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 three 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've written the book that both Google and Jetbrains recommend to start learning Kotlin applied to Android Development.

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, be 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.

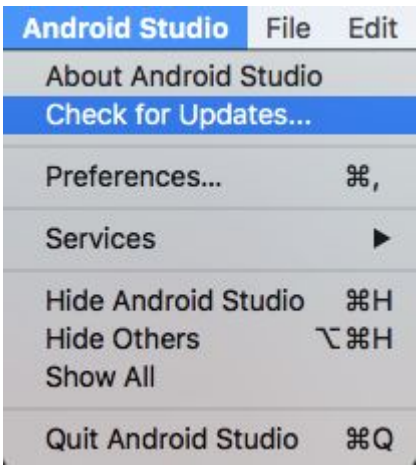
Thanks!
Antonio

Step 1: Install the latest stable version of Android Studio

It's possible that you have this step already done. Be careful with Canary versions, because Kotlin may be unstable on those.

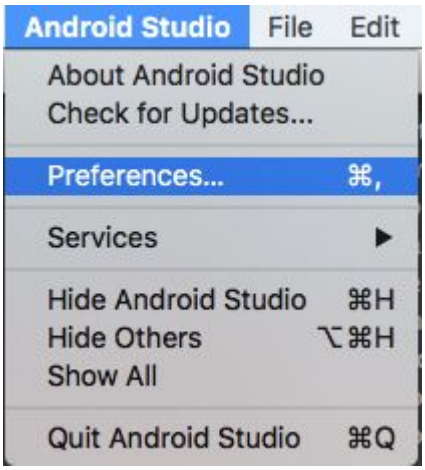
Latest version is Android Studio 3.0.

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.

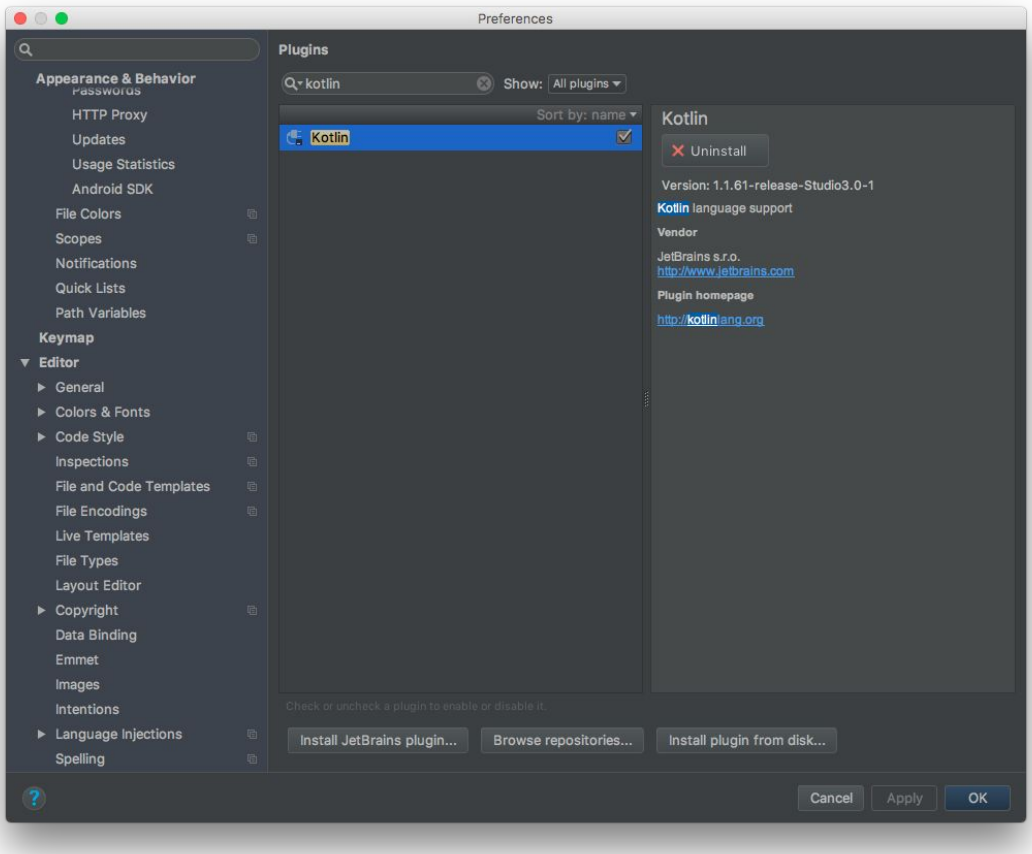
Step 2: Install the Kotlin plugin



Android Studio already includes the plugin installed by default, so we need don't need to install it manually.

