Android Development Fundamentals

#kpimobiledev

Tools

- Download Android Studio for your OS, following the instructions

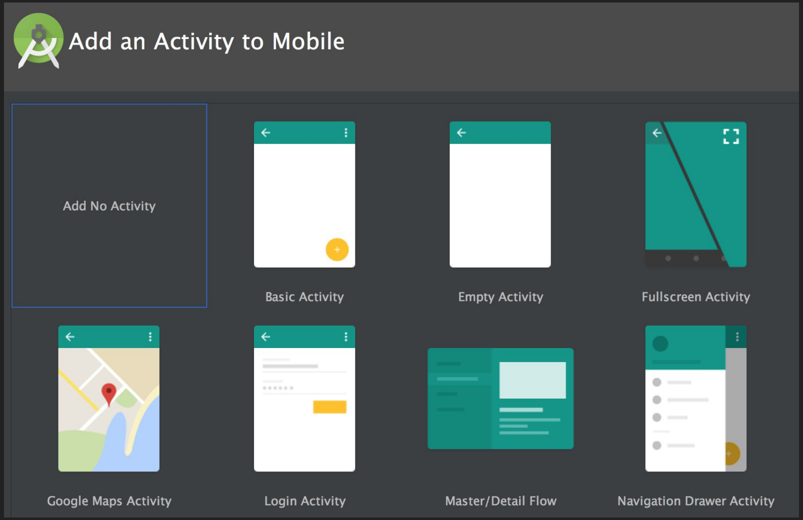
  https://developer.android.com/studio/index.html

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

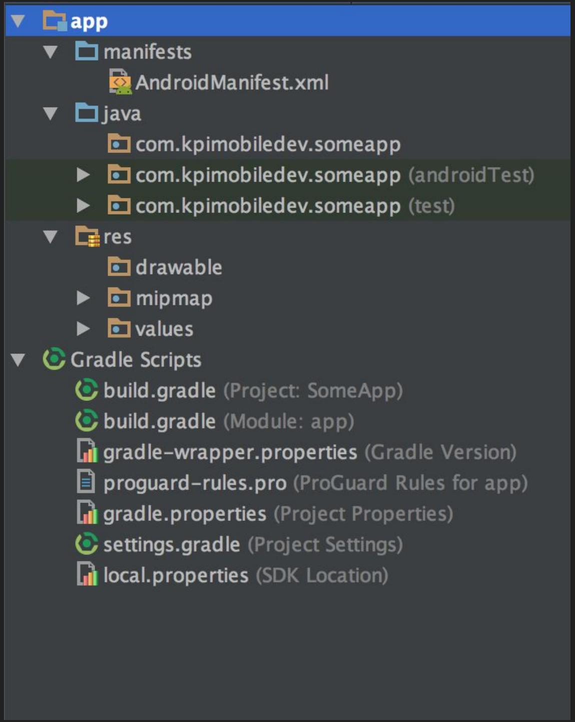
Let’s write some code

Create new Android Studio Project

- Set your app and company name
- Select platforms and minimum SDK for them
- Choose “Add no Activity”
- Finish



