Android Development Fundamentals

#kpimobiledev

Tools

- Download Android Studio for your OS, following the instructions

  https://developer.android.com/studio/index.html - get stable 2.3.3. version

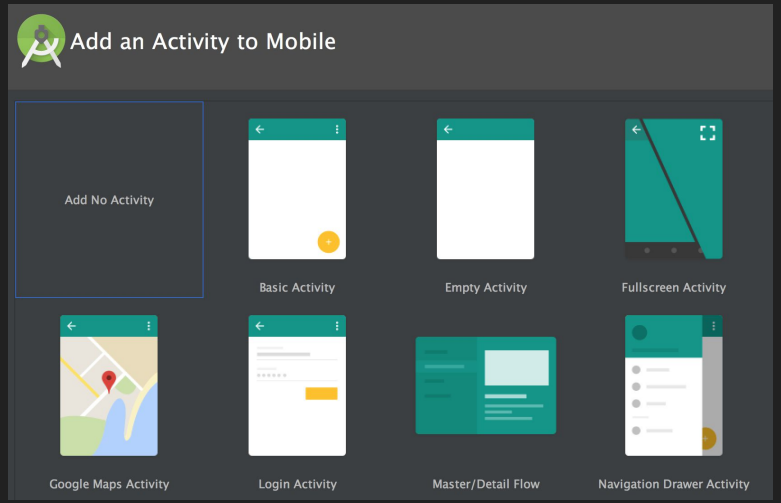
  https://developer.android.com/studio/preview/index.html - get 3.0. version (now in beta, but it’s stable enough and provides plenty nice features)

- Install Android SDK
- Configure Android Studio depending on your needs

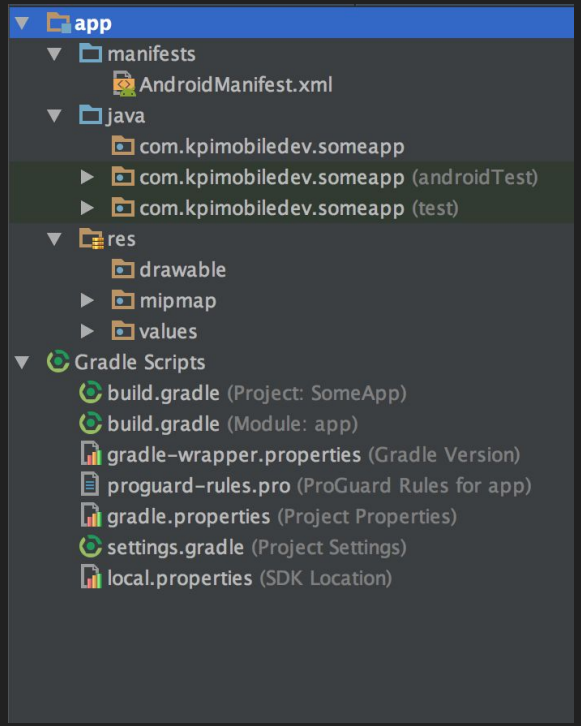
Let’s write some code

Create new Android Studio Project

- Set your app and company name
- If you use Android Studio 3.0, make sure “Include Kotlin Support” is checked, otherwise, you should configure kotlin later
- Select platforms and minimum SDK for them
- Choose “Add no Activity”
- Finish



Project Structure



- AndroidManifest.xml
- java folder for your app code and tests
- res folder for layouts, values, drawables, raw etc
- Gradle scripts for app building

Configure Kotlin (for Android Studio 2.3.3)

You should simply install kotlin plugin (go to File | Settings | Plugins | Install JetBrains plugin…), restart IDE and enjoy! Android Studio can show you ‘Kotlin not configured’ warning and offer fixit-solution for fast auto-configuring in project. After clicking ‘Configure’, your build-gradle files (both Project and Application level) will be changed.

