UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA

disim

Mobile Programming Laboratory

ANDROID
The First Application

Teachers

Ing. Tarquini Francesco, Ph.D

Ph.D in Computer Science Engineering

francesco.tarquini@univaq.it

Ing. D'Errico Leonardo

Ph.D Student in Computer Science Engineering

leonardo.derrico@graduate.univaq.it

Teaching Materials

Available on MOODLE platform

http://www.didattica.univaq.it

Google Drive Repository

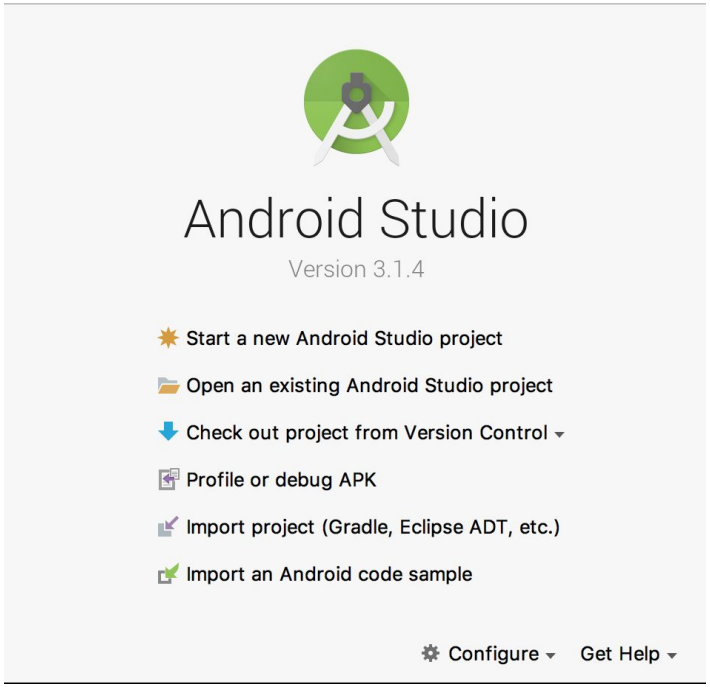
https://drive.google.com/drive/folders/1ISqZfn0i9Ub3eWNXbvW00rd0hD9ya8OL?usp=sharing

Topics

- Hello World
  - Packages
  - Android Version
  - Wizard
- Project
  - Activity
  - Layout
  - Resources
  - Manifest
  - Gradle Files

Hello World - Configuration

Run Android Studio and “Start a new Android Studio Project”.



The IDE gives us a wizard tool to create a new Project.

Hello World - Configuration

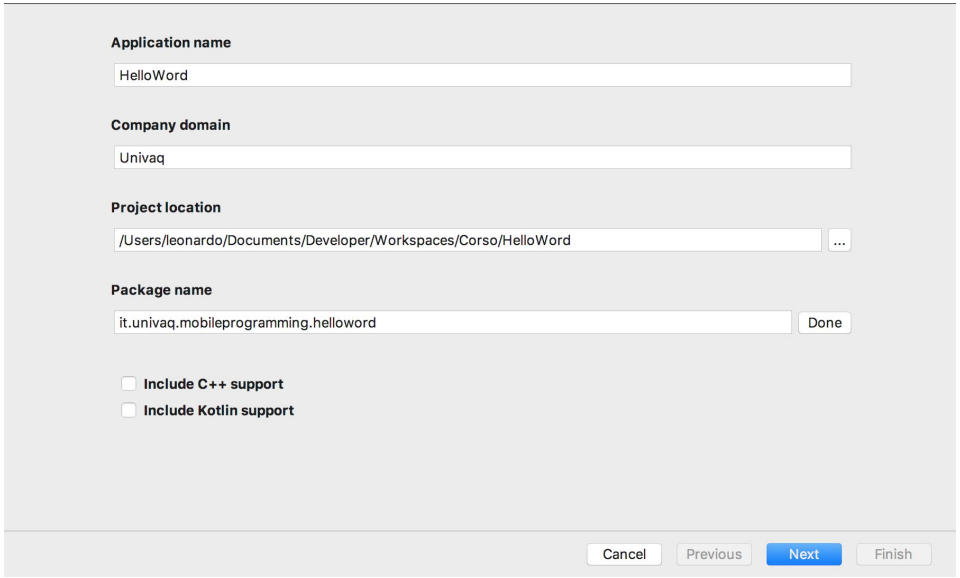
Application Name
application and folder (in hard drive) Name