Project Structure

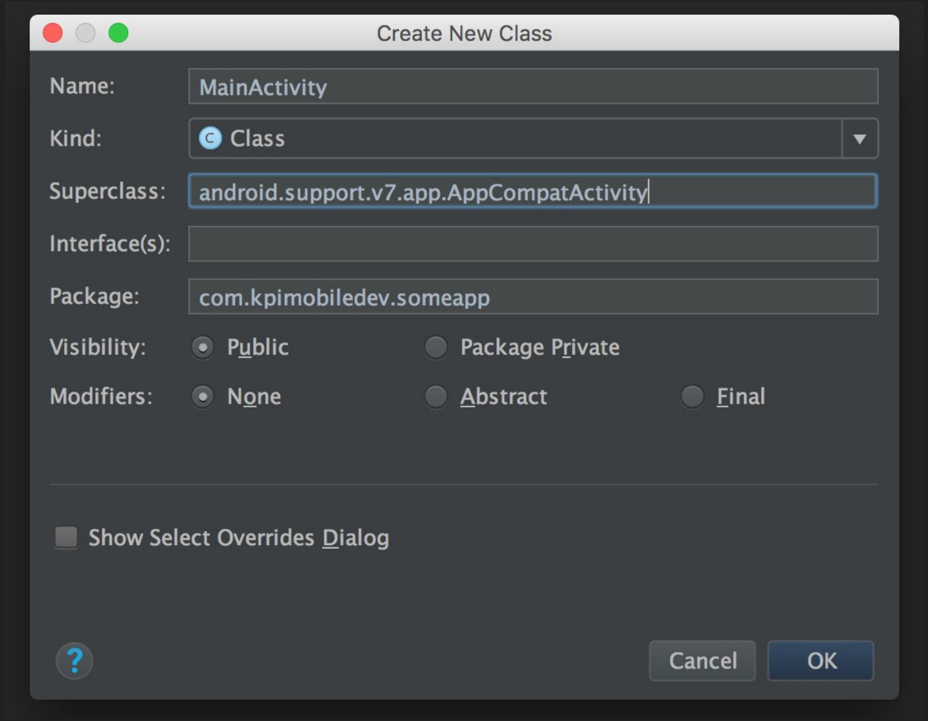


```
▼ app
    ▼ manifests
          AndroidManifest.xml
    ▼ java
          com.kpimobiledev.someapp
        ▶ com.kpimobiledev.someapp (androidTest)
        ▶ com.kpimobiledev.someapp (test)
    ▼ res
          drawable
        ▶ mipmap
        ▶ values
▼ Gradle Scripts
      build.gradle (Project: SomeApp)
      build.gradle (Module: app)
      gradle-wrapper.properties (Gradle Version)
      proguard-rules.pro (ProGuard Rules for app)
      gradle.properties (Project Properties)
      settings.gradle (Project Settings)
      local.properties (SDK Location)
```

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

Create an Activity

Create class MainActivity (or any name you like) in your project package and make it extend AppCompatActivity


Create New Class

Name: MainActivity

Kind: Class

Superclass: android.support.v7.app.AppCompatActivity

Interface(s):

