

Android应用基础

陶煜波



课程目录



1

Android 简介

2

第一个Android App

3

视图, 布局和资源

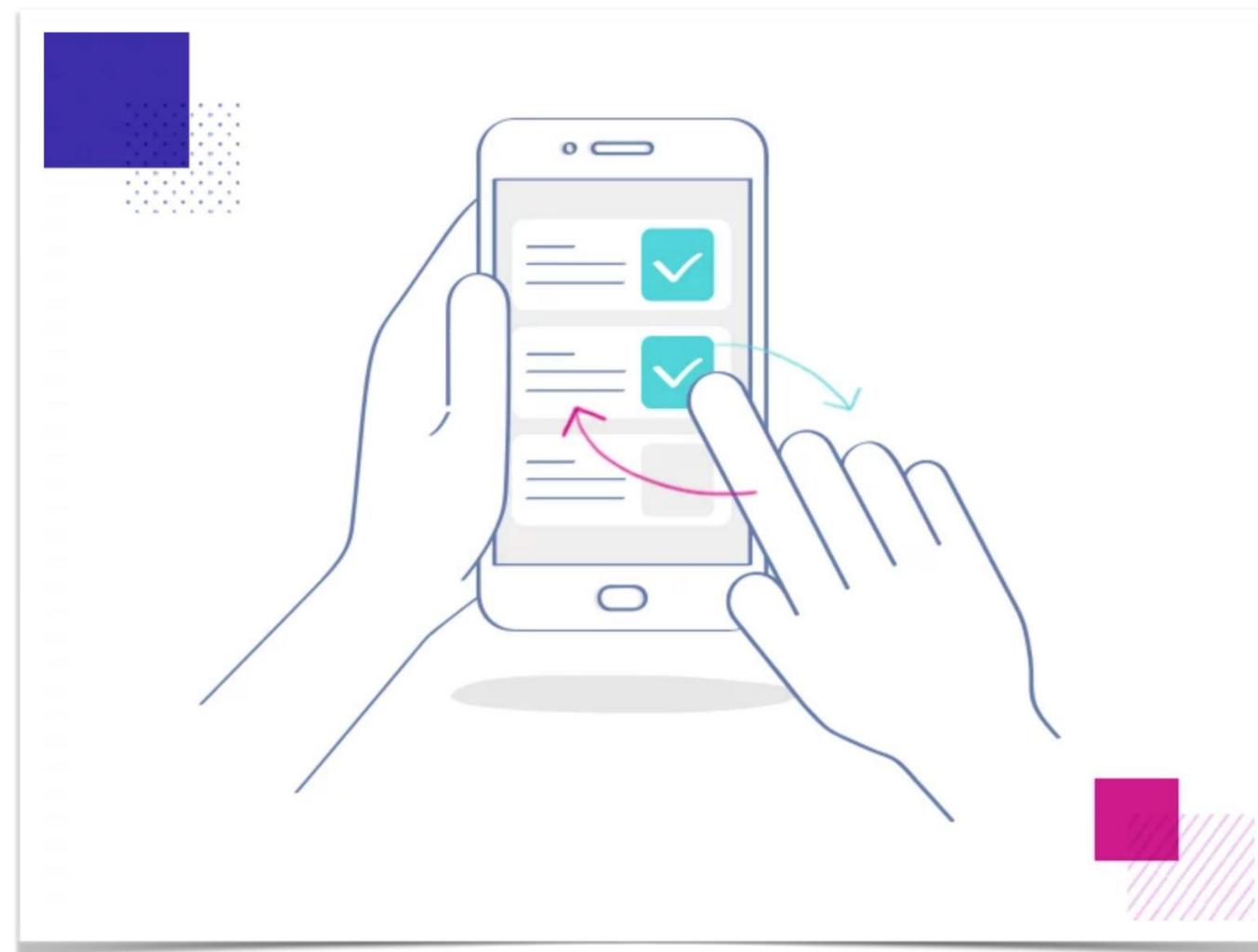
什么是Android

- Android是基于Linux内核的**移动操作系统**，主要设计用于智能手机和平板电脑等**触摸屏移动设备**
- Android是移动设备上最流行的操作系统。截止2024年Q4，Android占据了移动OS市场的**74%**份额；Google Play上Android应用数量为**168万**
- Android的特点是**高度可定制化**，**开源**



Android的交互方式

- 用户在屏幕上执行特定的输入，系统以**UI**的可见元素与用户进行**交互**
- **输入**：触摸手势（滑动，敲击，捏住等），支持字符，数字和表情符号的虚拟键盘，蓝牙，USB控制器等外围设备.....
- **输出**：菜单（选项式，上下文式，上下文操作栏式等），布局（堆栈式，网格式，相对式等）.....



Android的传感器

- Android系统集成多种类型的传感器，这些传感设备可以大致分为**硬件传感器**和**虚拟传感器**
- **硬件传感器**是内嵌在移动设备的真实物理传感器，直接测量环境数据，例如**加速度传感器**，**磁力计**
- **虚拟传感器**是从一个或多个硬件传感器中获取数据，例如**重力传感器**，**线性加速度传感器**



SensorSense App



Android 操作系统

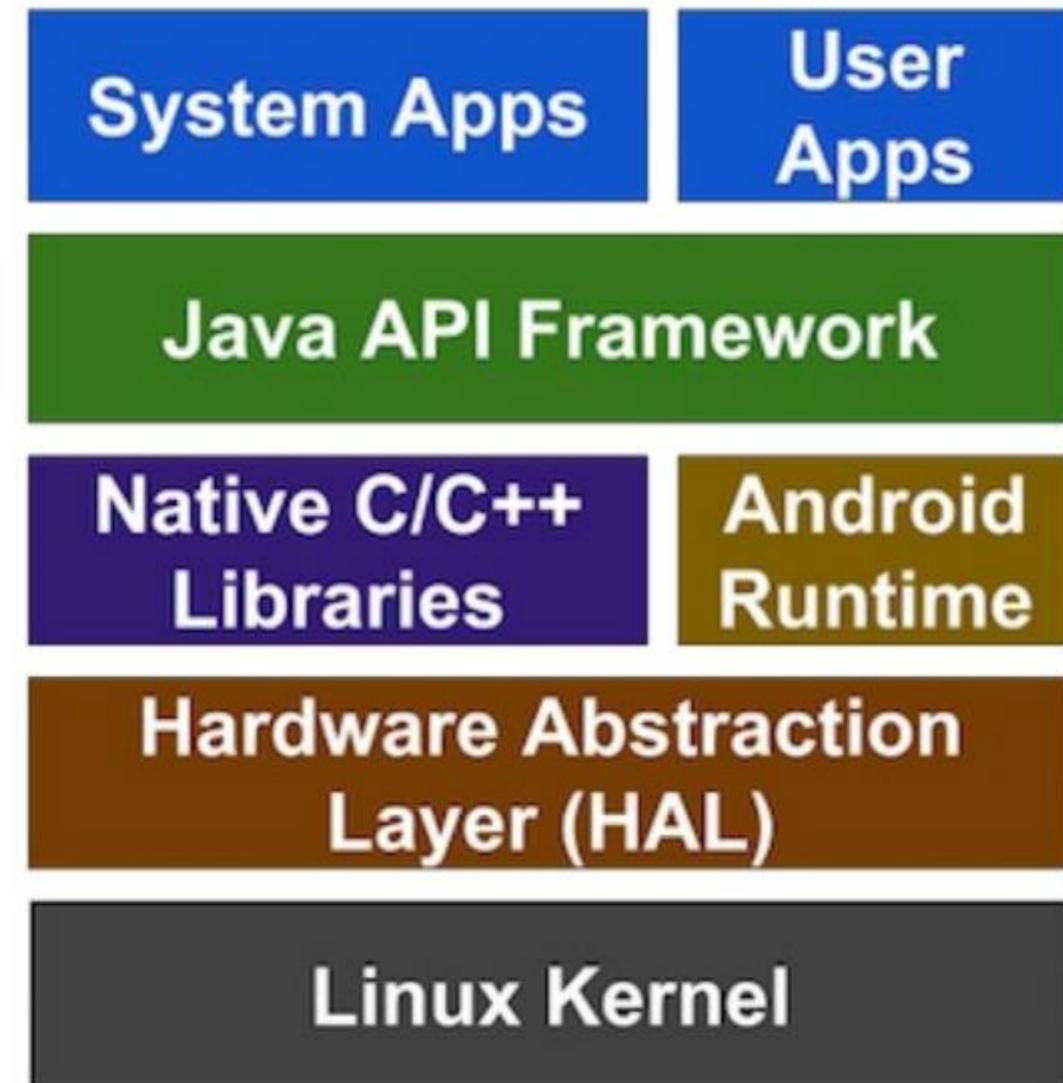
- Android操作系统是一种**多用户Linux系统**，每个应用都是一个不同的用户
- 默认情况下，每个应用都在其自己的**Linux进程内运行**

- 系统会为每个应用分配一个**唯一的用户ID**
- 系统为应用中的**所有文件设置权限**，使得只有分配给该应用的用户ID才能访问这些文件
- 每个进程都具有自己的**虚拟机 (VM)**，因此应用代码是在与其他应用隔离的环境中运行