Company domain
the name of the developer company

Project location
the complete path where store the files

Package Name
the name of the Java package and the application identifier in the world
[*suggestion*: use the most common pattern to define the package: **location.company.appname**]

Two flags
"**Include C++ support**" to include automatically the NDK to your project
"**include Kotlin support**" to write the application in Kotlin language



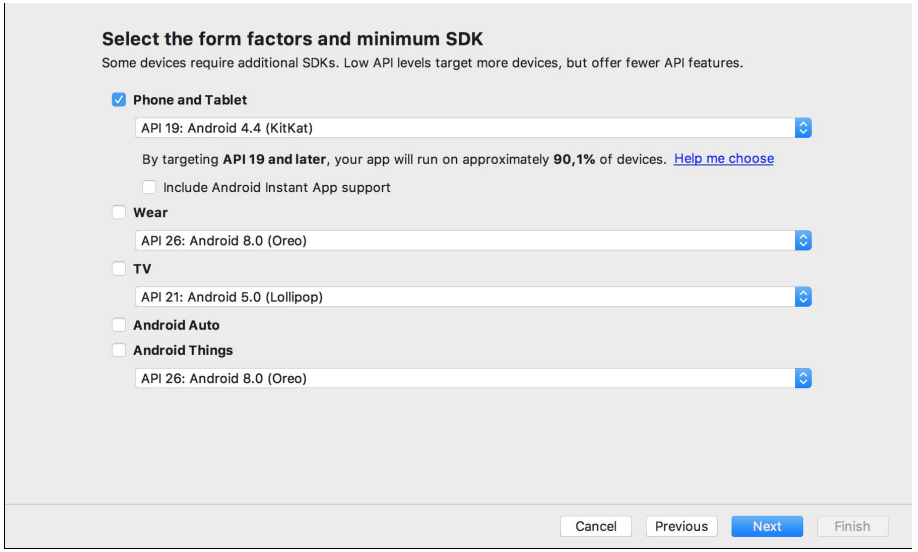
Hello World - Android Version

What do you want design?
More options, in this course we create an Phone and Tablet application

The target version is very important to define the application market share.

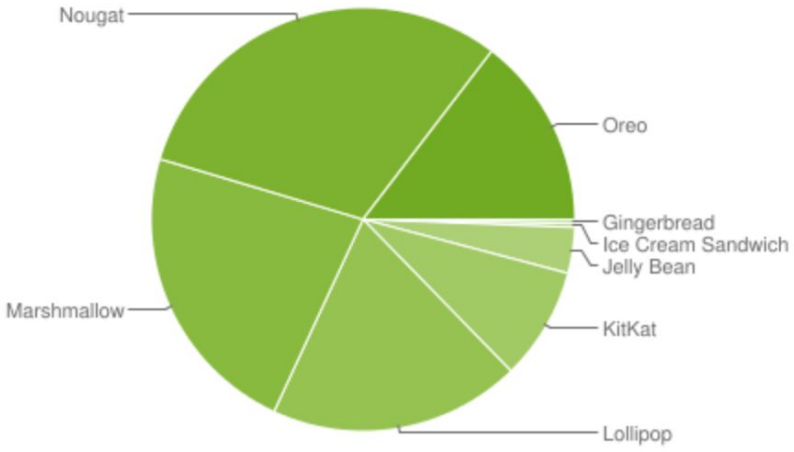
Using Android v4.4 our app will run on 90,1% of devices around the world.