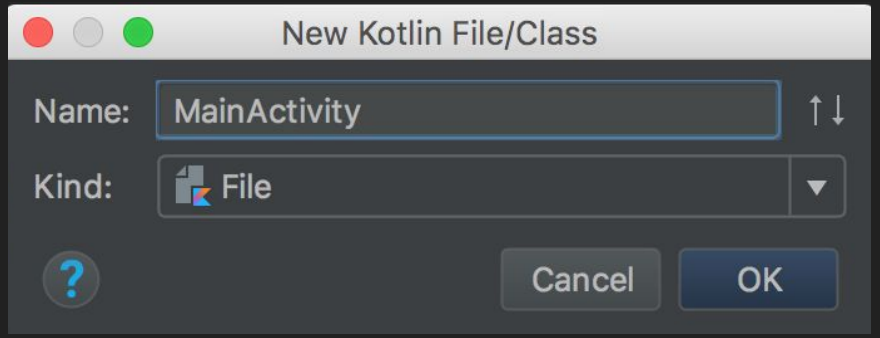
For build.gradle file of app level, add the following line:

```
apply plugin: 'kotlin-android-extensions'
```

For more info, read https://kotlinlang.org/docs/tutorials/kotlin-android.html

Create an Activity

Create class MainActivity (or any name you like) in your project package (in create variants, choose ‘Kotlin File/Class’