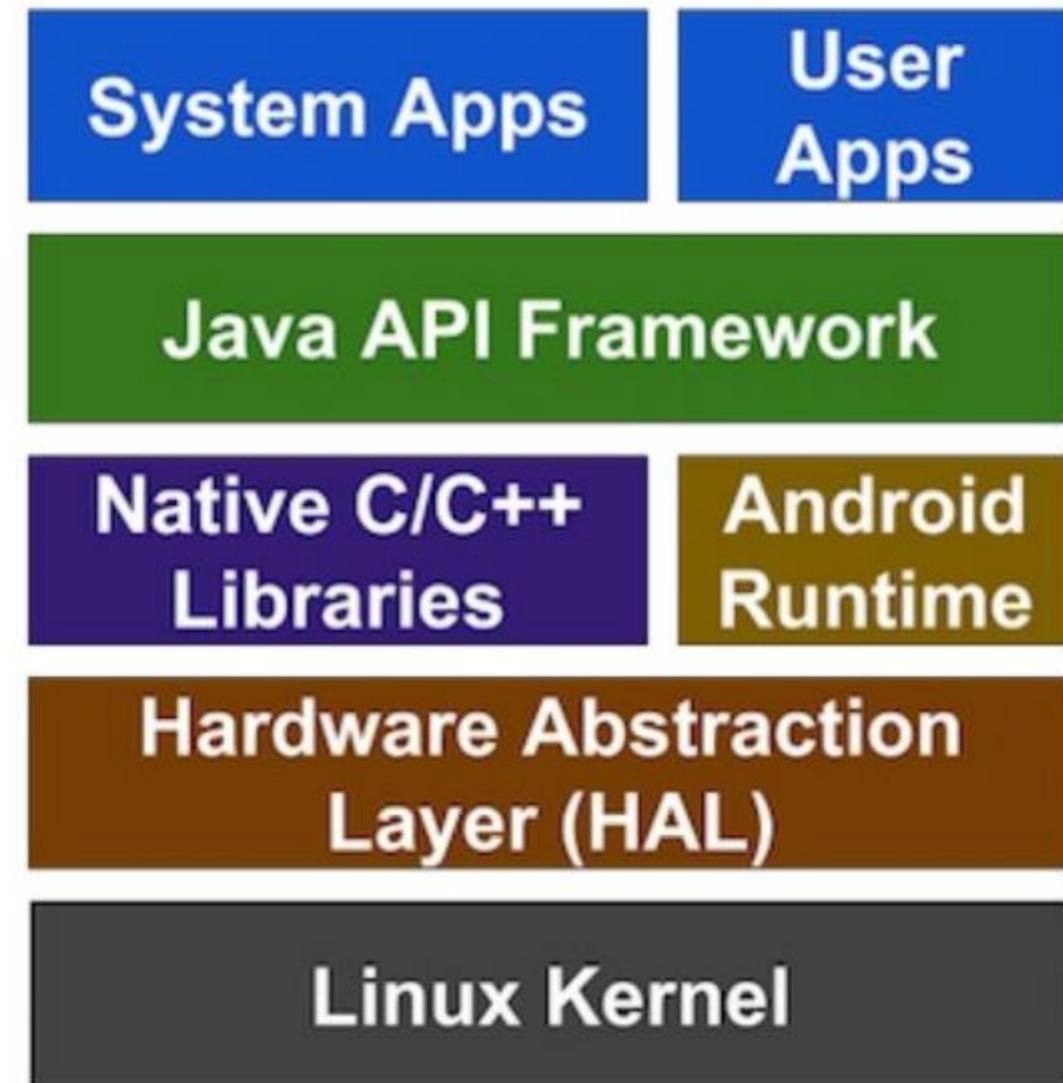
Android开发架构

- **Linux Kernel** : Android平台的基础是**Linux内核**。上层依赖于Linux内核的底层功能，如线程和low-level的内存管理。使用Linux内核使Android能够天然具备一些安全特性，同时设备制造商可以基于开源的Linux内核来开发他们的驱动程序



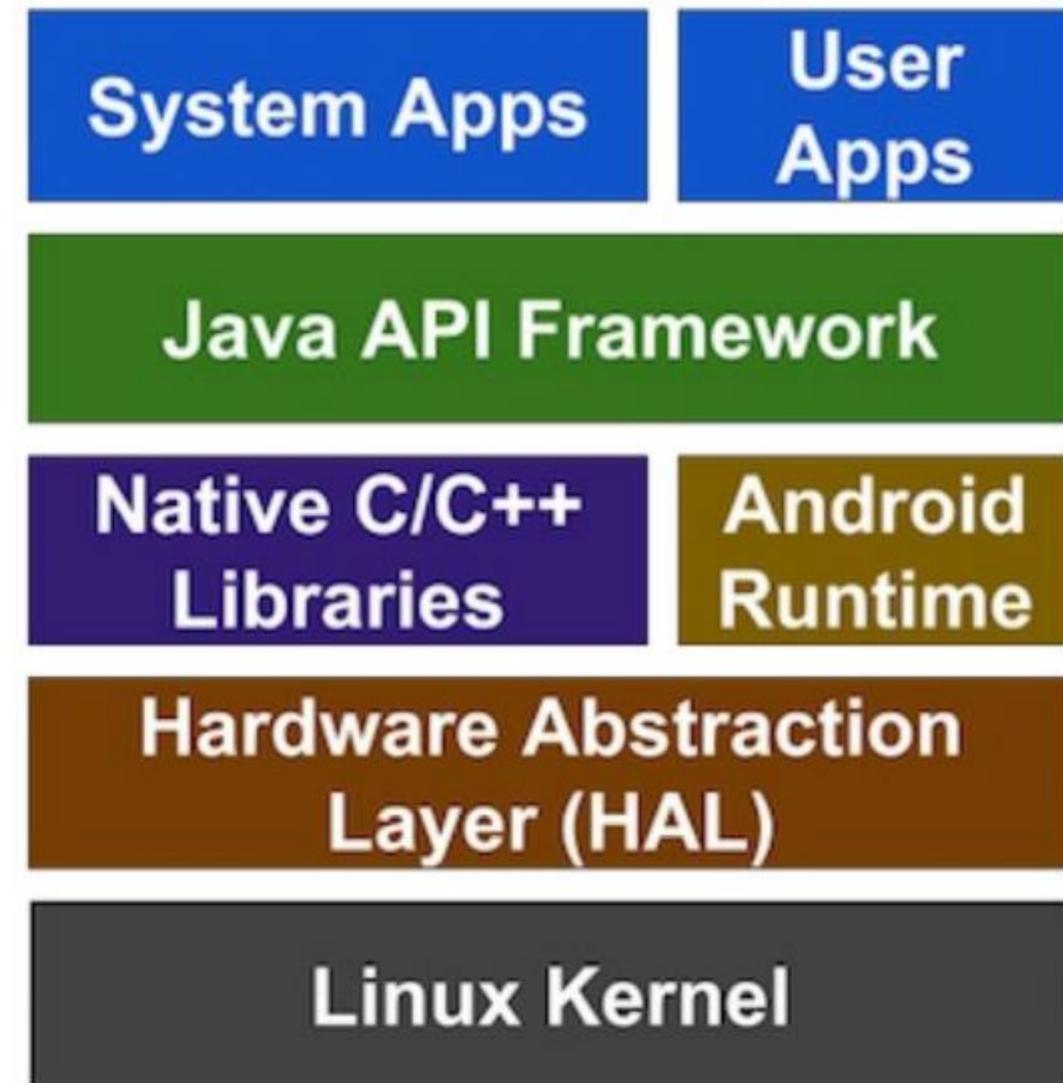
Android开发架构

- **HAL**：该层提供标准接口，可将设备硬件功能提供给更高级别的**Java/Kotlin API**框架。HAL由多个库模块组成，每个库模块实现特定类型硬件组件的接口，例如摄像头，蓝牙模块