Check Android Dashboard every month
https://developer.android.com/about/dashboards/index.html



Hello World - Android Version

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.1%
4.2.x		17	1.6%
4.3		18	0.5%
4.4	KitKat	19	7.8%
5.0	Lollipop	21	3.6%
5.1		22	14.7%
6.0	Marshmallow	23	21.6%
7.0	Nougat	24	19.0%
7.1		25	10.3%
8.0	Oreo	26	13.4%
8.1		27	5.8%



Semptember 28, 2018

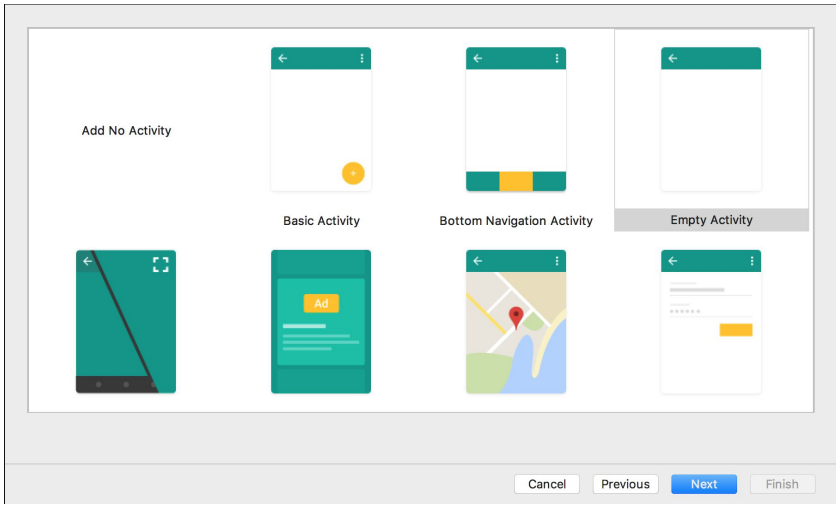
Hello World - Wizard

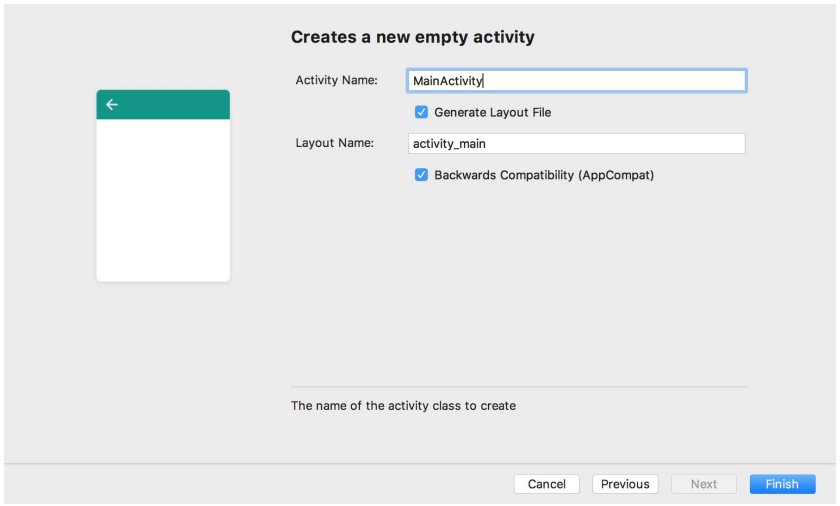
The wizard gives us the opportunity to start with a screen template

The developer has to define the name of:

- Activity Name: the Java file
- Layout Name: the layout file

The flag “**Backwards Compatibility**” permits to setup the project using the Support Library to extend the compatibility to the older versions of Android