Create Activity class and make it extend AppCompatActivity

You can use another name for Activity, but something ending with “Activity” word is preferred. Click cmd (ctrl) + O inside the class body to see all methods that can be overwritten. Choose onCreate() method, override it and add set its content view like here:

```
package com.kpimobiledev.someapp

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

/**
 * @author lidaamber
 */
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

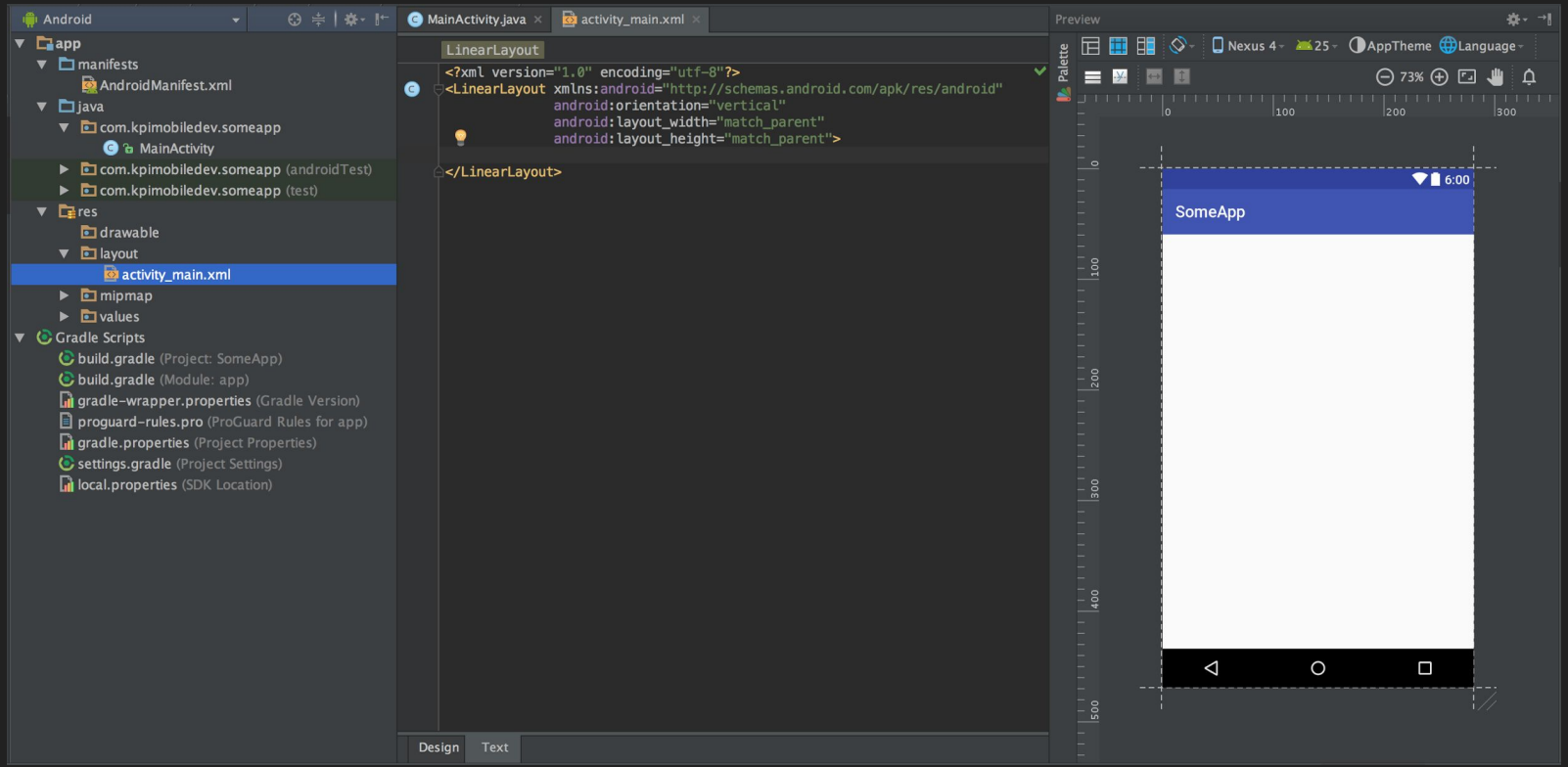
Create layout resource for your activity

If you write that code at first, Android Studio will highlight the mistake like “cannot resolve symbol activity_main”

Alt+Enter to automatically create layout resource with specified name

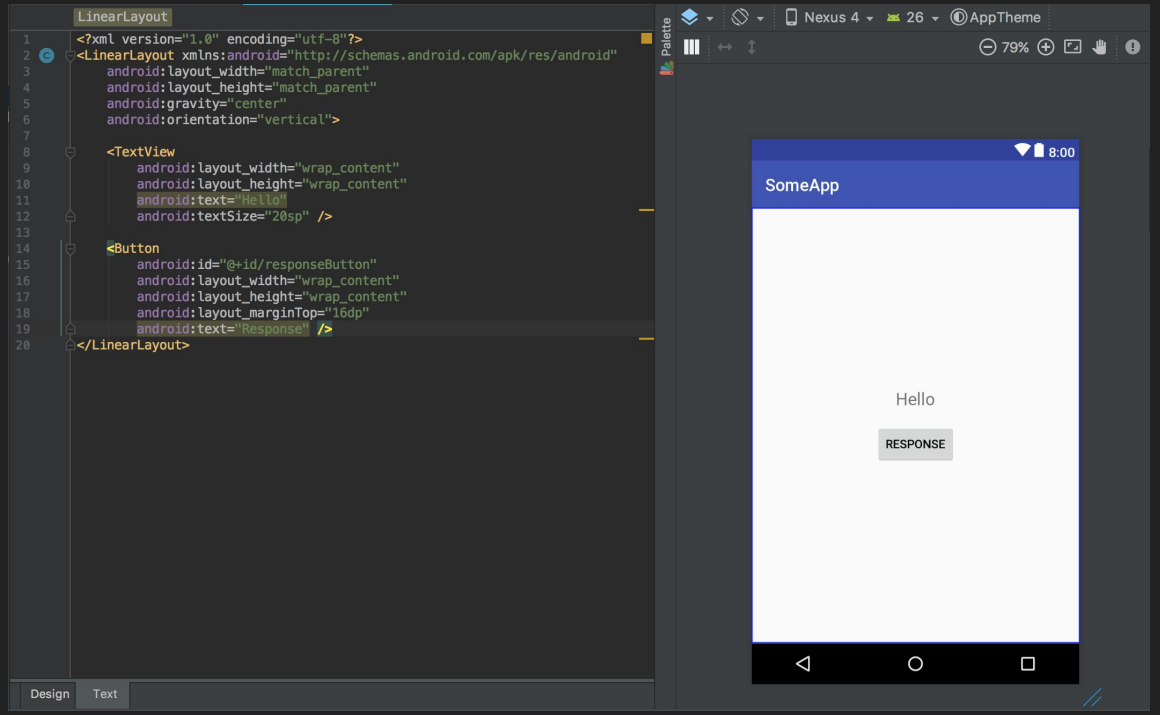
Or, actually, you can create it manually. Create folder “layout” in “res” directory and then create .xml-file with the name of your layout. Specify it in onCreate() method of your activity.