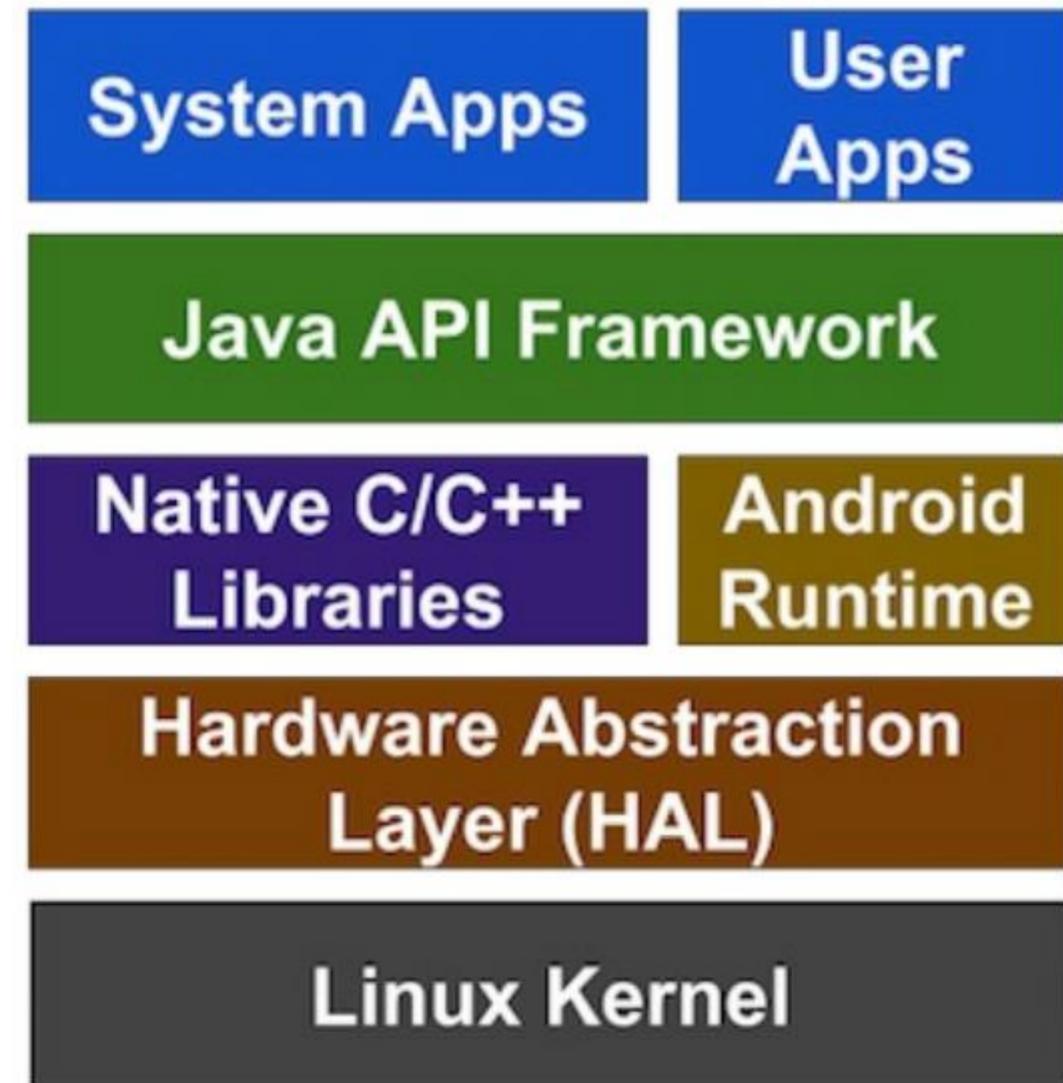
Android开发架构

- **Native C/C++ Libraries** : 很多核心的Android系统组件和服务都是用C和C++编写。这些库开放Java/Kotlin API提供给应用程序



Android开发架构

- **Android Runtime** : 每个应用程序都在自己的进程中运行，并拥有自己的Android Runtime实例
- **Java/Kotlin API Framework** : 通过Java/Kotlin语言编写的应用程序编程接口 (API)
- **Apps** : 用户应用程序和系统应用程序 (电子邮件，短信，日历，Internet浏览) 都位于此级别



Android 应用

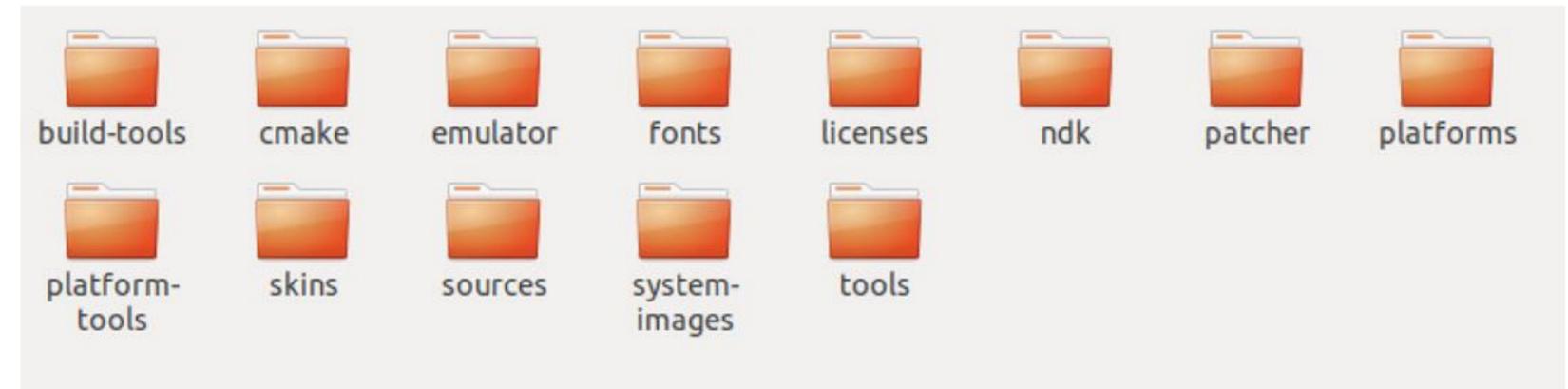
- Android 应用采用 **Java/Kotlin** 编程语言编写
- Android SDK 工具将代码连同所有数据和资源文件编译到个 **.apk** 后缀的文件中
- apk文件是Android 系统内**安装包程序**



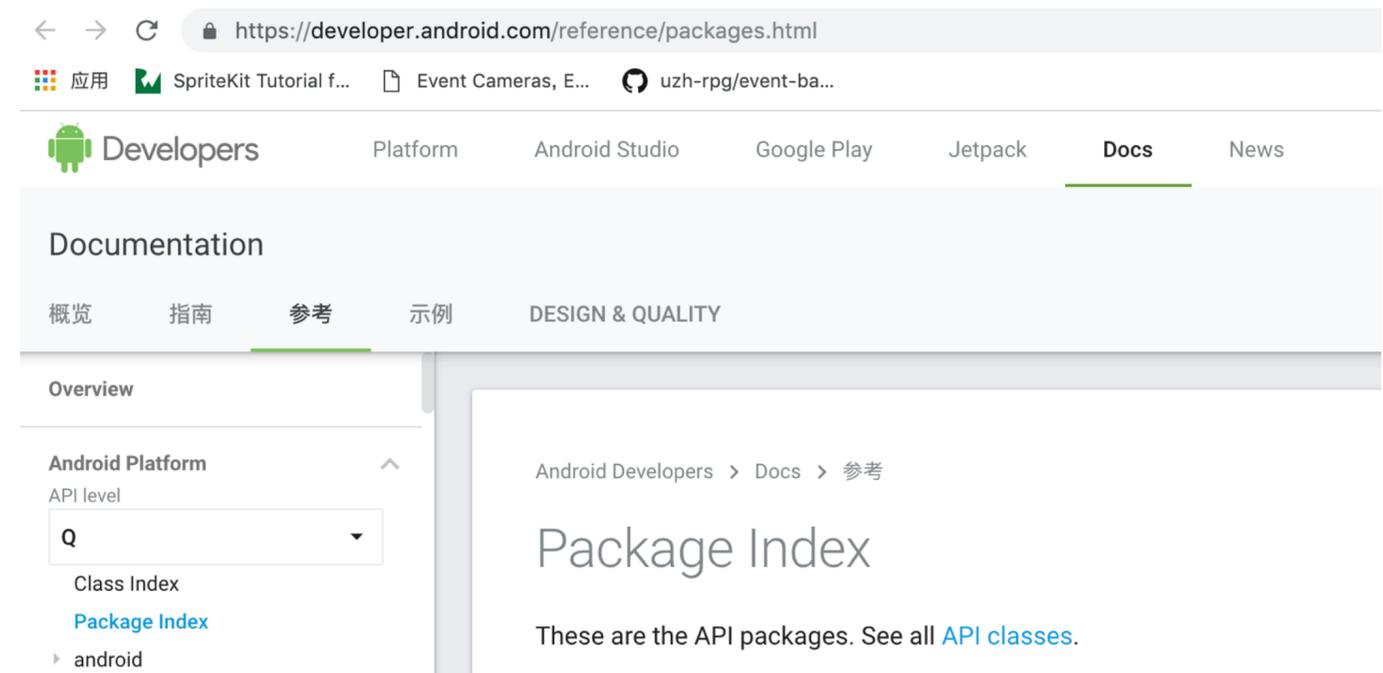
Android Software Development



- **Android software development(SDK)** 是Google提供的Android开发工具包（调试器，监视器，编辑器），在开发Android程序时，我们需要通过引入该工具包来使用Android相关的API
- Android SDK官方文档地址：developers.android.com/docs