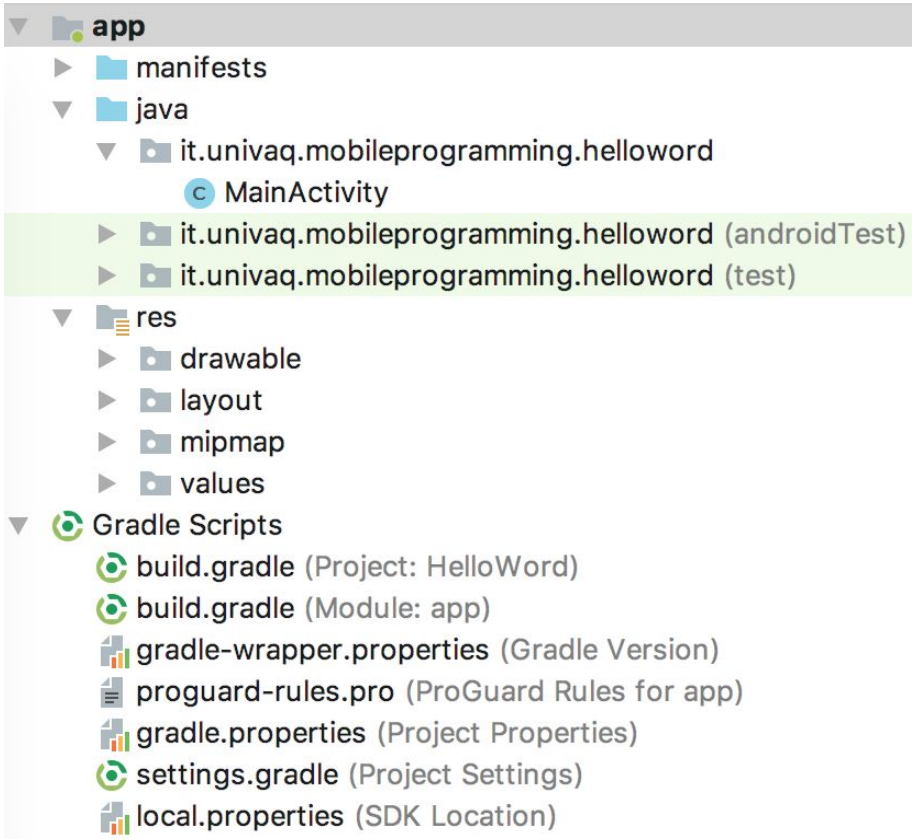
Project - Activity

Inside the “java” folder there are three times the package name of your application:

The developer can design the own Java application in the **only one** without annotation “androidTest” or “test”.

The packages with “test” annotations contain the Java class to create some mechanical test sessions.

For default the wizard create a Java class “**MainActivity.java**” that defines the logic of the first screen.



Project - Activity

What is an Activity?
In Android an Activity represents a screen of the application.

Every screen you create is a different Activity inside your application.

In the MVC Design Pattern the Activities are the Controller components.

The Class must **extends** the Android **Activity** Class or its children (for heritage), in this case **AppCompatActivity** (because we enabled the backward compatibility)

```
package it.univaq.mobileprogramming.helloword;

import android.support.v7.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);
        }
}
```

Project - Activity

Activities in the system are managed as an activity stack.