To be sure you have the latest version of the plugin, go to the plugins configuration section in Android Studio. To do it, go to *Android Studio → Preferences…*

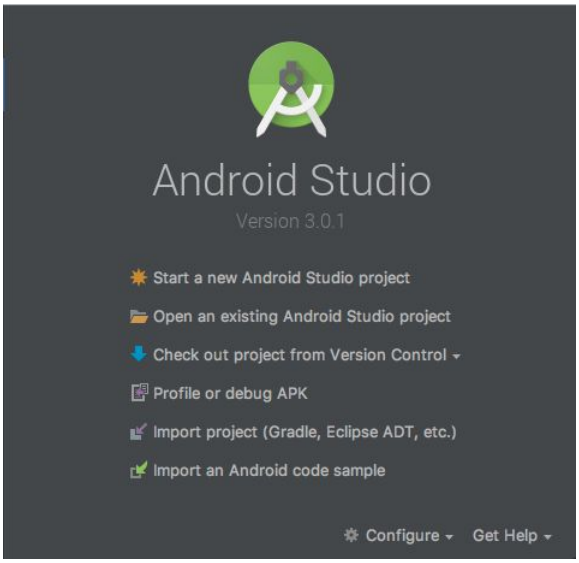
Then search *Plugins* in the side menu and click on it. Write "Kotlin" on the top search box. Check whether an "Update" button appears, and click on it in that case. Mine is up to date:



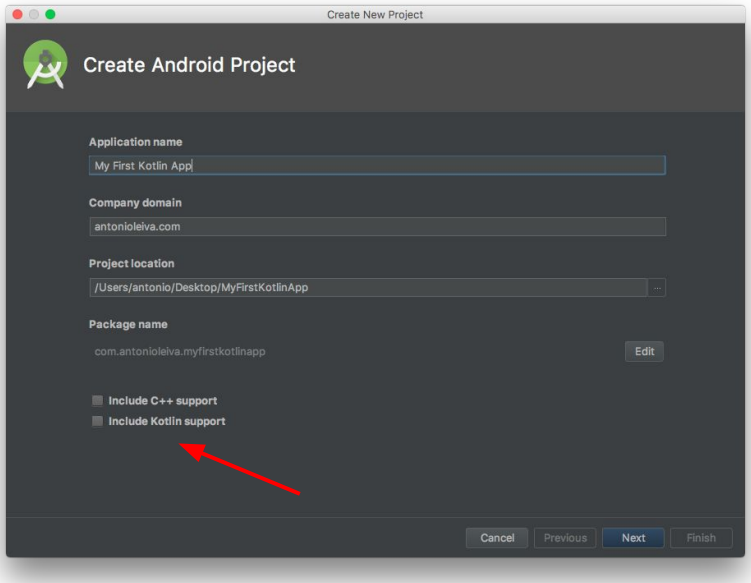
Step 3: Create a new Android Studio project

It is very simple. As I said, since Android Studio 3.0, Kotlin is fully integrated into the IDE, so you can create your projects directly in Kotlin. It's pretty straightforward.

However, I want to show you how to set it up by hand, which will be the case if you already have a Java project and want to start using Kotlin on it.



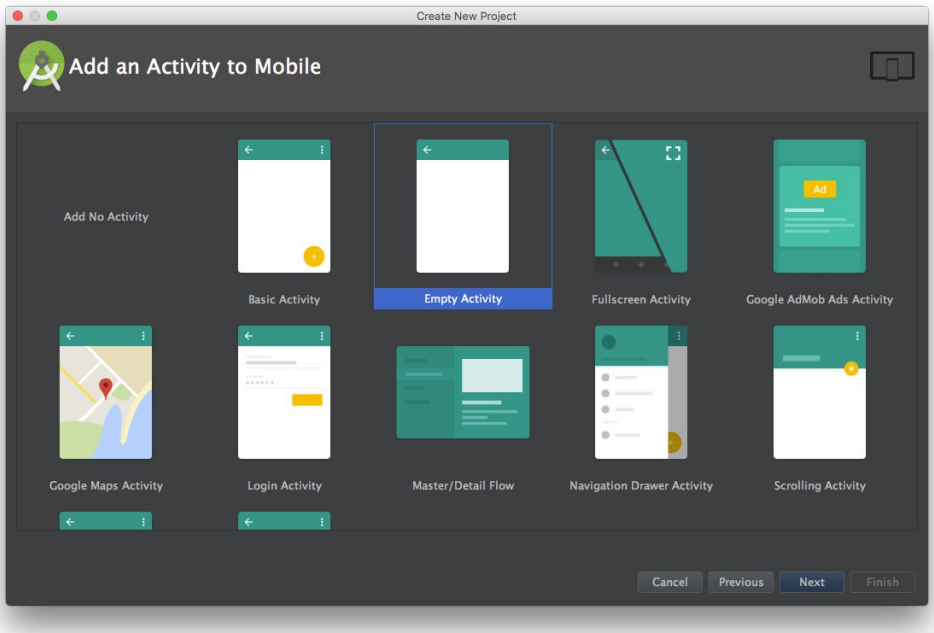
Choose *Start a new Android Studio Project*, and you'll see this screen:



Use the name and location you want. If you see, at the bottom you can check *Include Kotlin Support*. Normally you will use this option, but leave it unchecked because I want to show you a couple of things. 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, though nowadays it's pretty safe to use API 19.

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



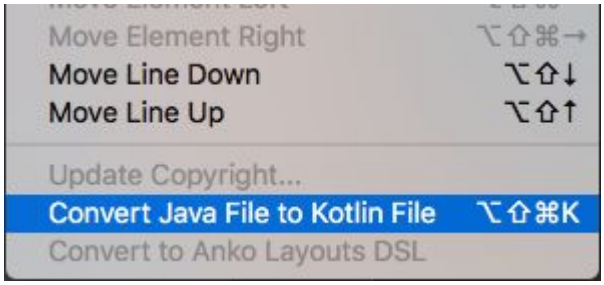
You can use the default values in the last screen of the wizard. Press *Finish*.

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:

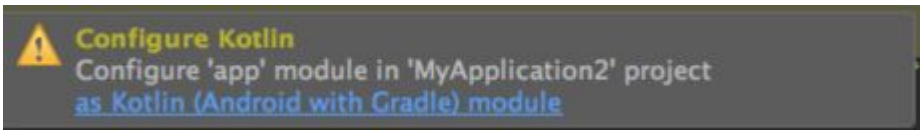
```
import android.support.v7.app.AppCompatActivity
import android.os.Bundle

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 content to help you with this).

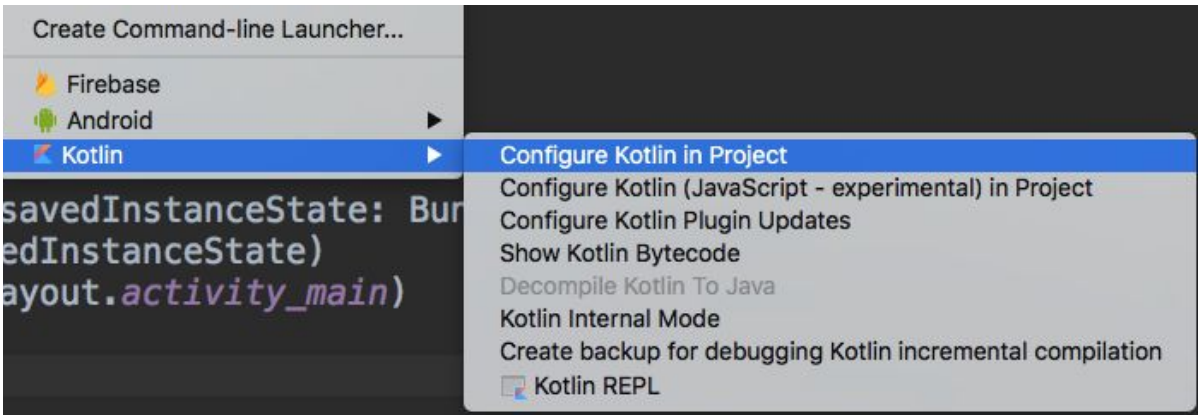
But you can see that, even having some differences, the code is perfectly understandable if you already know Java.

Step 5: Configure Kotlin in your project

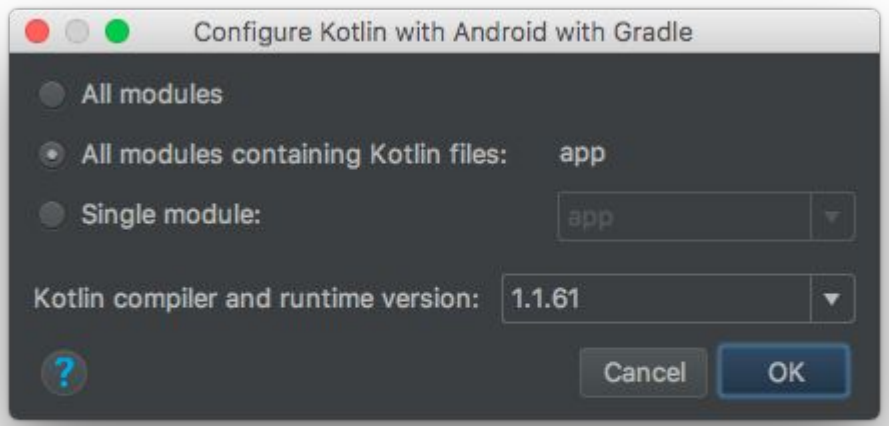
It's possible that you already see an alert like this one:



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:



A window like this one will open (the Kotlin version may differ):



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.

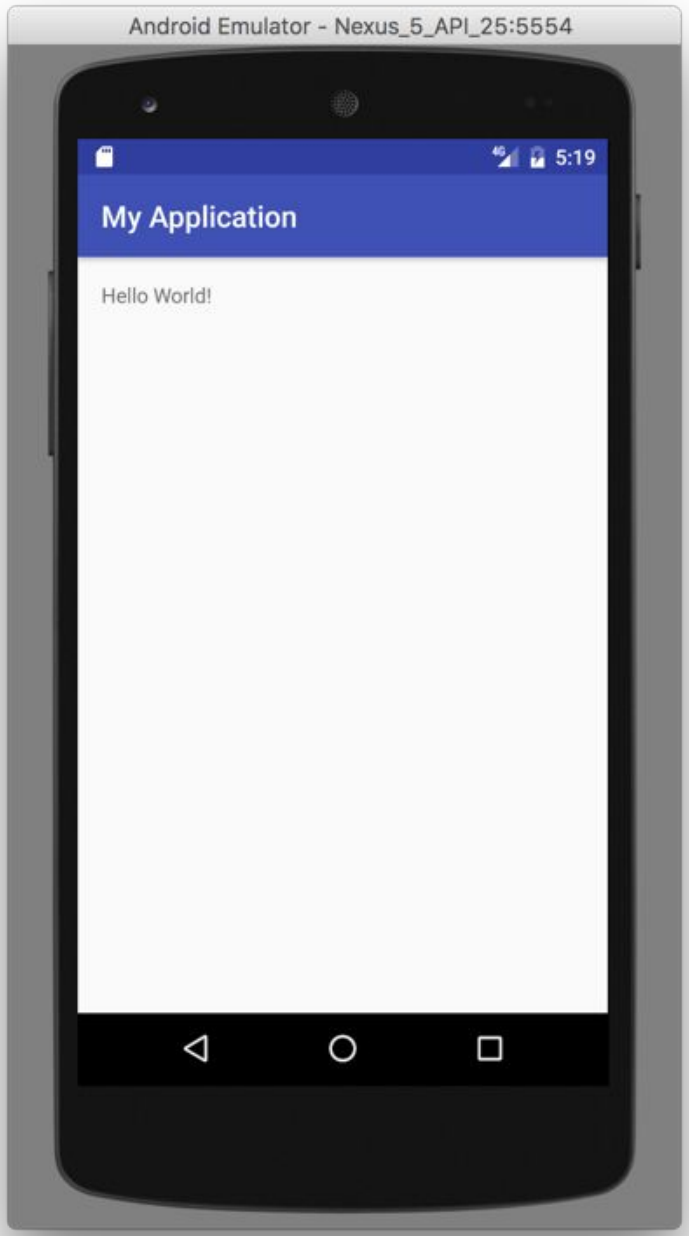
```
buildscript {
    ext.kotlin_version = '1.1.61'

    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.0.1'
        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

Step 6: Run the project

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



¡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 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 see something amazing. Apply this new plugin inside the *build.gradle* in the *app* module:

```
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:

```
implementation "org.jetbrains.kotlin:kotlin-stdlib-jre7:$kotlin_version"
implementation "org.jetbrains.anko:anko-common:0.10.3"
```

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 example of what Kotlin can do for 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 following 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