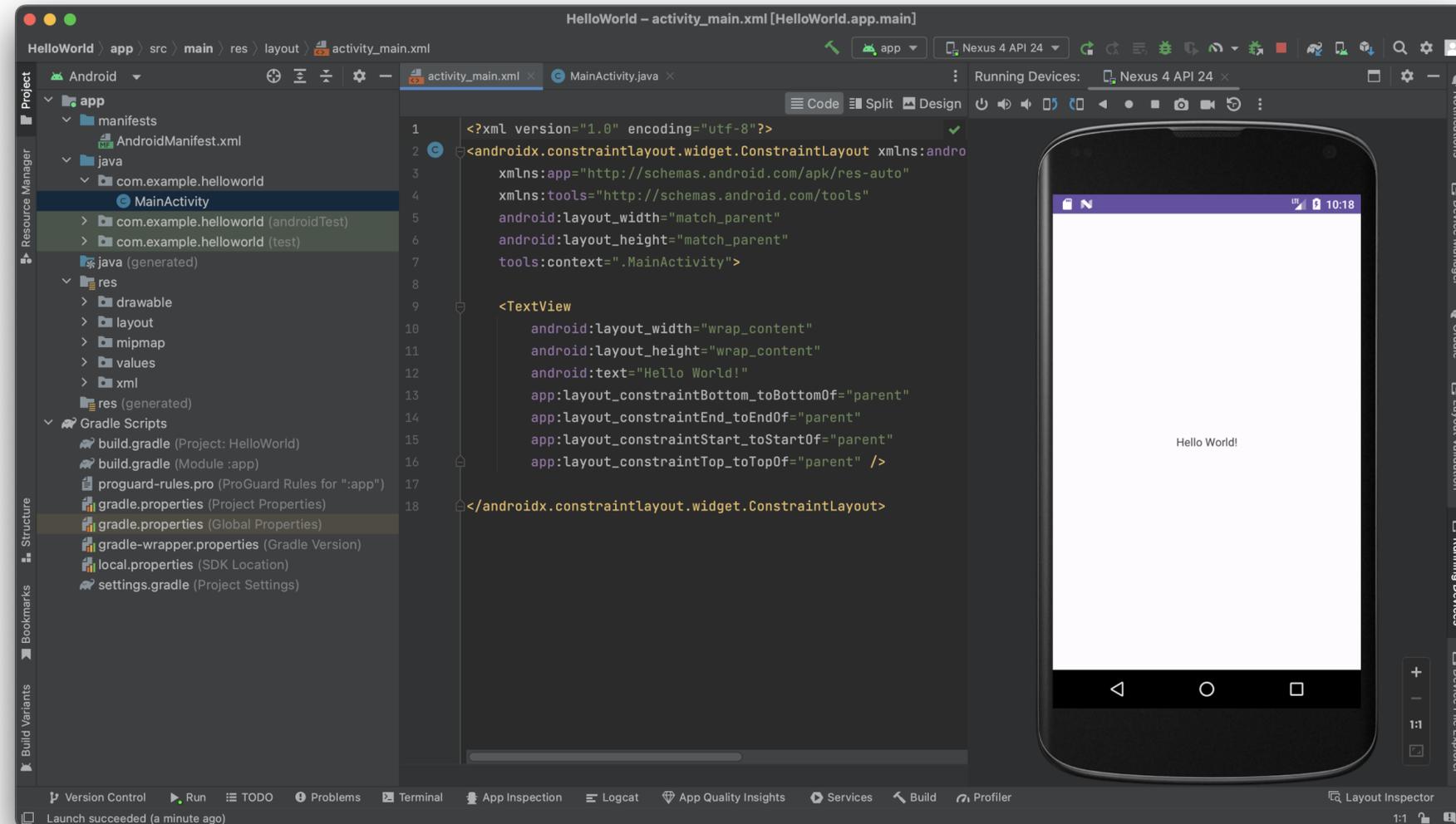


Android SDK 解压后包含的文件、文件夹



Android Studio

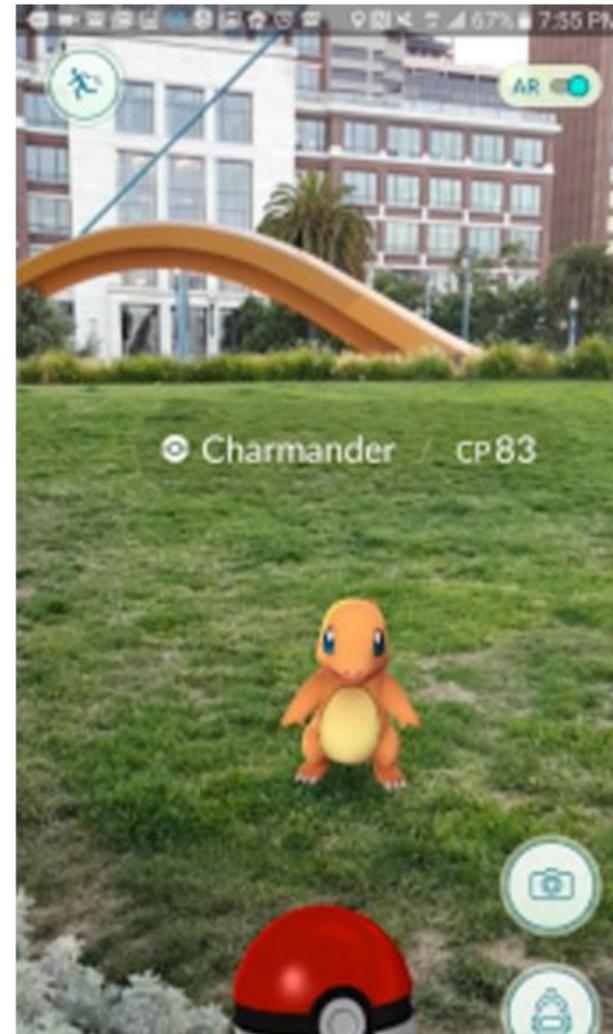
- **Android Studio**是用于Google Android操作系统的官方集成开发环境（IDE），基于JetBrains的IntelliJ IDEA软件构建，并且专门为Android开发而设计
- Android Studio用于开发、运行、调试、测试和打包应用程序，包括监视器和性能工具模拟器，可视化布局编辑器