Package: com.kpimobiledev.someapp

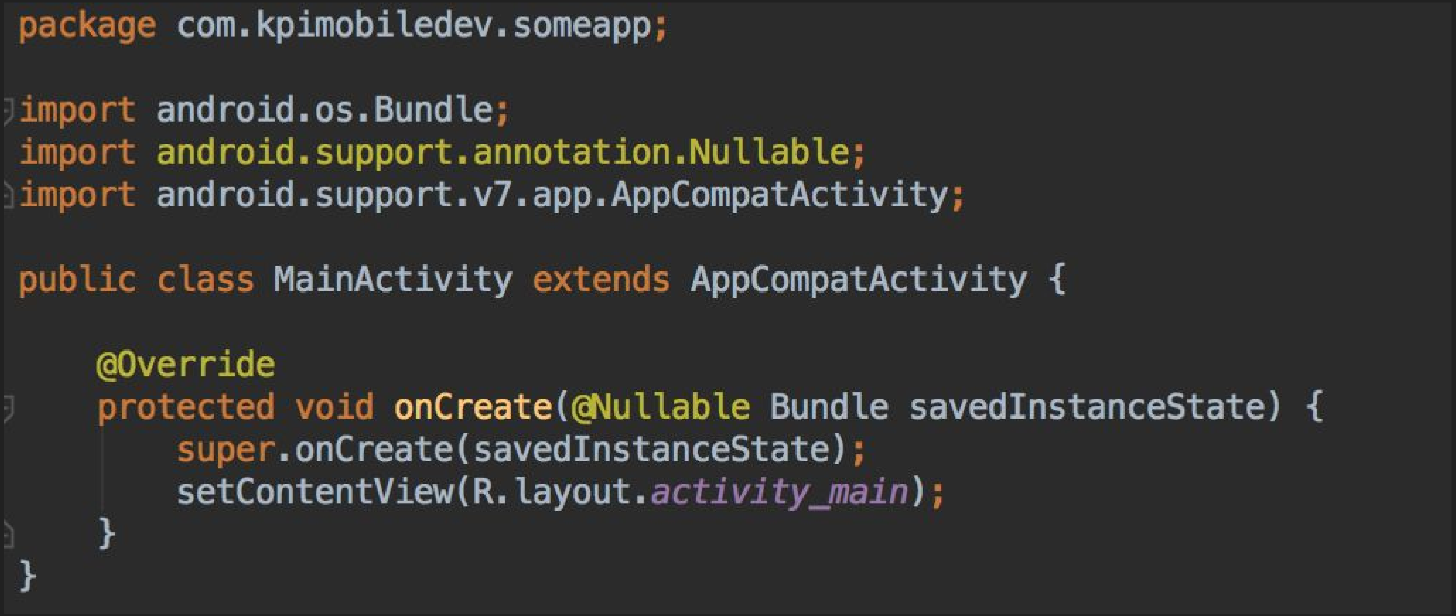
Visibility: [✓ Public] Package Private

Modifiers: [✓ None] Abstract Final

☐ Show Select Overrides Dialog

Override the onCreate() method of your activity

cmd (ctrl) + O to see all methods that can be overwritten



```java
package com.kpimobiledev.someapp;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

