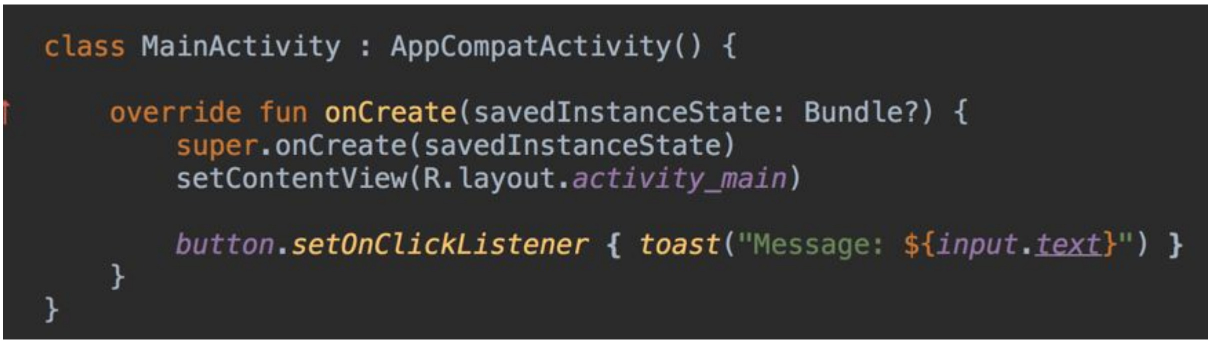
The *kotlin-android-extensions* plugin will let you recover the views from the XML without having to use *findViewById* or any other libraries. The views are just there ready to be used.

Also add Anko library, which I urge you to take a look when you have some more knowledge about Kotlin:



```
compile "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
compile 'org.jetbrains.anko:anko-common:0.9'
```

Here, we are only using its *toast* method. It's really easy, but I want you to see what you can achieve by using Kotlin:



```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        button.setOnClickListener { toast("Message: ${input.text}") }
    }
}
```

It's that easy! You don't need anything else to set a listener and, when the button is clicked, show a *toast* with the content of the *EditText*.

Run the app again, and check how it works.

Let me know your opinion!

What do you think about it? In no time you've been able to create a Kotlin project and to add some functionality, that in Java would have required at least 10 lines, in only one line.

This is a little view of what Kotlin can do from you as an Android developer.

Don't worry if you don't completely understand the code in the example. If you want to keep learning, in the next few days you'll receive some extra emails where I'll tell you about this and some other things, so that you can create complete Apps from scratch.

If you've received this guide by any other means, I recommend you to subscribe at antonioleiva.com so that you can receive the emails with the extra content.

You can also find the new content by following me on Twitter, where I also share articles that I find interesting.

Thanks, and talk again soon!

All the best,

Antonio Leiva
antonioleiva.com