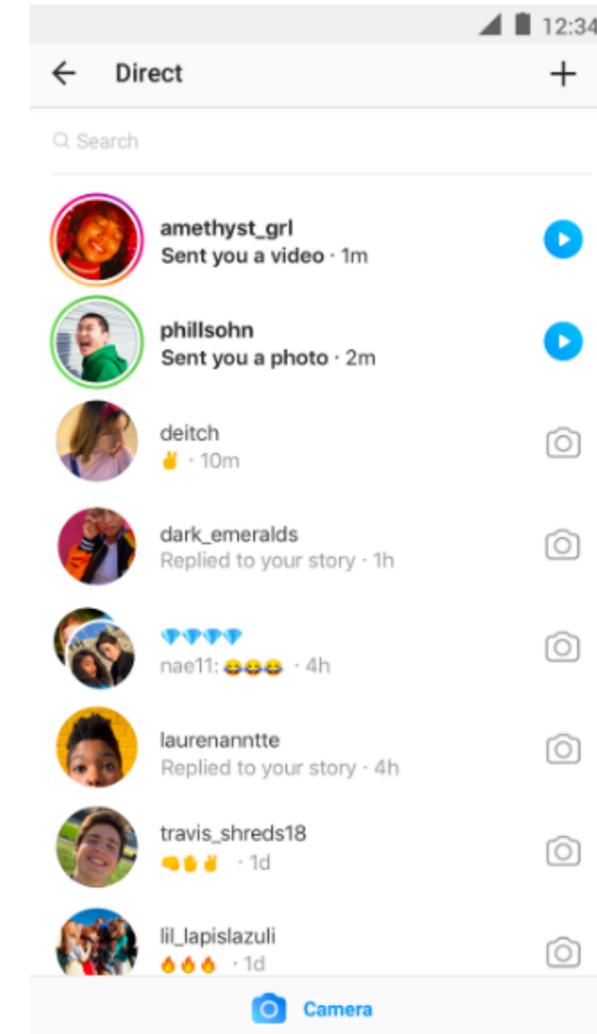
Some Android Apps



Wechat



Pokemon GO



Instagram



课程目录



1

Android 简介

2

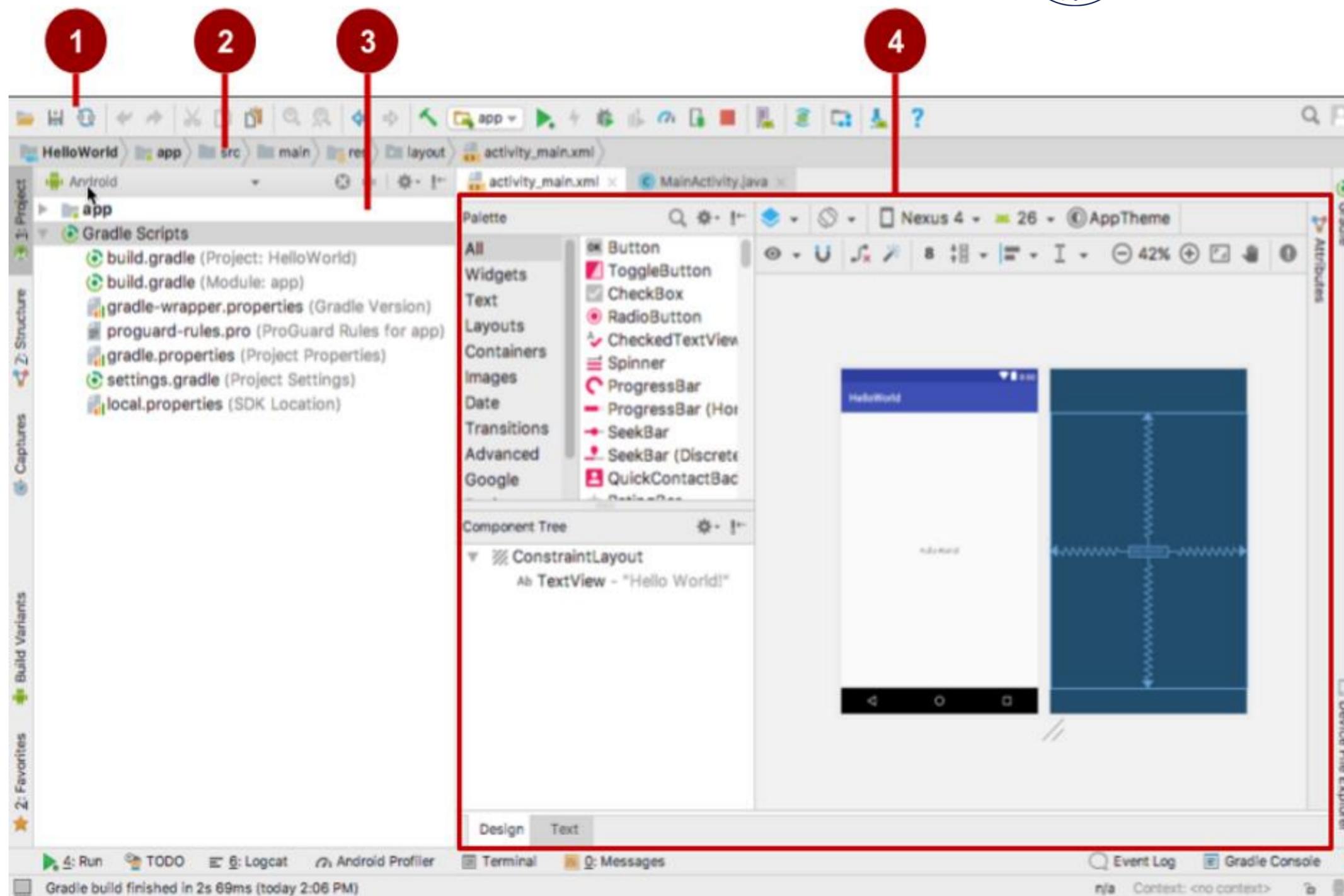
第一个Android App

3

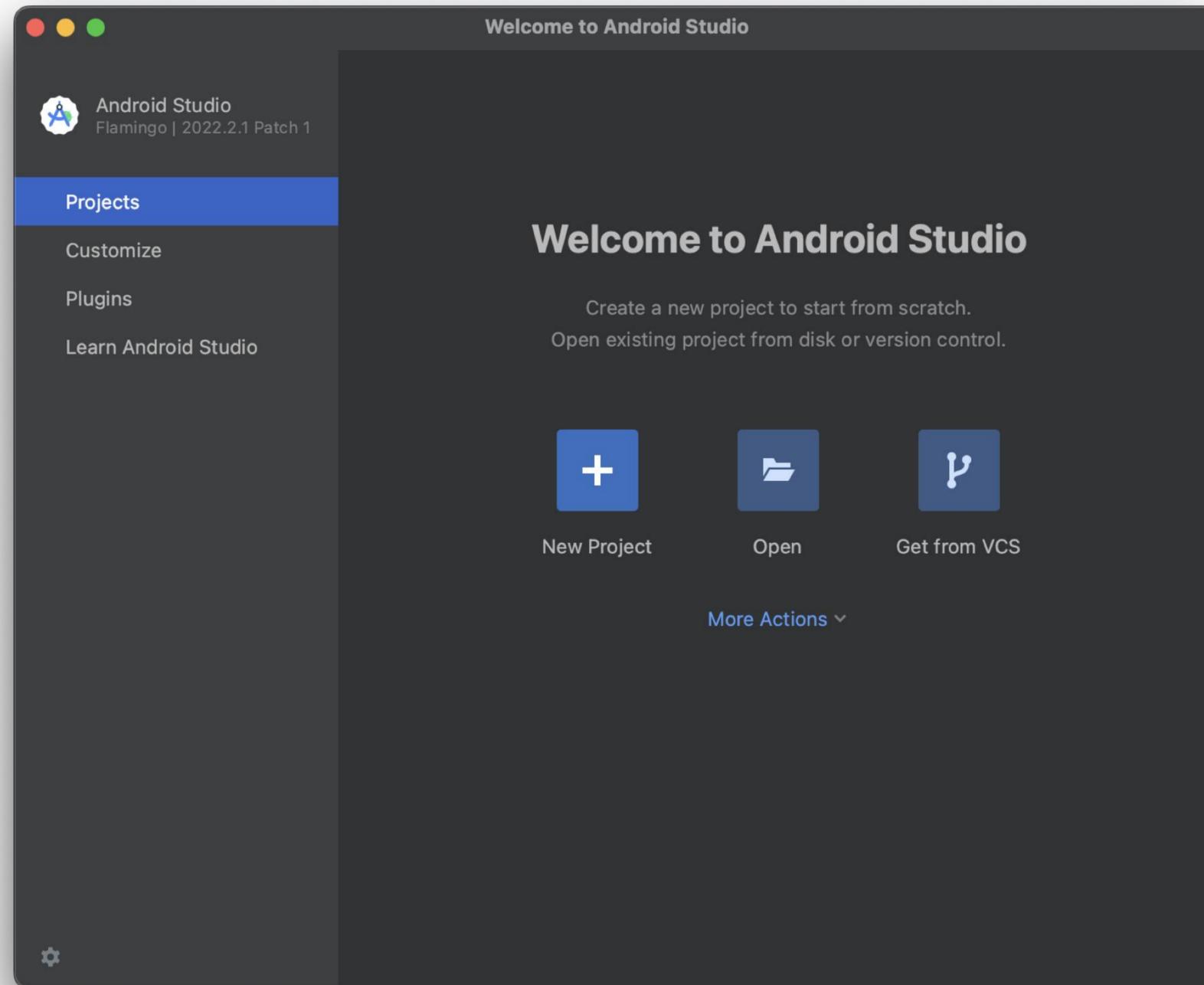
视图, 布局和资源

Android Studio界面

- ① 工具栏
- ② 导航栏
- ③ 项目面板
- ④ 编辑器
- ⑤ 其他面板的标签页

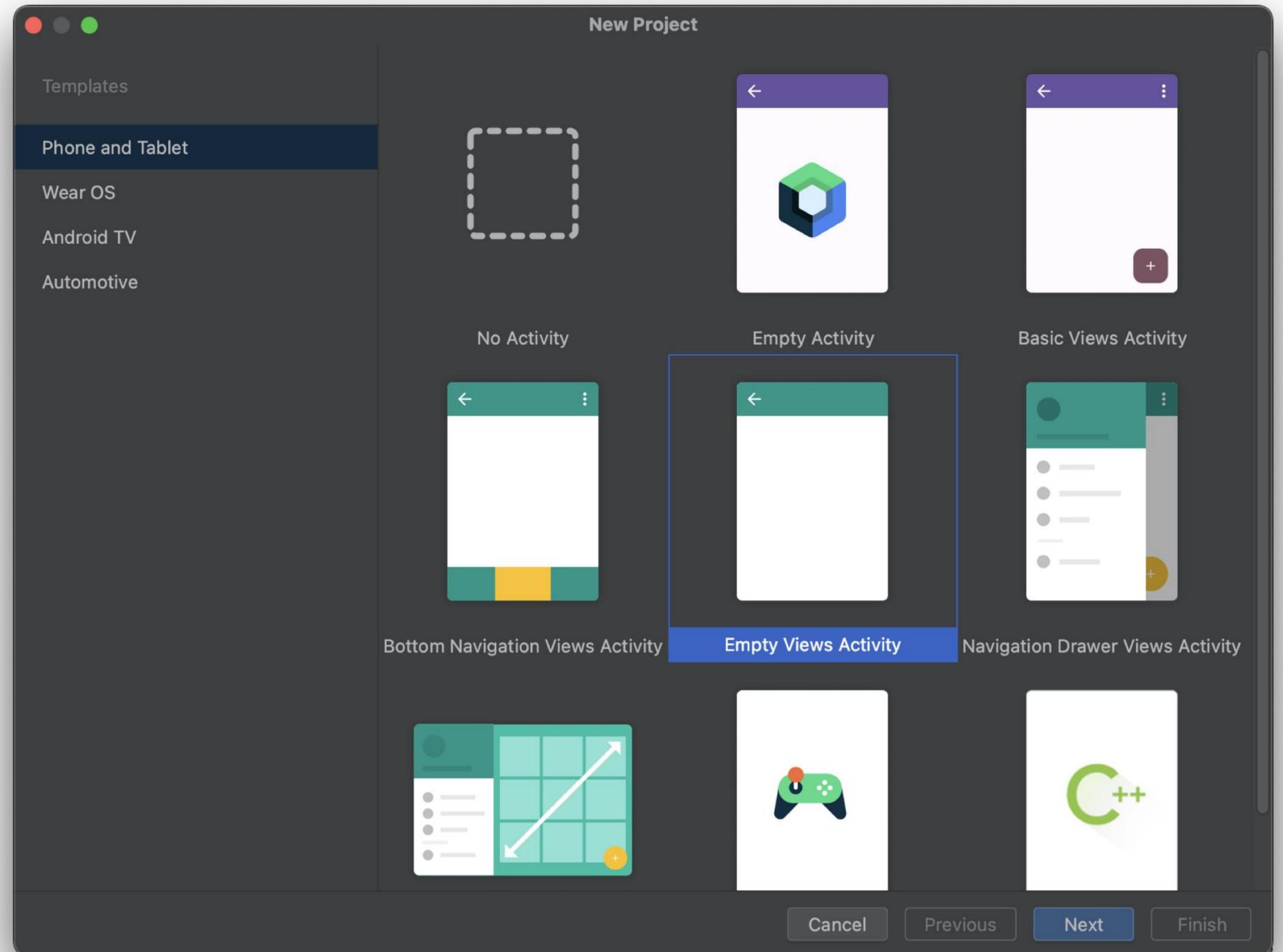


打开Android Studio



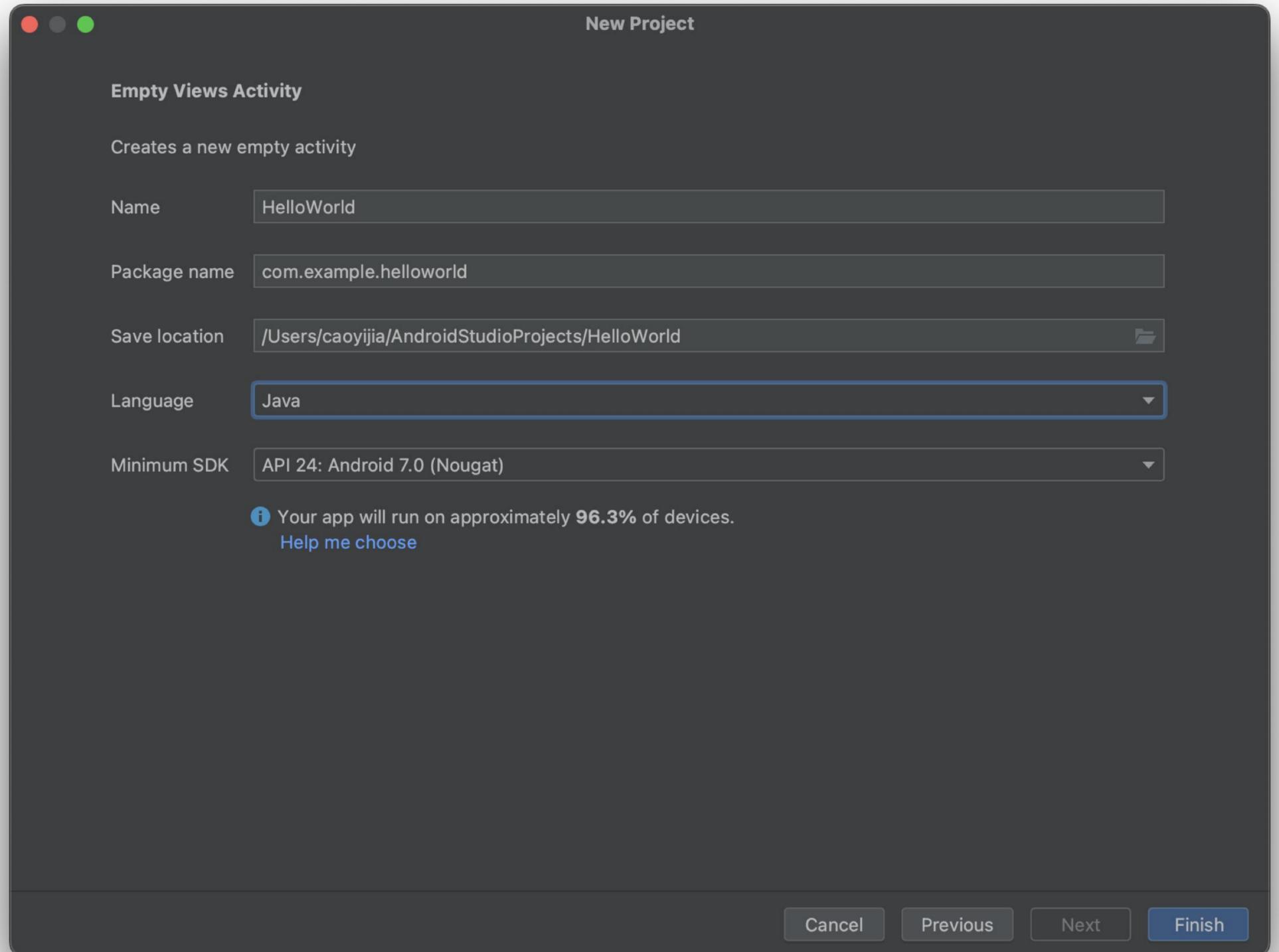
选择项目类型

- **Activity模板选择界面**。因为是一个简单的HelloWorld项目，所以这里我们选择 **Empty Activity** 或者 **Basic Activity**