Congratulations, you've made your first layout!



Let’s place something there

For instance, place TextView element and a button in the center of your layout, give a button an id. You can use Design Tab and simply drag elements to screen layout or write some xml. For future, try better use Constraint Layouts and design them only with Design tab.



Now let’s get back to Activity

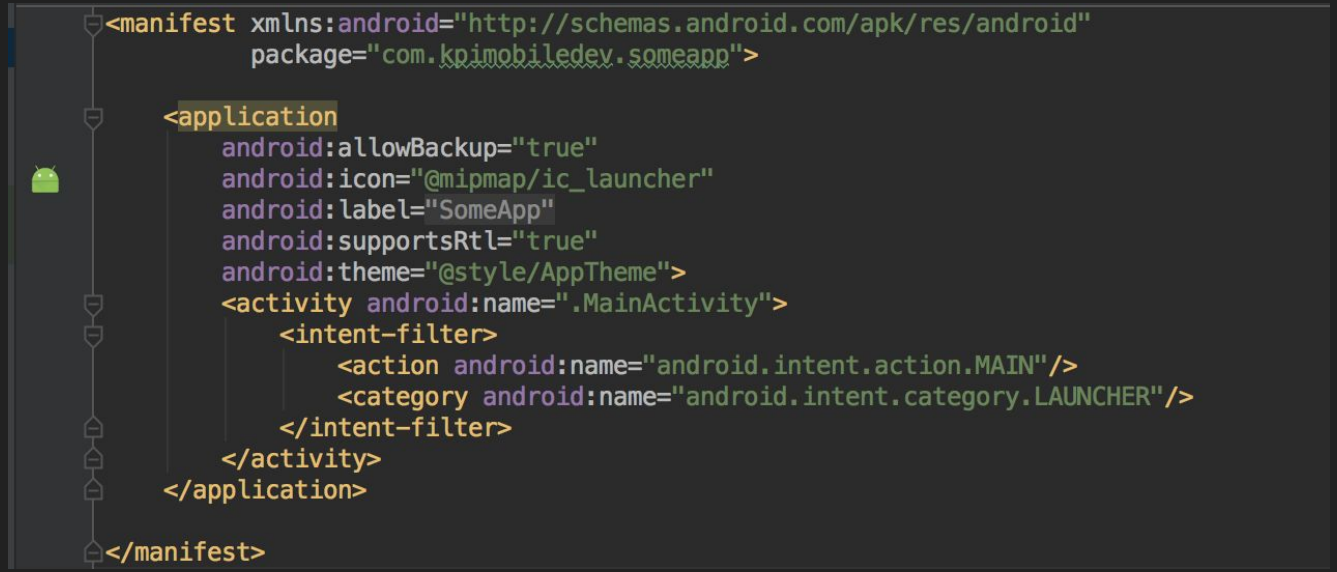
In Java, you would need to find your button on view by its id. If you don’t use kotlin extensions plugin, you would need to do that like in Java, but usage of kotlin extensions is highly recommended to avoid boilerplate code and findViewById() hell. Kotlin extensions take all the binding job on it, and you can use layout elements just requesting their ids on layout. Create OnClickListener to your button. Make it respond you :)

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

    responseButton.setOnClickListener {
        Toast.makeText(this, "Hello back!", Toast.LENGTH_LONG).show()
    }
}
```