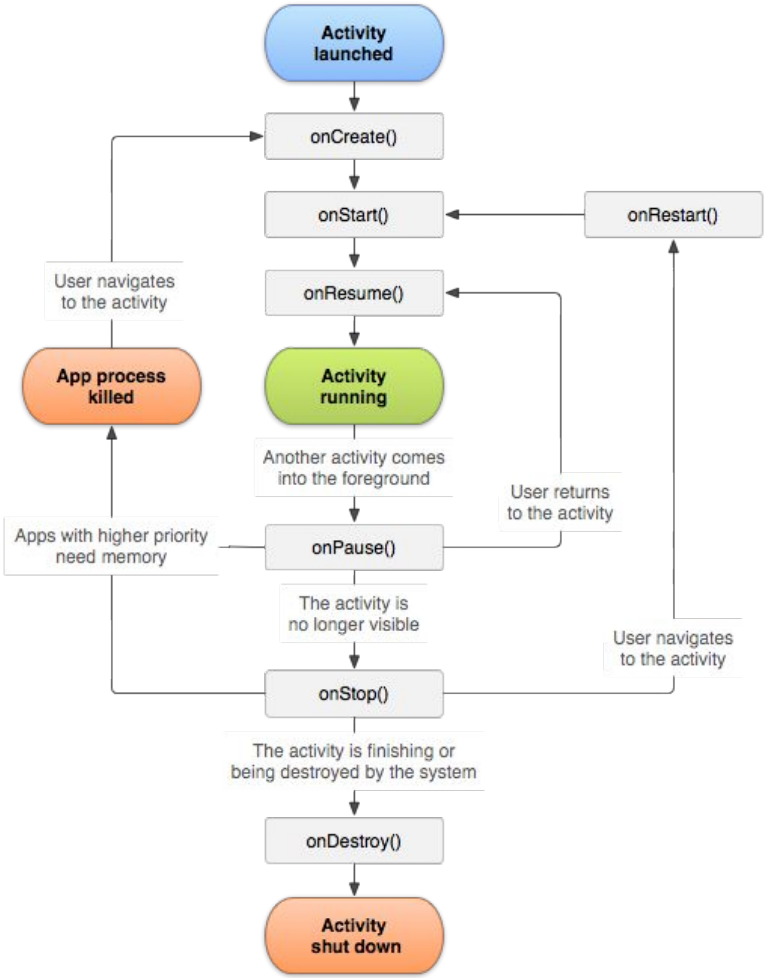
An activity has essentially four states:

If an activity is in the foreground of the screen (at the top of the stack), it is active or running.

If an activity has lost focus but is still visible, it is paused. A paused activity is completely alive.

If an activity is completely obscured by another activity, it is stopped.

If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, or simply killing its process. When it is displayed again to the user, it must be completely restarted and restored to its previous state.



Project - Activity

The start point where the developer can write the Java code is one of the Activity Callback:

- onCreate
- onStart
- onRestart
- onResume
- onPause
- onStop
- onDestroy

"*onCreate*" method usually is the main Java start point, where you can define the user interface of the screen, linking the pertinent XML layout file, using the method "*setContentView*"

```
package it.univaq.mobileprogramming.helloword;

import android.support.v7.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);
        }
}
```

Project - Fragment

Google introduced the Fragment starting from Android 3 Honeycomb.

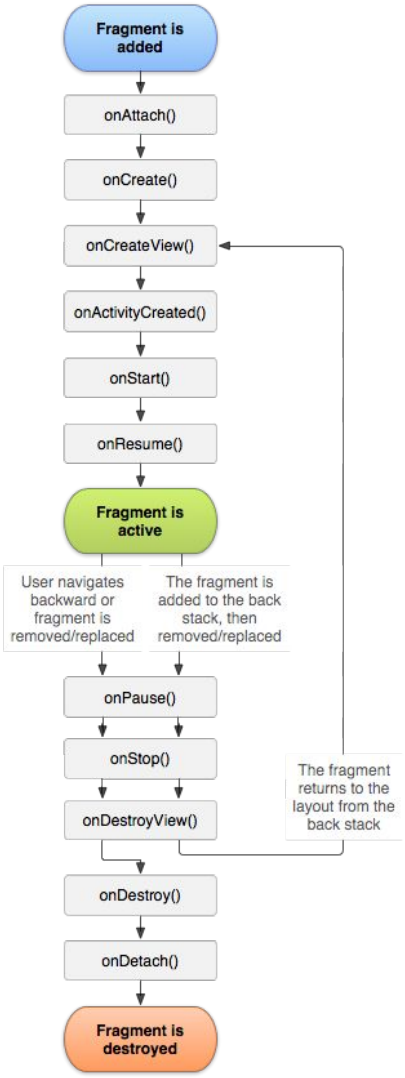
A Fragment is a part of the screen, thus a screen can have one or more fragments.