输入项目信息

- **Name** : 输入工程名称
HelloWorld
- **Language**: 选择Java
- **Minimal API Level** : 保持
默认的API 24



New Project

Empty Views Activity

Creates a new empty activity

Name HelloWorld

Package name com.example.helloworld

Save location /Users/caoyijia/AndroidStudioProjects/HelloWorld

Language Java

Minimum SDK API 24: Android 7.0 (Nougat)

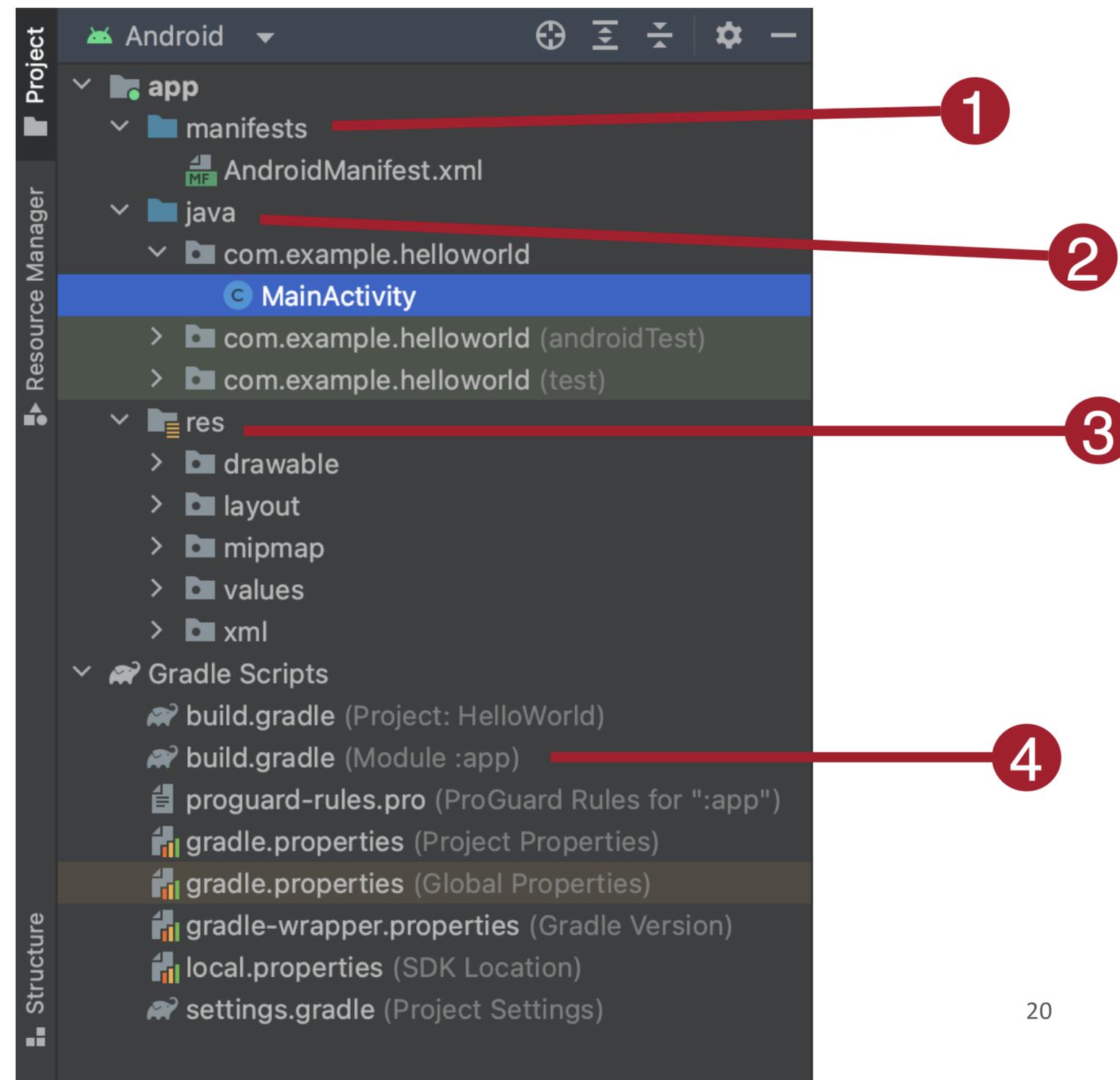
i Your app will run on approximately 96.3% of devices.
[Help me choose](#)