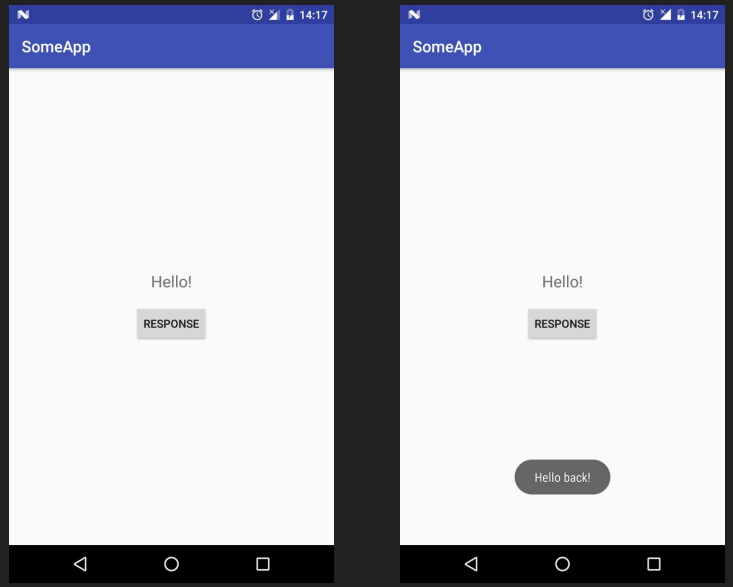
Manifest manipulations

Open AndroidManifest.xml and add information about your activity there.



Run your app

Run your app by Ctrl+R. Choose a device to run on: connect your device to the computer or use the emulator. Click the button to check the response.

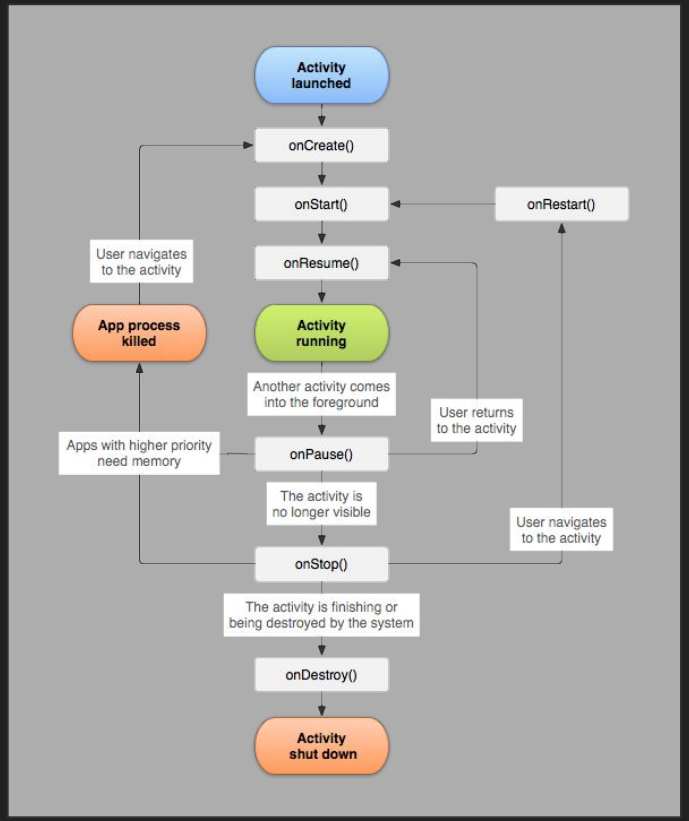


You've just made your first app
Now let's go deeper

What’s the Activity?

- An entry point in interaction with user
- One of the main app components
- Represents a screen with UI
- Responsible for certain *option* of your app
  - ChatActivity
  - MailListActivity
  - ProfileActivity
  - etc

https://developer.android.com/guide/components/activities.html



Activity Lifecycle

Is there anything besides Activity?