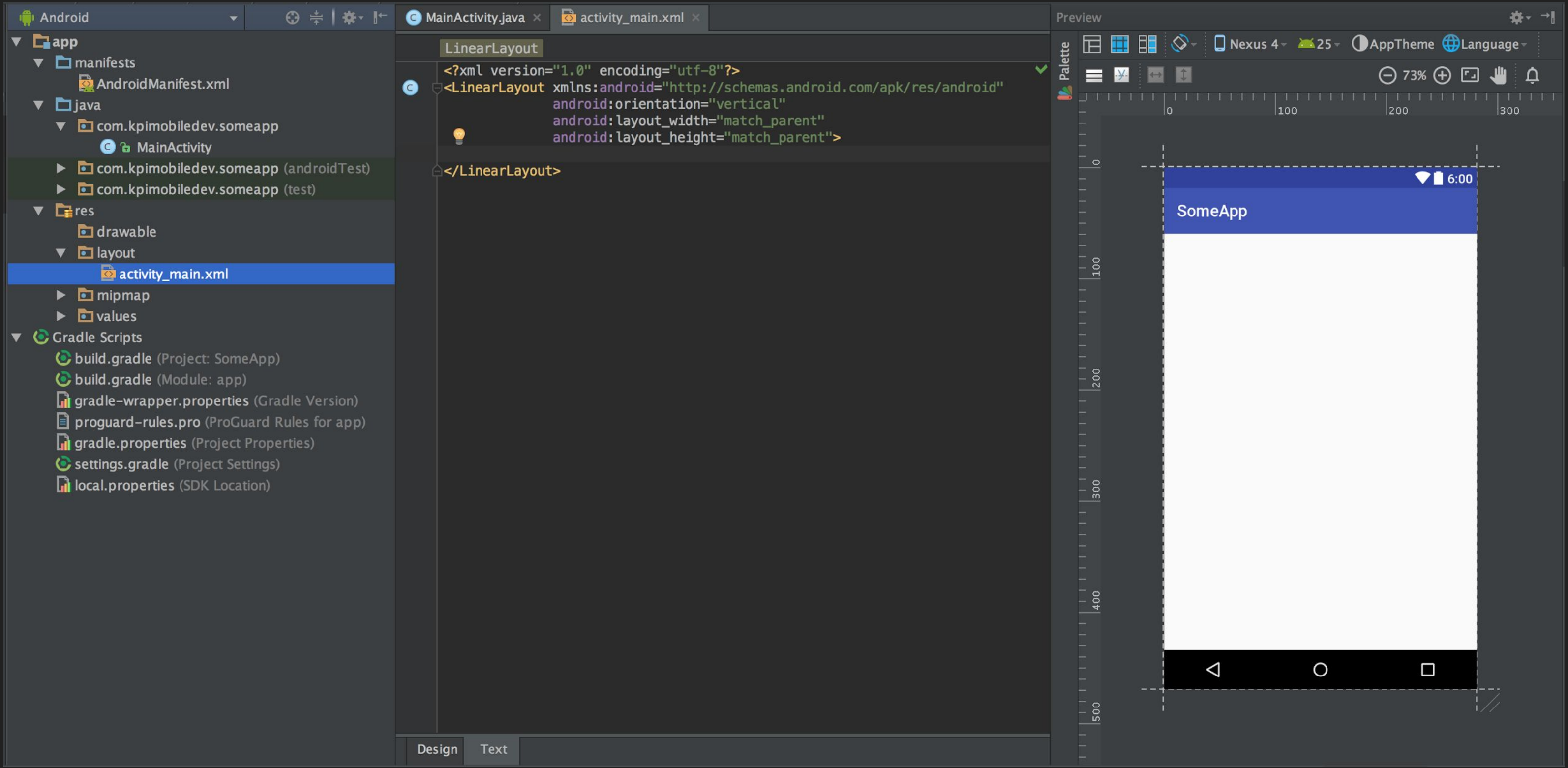
Create layout resource for your activity

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!

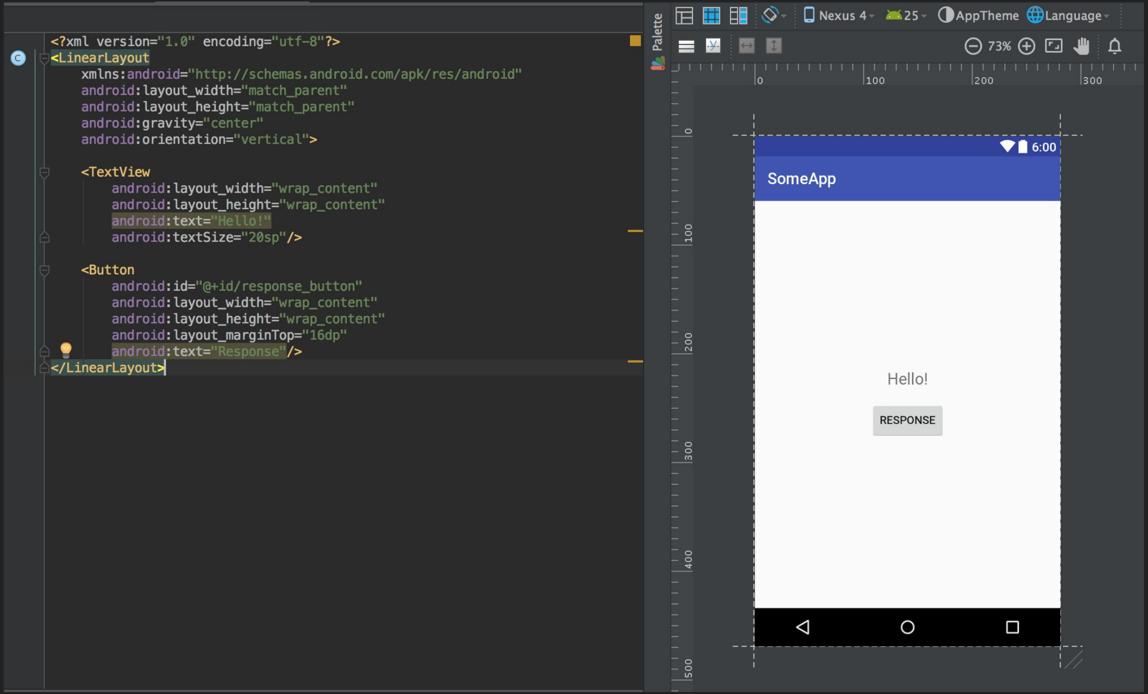


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_width="match_parent"
              android:layout_height="match_parent">

</LinearLayout>
```