A Fragment has a own lifecycle more or less the Activity.

A Fragment is a Java class extending the Fragment Class or one of its children (heritage).

An Activity manages the fragments by a pertinent FragmentManager (or SupportFragmentManager if you are using the backward compatibility):

- add
- replace
- remove



Project - Fragment

How can you implement a Fragment?

Create a Class that **extends** Fragment Android Class

```
public class PlaceholderFragment extends Fragment {

        public PlaceholderFragment() {}

        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
                View rootView = inflater.inflate(R.layout.fragment_main_menu, container, false);
                return rootView;
        }
}
```

Manage the Fragment in Activity Class

```
getSupportFragmentManager()
    .beginTransaction()
    .add(R.id.container, new PlaceholderFragment())
    .commit();
```

Project - Layout

In Android every user interface is a resource and is defined in a specific XML file inside the folder "*layout*".

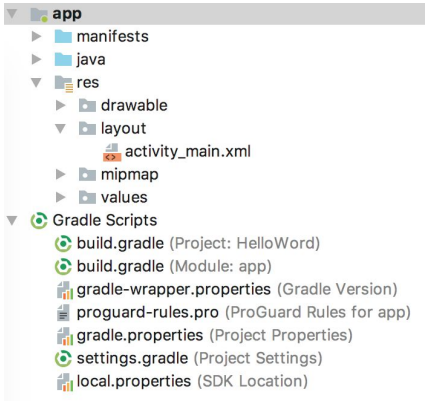
In our project, the file "activity_main.xml" is the user interface linked to MainActivity.java

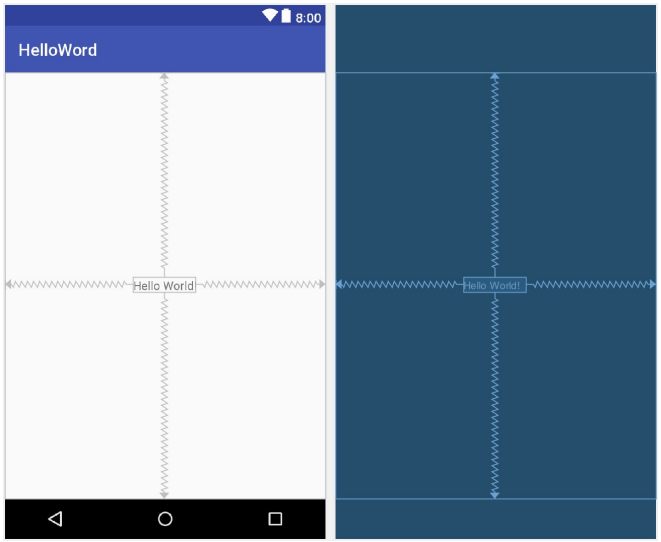
AndroidStudio gives us two ways to develop an interface:

Design: using the graphic tool

Text: writing in XML language

It is possible to create all components of the UI via Java language but usually is recommended to use the XML component.





Project - Layout

This is the code inside “activity_main.xml”.

It have two components:

- ConstraintLayout: a parent container
- TextView: a child view

Every component in a screen is represented by a specific Java Class that inherits from View Java Class.

All Views are divided in

- **Layouts**: FrameLayout, RelativeLayout, LinearLayout, ConstraintLayout, etc...
- **Containers**: ListView, RecyclerView, Toolbar, etc...
- **Views**: TextView, EditText, Button, ImageView, ImageButton, Switch, FloatingActionButton, etc…

In every screen exists a basic FrameLayout where is added the XML layout code.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">

  <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" />

</android.support.constraint.ConstraintLayout>
```

Project - Layout

Some roles
The first item in the screen must define the XML namespace linked to:
"http://schemas.android.com/apk/res/android"

Every item **MUST** declare its size in terms of **layout_width** and **layout_height**. Mainly you can use the follow values to define these field:

match_parent (**fill_content**): constant to resize the item as its parent

wrap_content: it means that the view is just big enough to enclose its content

device-independent pixel (**dip** or **dp**): a specific value

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">

  <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" />

</android.support.constraint.ConstraintLayout>
```

Project - Resources

In every Android project exists a **res** folder including all project resources.