Cancel Previous Next Finish



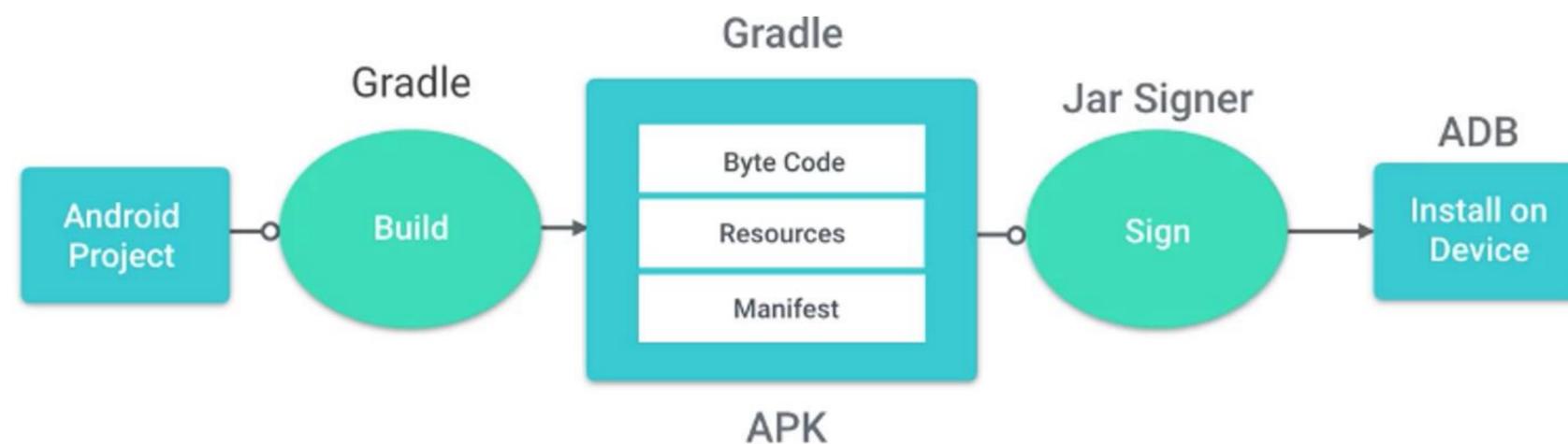
项目文件夹

- 1 manifests—**清单文件** (Manifest file)
- 2 java—Java 或 Kotlin**源代码**
- 3 res—**资源** (格式为XML) – 布局, 字符串, 图片, 颜色
- 4 build.gradle—Gradle **构建文件**



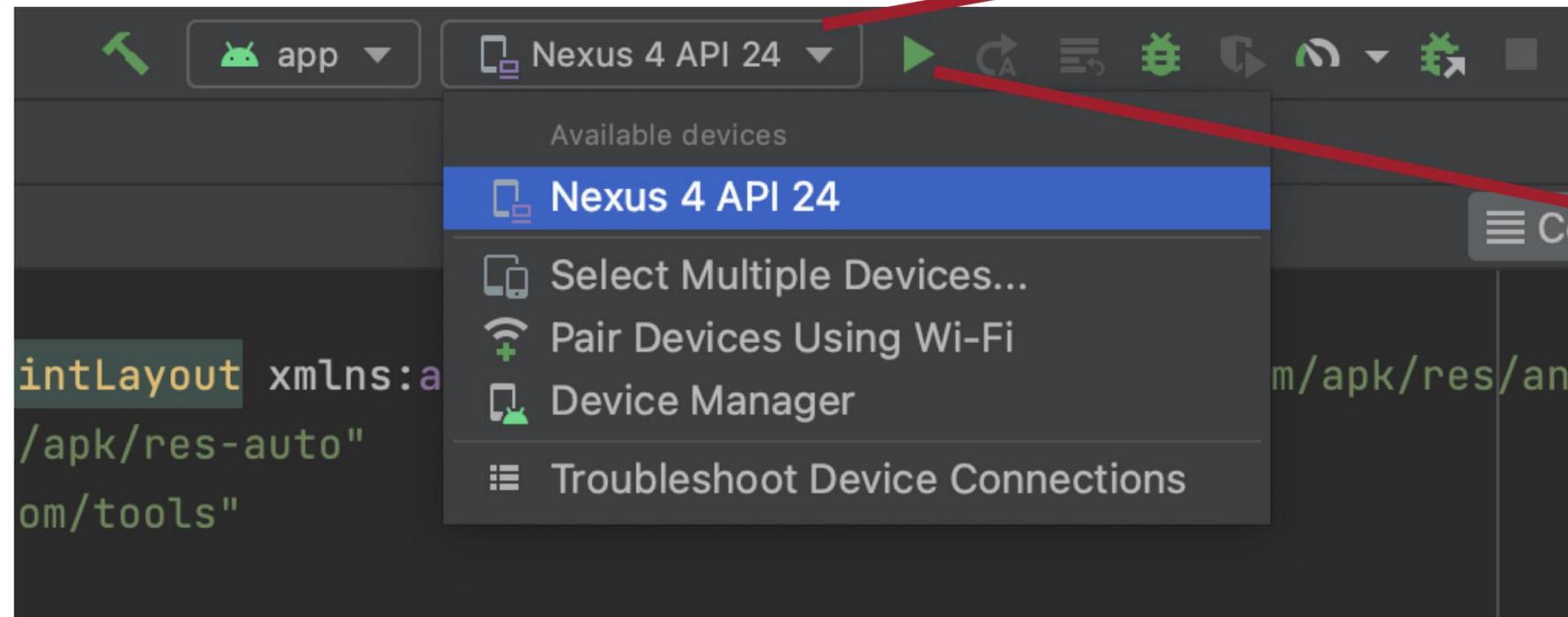
Gradle

- Gradle是用于多语言软件开发的**构建自动化工具**。它控制从**编译和打包到测试，部署和发布**的任务中的开发过程。支持的语言包括Java（Kotlin, Groovy, Scala），C / C ++和JavaScript
- 一个安卓项目中有三个.gradle文件：
`build.gradle(Project)`，
`build.gradle(Module)`， `settings.gradle`