Android
app
manifests
AndroidManifest.xml
java
com.kpimobiledev.someapp
MainActivity
com.kpimobiledev.someapp (androidTest)
com.kpimobiledev.someapp (test)
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
build.gradle (Project: SomeApp)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
MainActivity.java
activity_main.xml
LinearLayout
Design
Text
Preview
Palette
Nexus 4
25
AppTheme
Language
73%
6:00
SomeApp

Let’s place something there

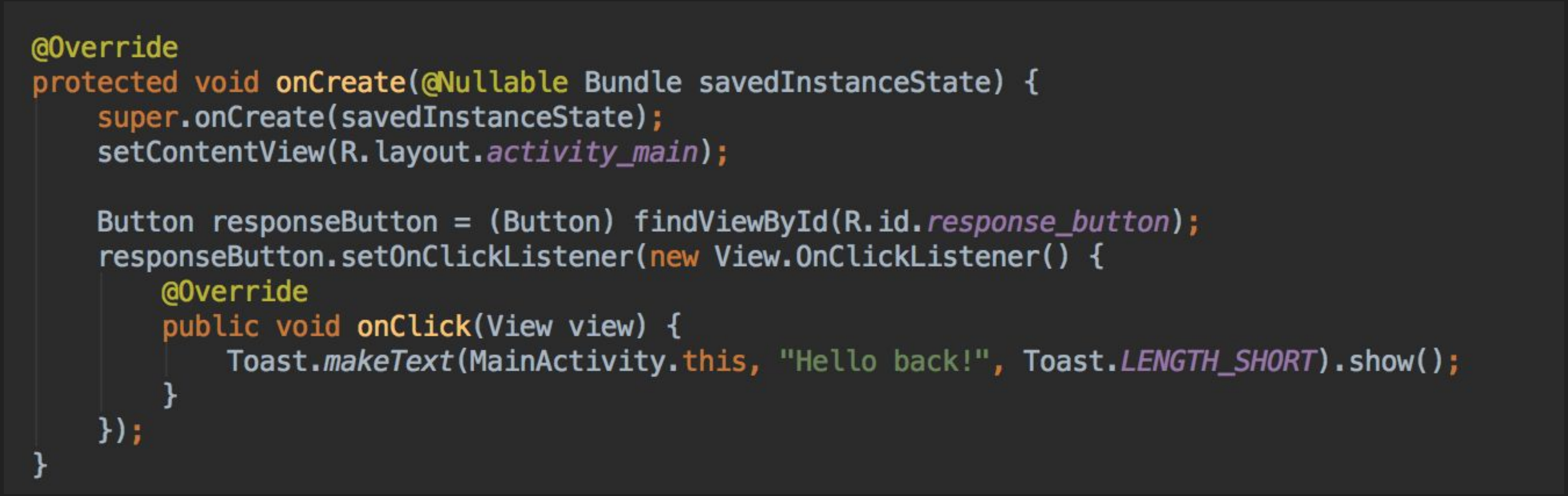
For instance, place TextView element and a button in the center of your layout, give a button an id. You can use Design Tab and simply drag elements to screen layout or write some xml.



```
<?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>
```

Now let's get back to Activity

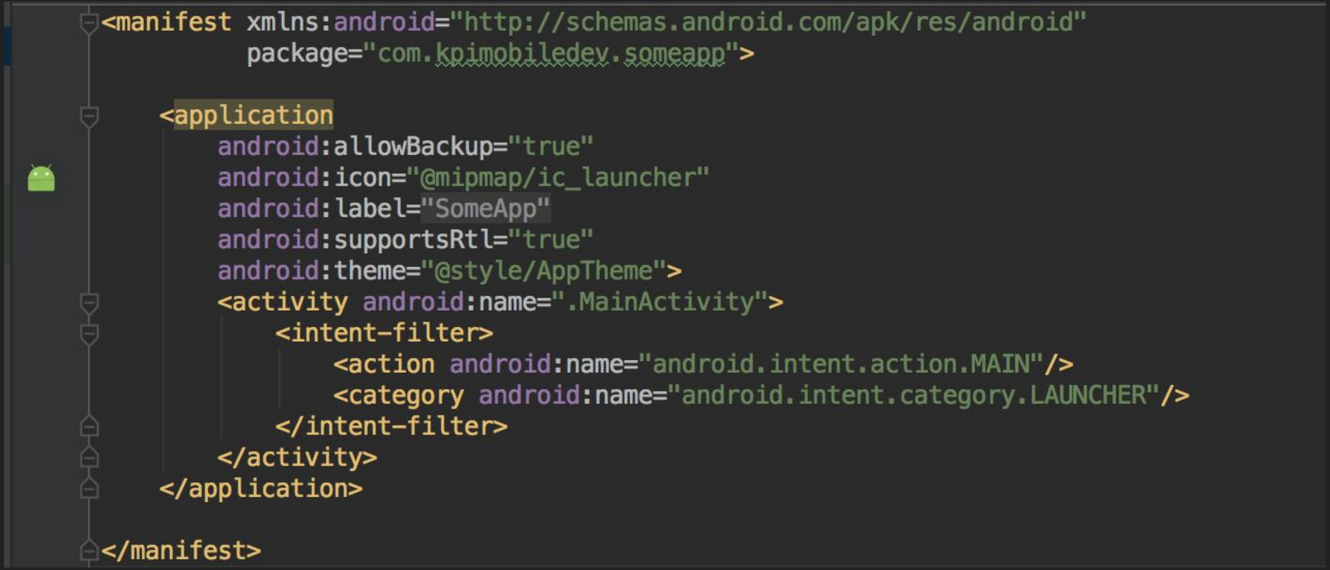
Now you need to bind your layout button to object, responsible to it. Find it by id and create OnClickListener to your button. Make it respond you :)



```java
@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Button responseButton = (Button) findViewById(R.id.response_button);
    responseButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Toast.makeText(MainActivity.this, "Hello back!", Toast.LENGTH_SHORT).show();
        }
    });
}
```

