Project Overview

This lesson will cover Android project structure, main project files and folders.

WE'LL COVER THE FOLLOWING

- Structure
- Project files
- Module files
- Android widget

Structure

Let’s take a look at the structure of what a typical Android project looks like.

- *app* - root module folder
  - *build.gradle* - module config file
  - *src/main/AndroidManifest.xml* - module manifest file
  - *src/main/java* - module source folder for Java or Kotlin files
  - *src/main/res* - module resource folder
- *build.gradle* - project config file
- *gradle, gradle.properties, gradlew, gradlew.bat* - Gradle related files for to build android project
- *settings.gradle* - project settings file

```
project-name
├── app
│   ├── build.gradle
│   └── src
│       └── main
│           ├── AndroidManifest.xml
│           ├── java
│           │   └── com/travelblog/MainActivity.java
│           └── res
│                  ├── layout/activity_main.xml
│                  └── values/

├── build.gradle
├── gradle
│   └── wrapper
│       ├── gradle-wrapper.jar
│       └── gradle-wrapper.properties
├── gradle.properties
├── gradlew
├── gradlew.bat
└── settings.gradle
```

The android project may consist of one or several modules. Small to medium projects usually have one module, while large projects tend to have multiple modules. Each module may contain a separate feature or common logic. The definition and composition of modules will be explained in detail later in the lesson.

Project files usually apply some configuration to every module, which is part of this project. Module files usually contain our source code and resources.

Project files

The *settings.gradle* file is a project file which contains the list of modules and project name. In our case module name is *app* and the project name is '*Travel Blog*'.

```
include ':app'
rootProject.name='Travel Blog'
```

settings.gradle

The *gradle.properties* file defines settings to configure a build environment.

```
# Project-wide Gradle settings.
# IDE (e.g. Android Studio) users:
# Gradle settings configured through the IDE *will override*
# any settings specified in this file.
# For more details on how to configure your build environment visit
# http://www.gradle.org/docs/current/userguide/build_environment.html
# Specifies the JVM arguments used for the daemon process.
# The setting is particularly useful for tweaking memory settings.
org.gradle.jvmargs=-Xmx1536m
```

gradle.properties

gradle.properties

The *gradle, gradlew, gradlew.bat* - files related to Gradle wrapper. This means that we don’t have to manually install Gradle ourselves.

Finally, *build.gradle* file is a top-level build file where we can add configuration options common to all modules. In our case we want all of our modules to have access to *Google’s Maven* repository and *Bintray’s JCenter* repository for dependencies. These repositories allow projects to use some core Android functionalities.

```
buildscript {
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.5.0'
    }
}

allprojects {
    repositories {
        google()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

build.gradle

Module files

Every module has a unique name, in our case module is called *app*. That’s where we put application source code.

The module *build.gradle* file contains a set of configurations related to this module only, such as:

- `compileSdkVersion` - the version of Android SDK to compile the project
- `minSdkVersion` - the minimal supported Android version
- `targetSdkVersion` - the target version of Android SDK, used to tell the system to enable compatibility behaviours
- `applicationId` - unique identifier of the application on the device and in

Google Play Store

- `versionCode` - an internal version number
- `versionName` - the version name displayed to users
- `compileOptions` - compile options to achieve some features of Java 1.8
- `dependencies` - first-party and third-party library dependencies, discussed in the next lessons

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "com.travelblog"
        minSdkVersion 21
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
    }

    compileOptions {
        sourceCompatibility = 1.8
        targetCompatibility = 1.8
    }
}

dependencies {
    // ui
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.google.android.material:material:1.1.0-alpha10'
}
```

build.gradle

The *AndroidManifest.xml* is the place where we declare all main components used in our application, such as activities, services, permissions, etc.

Our manifest file currently contains the following:

- `package` - the package name of the application, in our case `com.travelblog`
- `theme` - the global application theme, in our case `MaterialComponents` theme
- `label` - the label which is used as a value for the application icon
- `activity` - the activity, we currently only have one `MainActivity`
- `intent-filter` - the set of options used to give the activity some unique behaviour, in our case we declare that it's the main activity which should

be launched when user click on the application icon from the launcher

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.travelblog">
    <application
        android:theme="@style/Theme.MaterialComponents.DayNight.NoActionBar"
        android:label="Travel Blog">
        <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>
```

AndroidManifest.xml

All resource-related files must be placed inside one of the predefined, sub-folders of *src/main/res* folder. One of such pre-defined folders is the *layout* folder, which contains all of our layout files. The other is the *values* folder, which usually contains some colors, styles, dimensions, etc.

For now, we only have one layout file, *activity_main.xml*, which describes layout structure. We are going to talk about it in more detail in the next lesson.

```
<androidx.constraintlayout.widget.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    <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>
```

activity_main.xml

Finally, we have a *src/main/java* folder, which contains Java source code divided by packages.

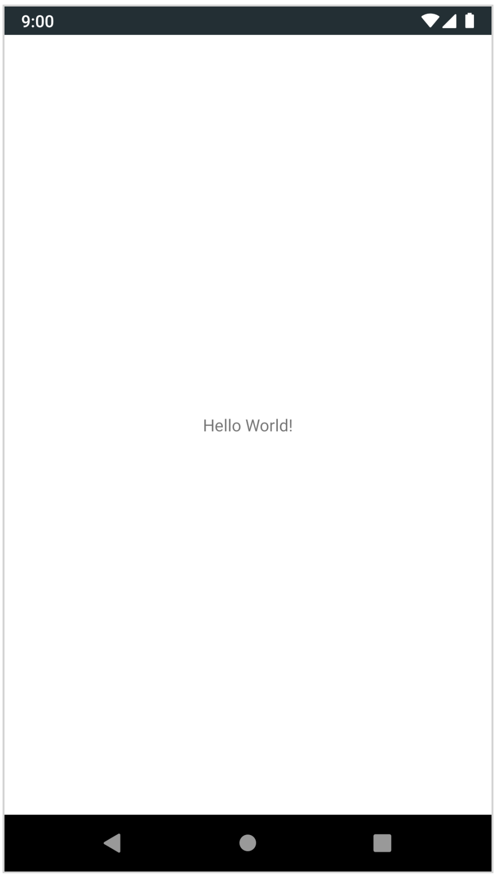
Currently, we only have one class file *MainActivity* which represents our main screen. We will talk about it in more detail in the next lesson.

```java
package com.travelblog;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

MainActivity.java

Android widget

In this lesson, we will use an **educative** Android widget to run and test the Android projects. You can navigate to see all the different files and their content.



Hello World!

Hit the *run* button and wait until the Android application is compiled and the emulator is started. The first time it may take up to 3 minutes to build the project, but all consecutive builds will be much faster.

After you see the *BUILD SUCCESSFUL* message in the terminal window, switch to the *Output* tab to see the Android emulator and the launched application.

Try to spend some time playing with the emulator.

> In case you make some changes to the code, save your changes and press *run* again to restart the application. Please note, it might take some time so you can observe the terminal tab to get an idea of the progress.

```
package com.travelblog;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

In the next lesson, we will cover commonly used Android libraries.