Yes, there are other components of application, like

- Services
- ContentProviders
- BroadcastReceivers

AndroidManifest.xml

It represents the most important information about your app

- Java package which serves as identifier of your application
- Description of all app components (Activities, Services, ContentProviders and BroadcastReceivers)
- SDK and libraries you use
- Permissions
- etc

https://developer.android.com/guide/topics/manifest/manifest-intro.html

What does MainActivity description mean?

```
<activity android:name=".MainActivity">
    <intent-filter>                                          // What intents can Activity handle
        <action android:name="android.intent.action.MAIN"/>  // This is a main entry point of application
        <category android:name="android.intent.category.LAUNCHER"/>  // This activity will be first called when app starts
    </intent-filter>
</activity>
```

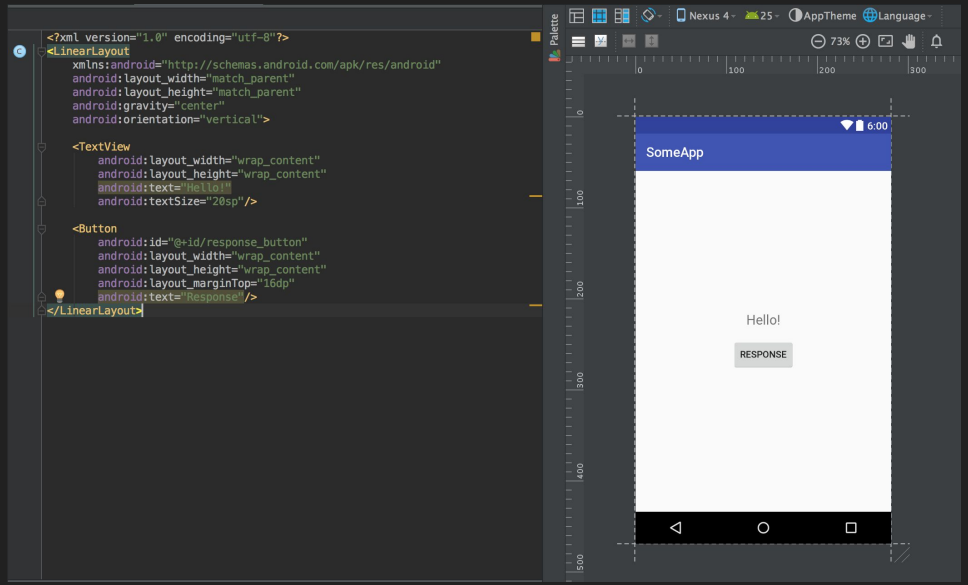
Let’s get back to onCreate()

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

    responseButton.setOnClickListener {
        Toast.makeText(this, "Hello back!", Toast.LENGTH_LONG).show()
    }
}
```

setContentView() method binds layout file to activity and sets its view as rendered layout. View is a container for nested layout elements, so you can use your layout elements in code to get/set their properties, make event listeners for them etc. In Kotlin and using Kotlin Extensions, you can simply request your layout elements by their identifiers and work with them like with objects. In such a way, you can use button from your layout by writing its identifier responseButton and set OnClickListener to the button, or passing closure to execute when the button is clicked (like we did in code above). Once button is clicked, closure content is executed.

And to layout




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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello!"
        android:textSize="20sp"/>

    <Button
        android:id="@+id/response_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="Response"/>
</LinearLayout>
```

LinearLayout is a container where nested elements are represented one after another, like in list, aligned vertically or horizontally. There are also other containers, most of them end with Layout word and they differ by ways of positioning nested elements. All elements, both containers and simple ones, must have layout_width and layout_height attributes. You should give element an identifier to find it in code. With newer versions of Android Studio, developers are highly recommended to use ConstraintLayouts and forget about working with raw xml.

Q&A

@lidaamber