Manifest manipulations

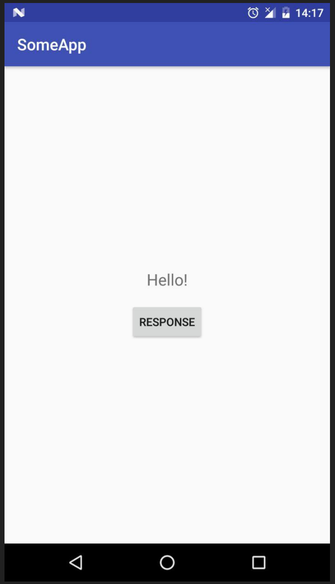
Open AndroidManifest.xml and add information about your activity there.

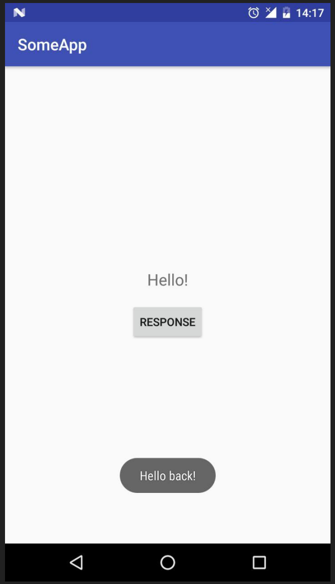


```xml
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
          package="com.kpimobiledev.someapp">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="SomeApp"
        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>
```

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.



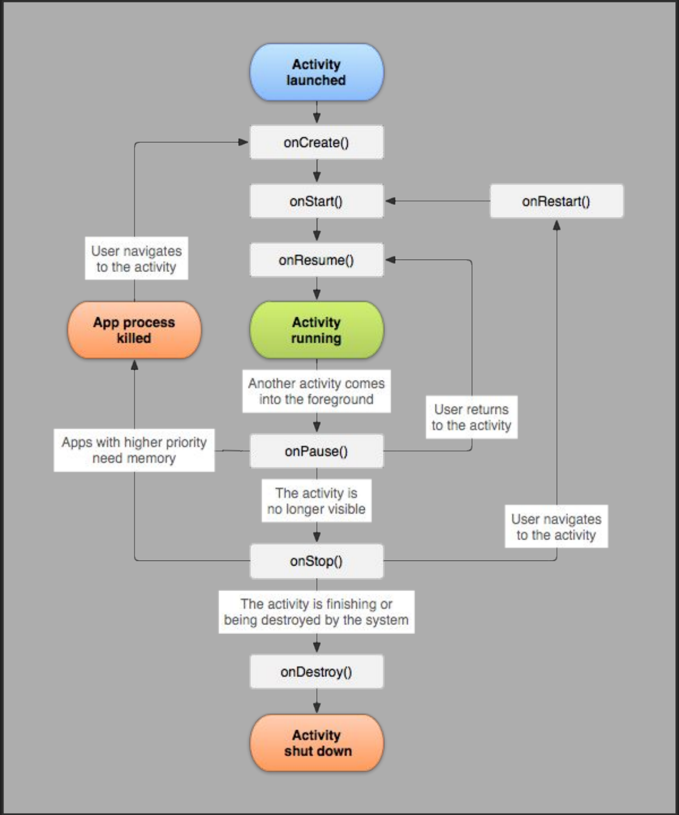


You've just made your first app
Now let's go deeper

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 launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