Images, Layouts, Strings, Colors, Styles, Fonts, Audio, Video and so on.

Android Studio organizes the resources in a set of folder in according with the resources types:

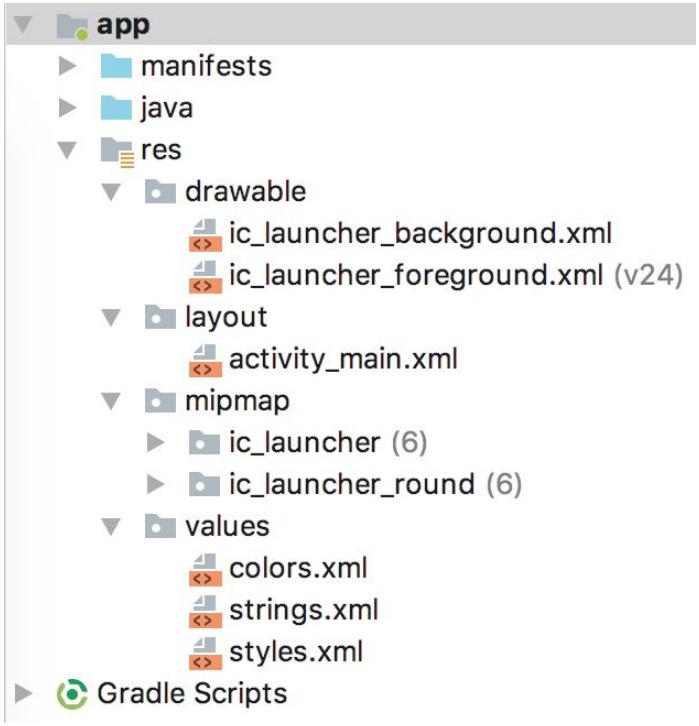
Drawable: images or everything is drawable as Shape

Layout: graphics screen description

Mipmap: the application icon

Values: application Strings, dimension values, colors and styles

Every resources can be qualified in according to the size, language, android version, location, and so on, of the device where the application runs.



Project - Manifest

The Manifest is an XML file that defines the map of the application and its configuration.

<application> tag is a index of every screen of the application and also defines the name of the application and its icon

<activity> tag represents a screen of the application

<intent-filter> define the scope

<action> specify in this case that this is the start point of the app

<category> specify that in system launcher is present the link to this app

Be Careful! Every Activity create **MUST** be declared in the Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
  package="it.univaq.mobileprogramming.helloword">

  <application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    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>
```

Project - Gradle Files

The build.gradle files define the app and AndroidStudio setup.

The build.gradle of the app module declares the configuration of:

- the min, target and compile SDK
- the version of your app
- the dependencies

The dependencies permits to include third library in our project.

The build.gladle of the project defines the version of the AndroidStudio, Gradle and Kotlin plugins.

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

android {
  compileSdkVersion 28
  defaultConfig {
     applicationId "it.univaq.mobileprogramming.helloword"
     minSdkVersion 19
     targetSdkVersion 28
     versionCode 1
     versionName "1.0"
     testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
  }
  buildTypes {
     release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
     }
  }
}

dependencies {
  implementation fileTree(dir: 'libs', include: ['*.jar'])
  implementation 'com.android.support:appcompat-v7:28.0.0-rc01'
  implementation 'com.android.support.constraint:constraint-layout:1.1.2'
  testImplementation 'junit:junit:4.12'
  androidTestImplementation 'com.android.support.test:runner:1.0.2'
  androidTestImplementation
'com.android.support.test.espresso:espresso-core:3.0.2'
}
```