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

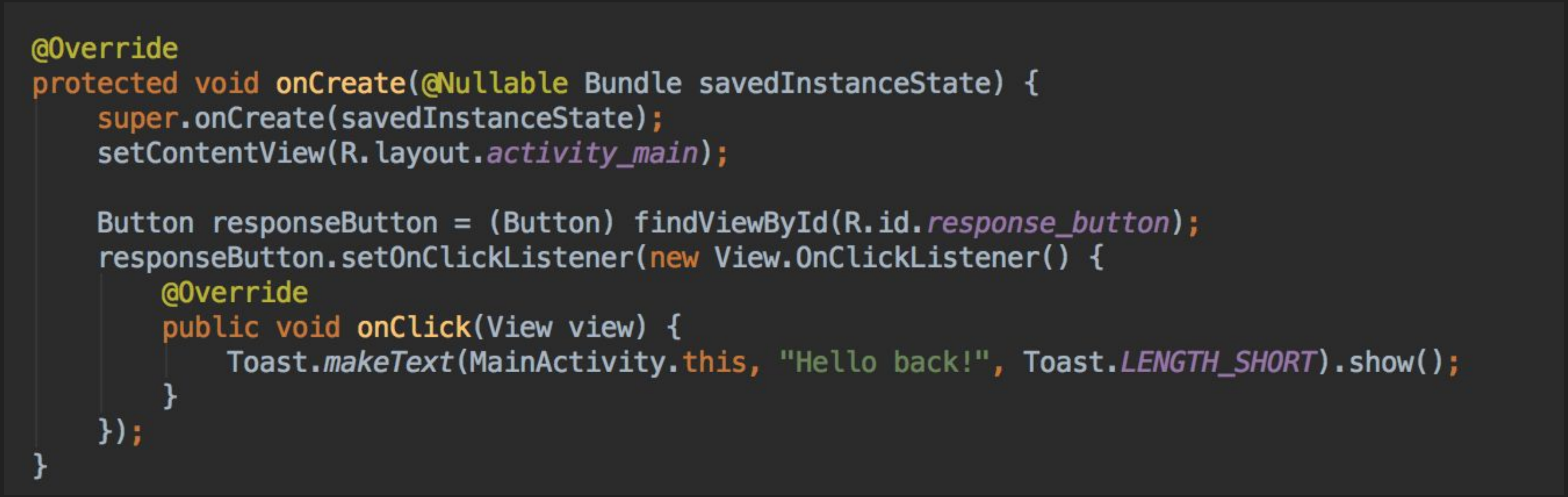
- Java package which serves as identifier of your application
- Description of all app components (Activities, Services, ContentProviders and BroadcastReceivers)
- SDK and libraries you use
- Permissions
- etc

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>
```

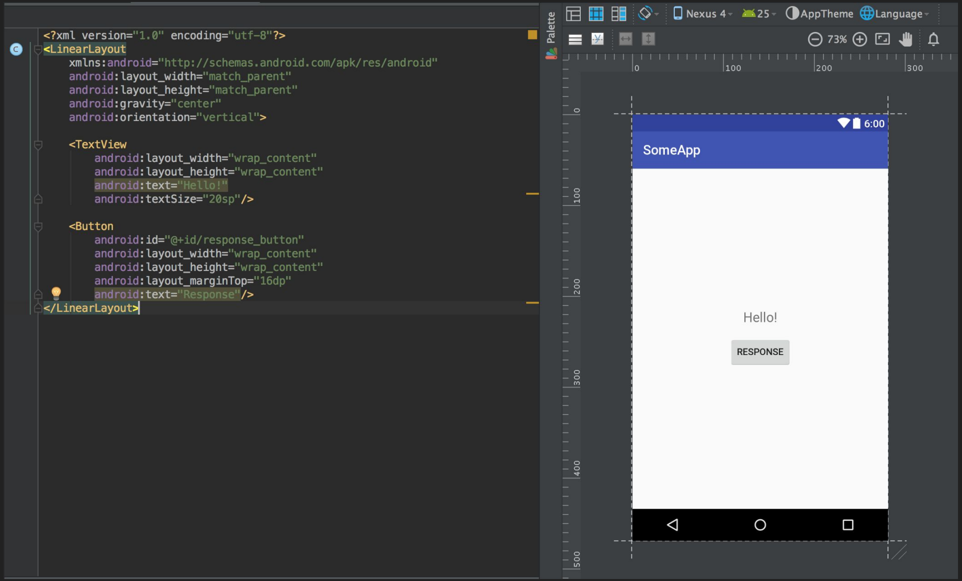
Let’s get back to onCreate()



```
@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Button responseButton = (Button) findViewById(R.id.response_button);
    responseButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Toast.makeText(MainActivity.this, "Hello back!", Toast.LENGTH_SHORT).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 look for your layout elements in code to get/set their properties, make event listeners for them etc. In such a way, you can find button element with findViewById() method and set OnClickListener to the button. Once button is clicked, the onClick() method of the listener is called.

And to layout



```xml
<?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