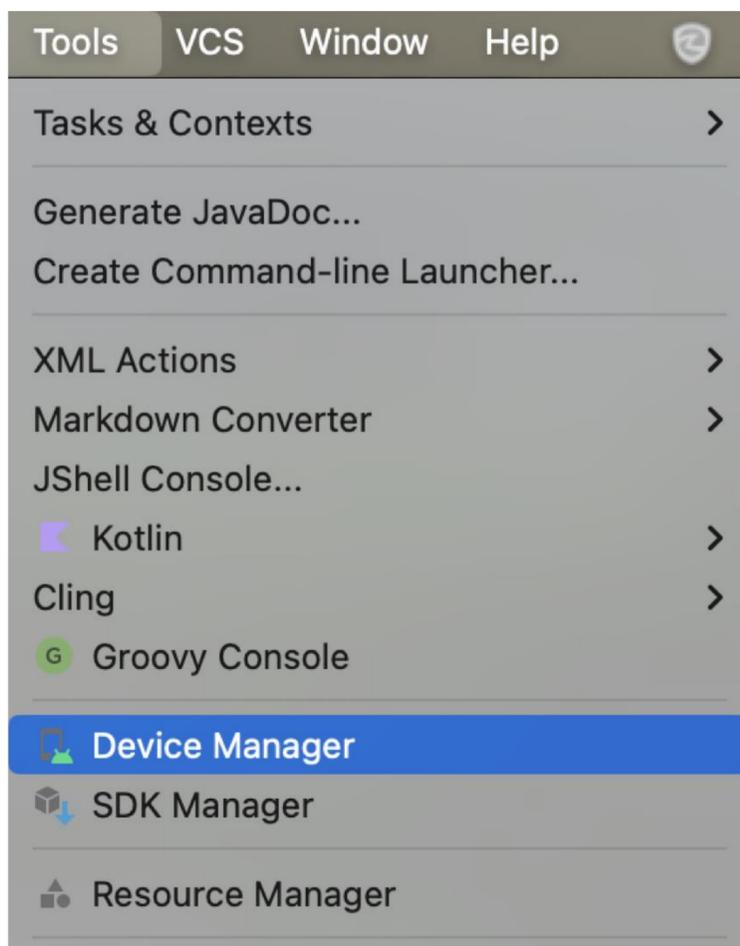
运行项目

- 1 运行工程
- 2 选择虚拟设备或者实体设备
- 3 点击OK

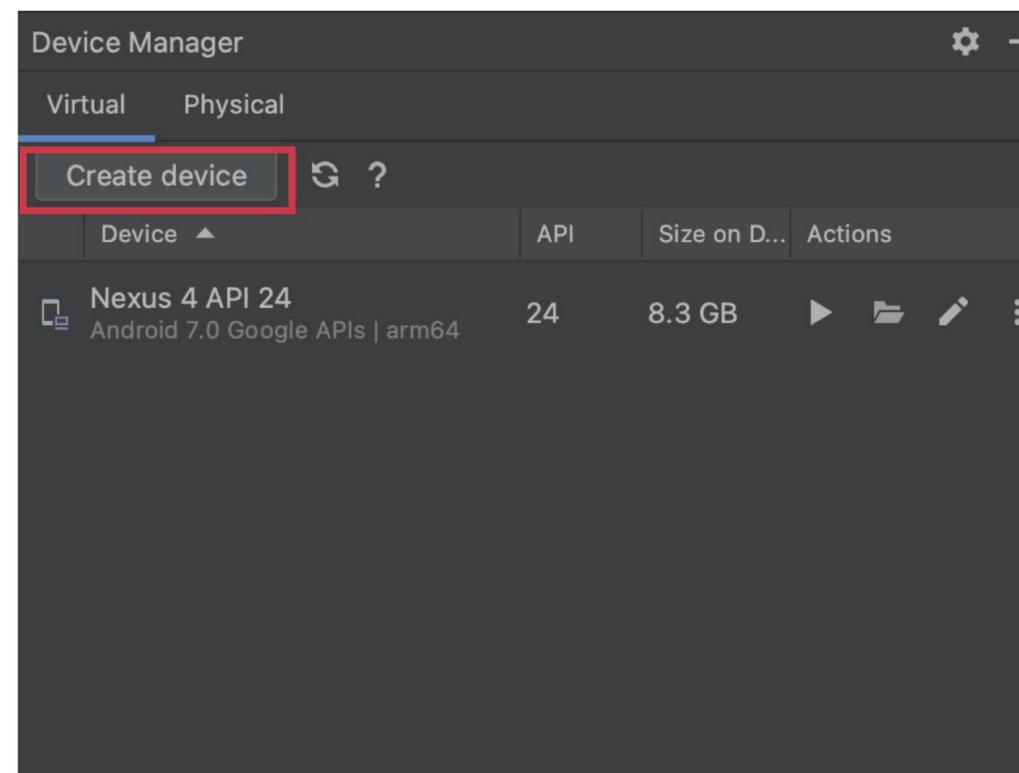


创建虚拟设备

- 点击“Tools > **Device Manager**”进入设备管理界面

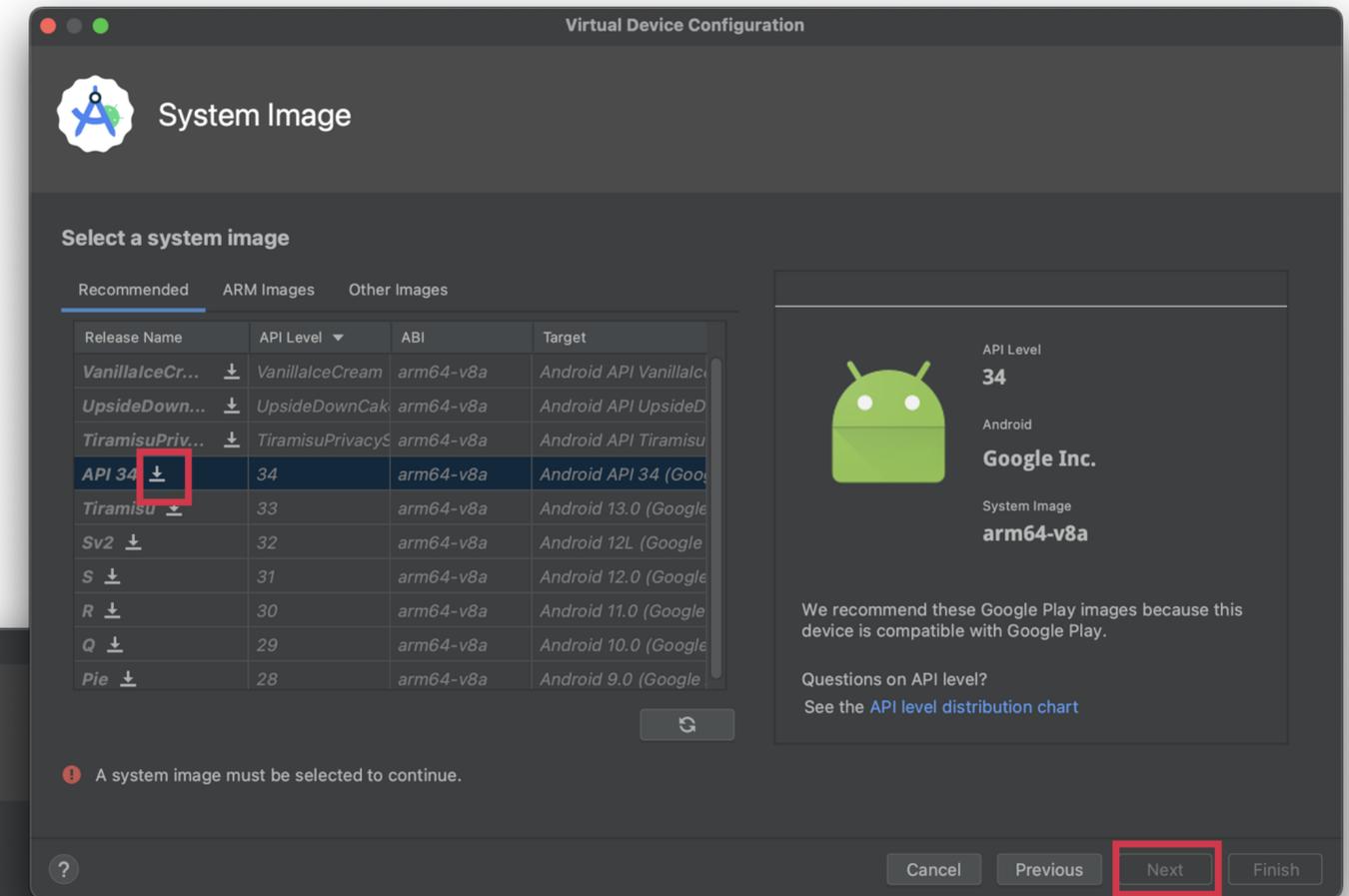
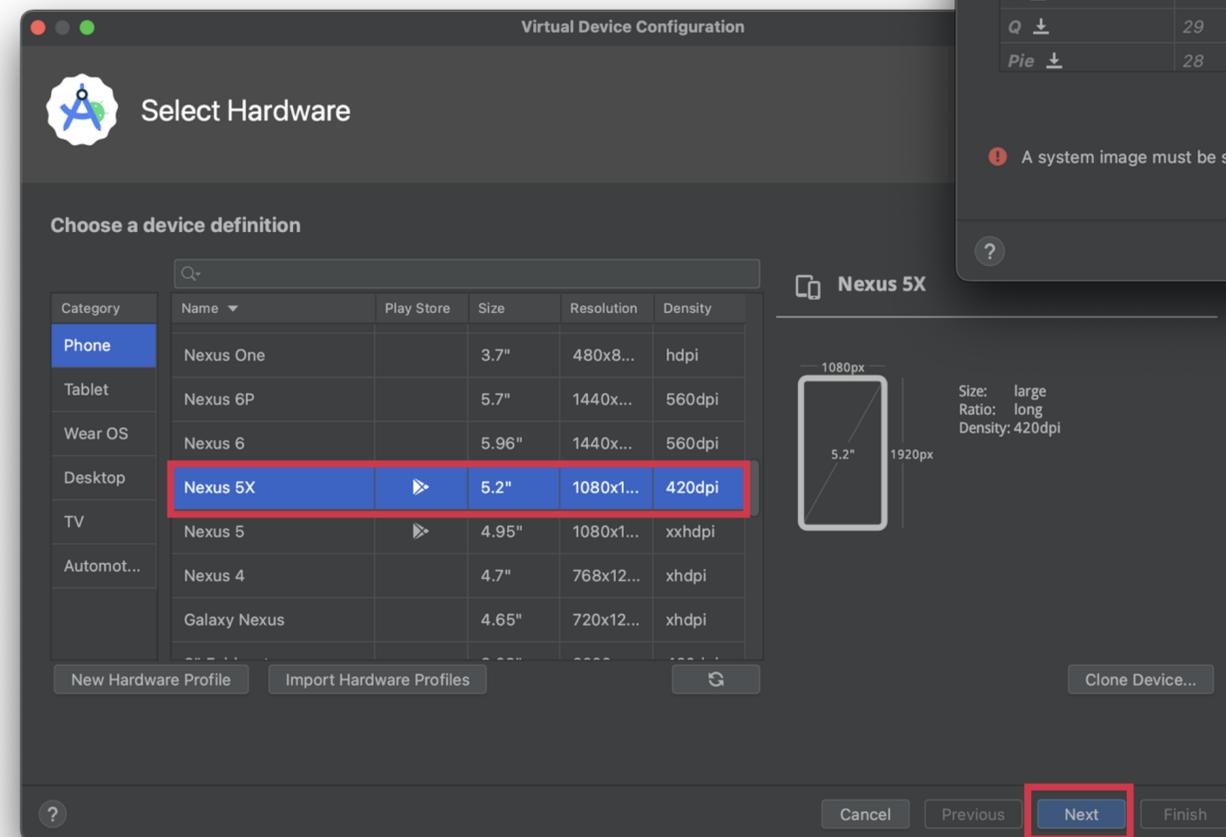


- 点击“**Create device**”创建新的虚拟设备

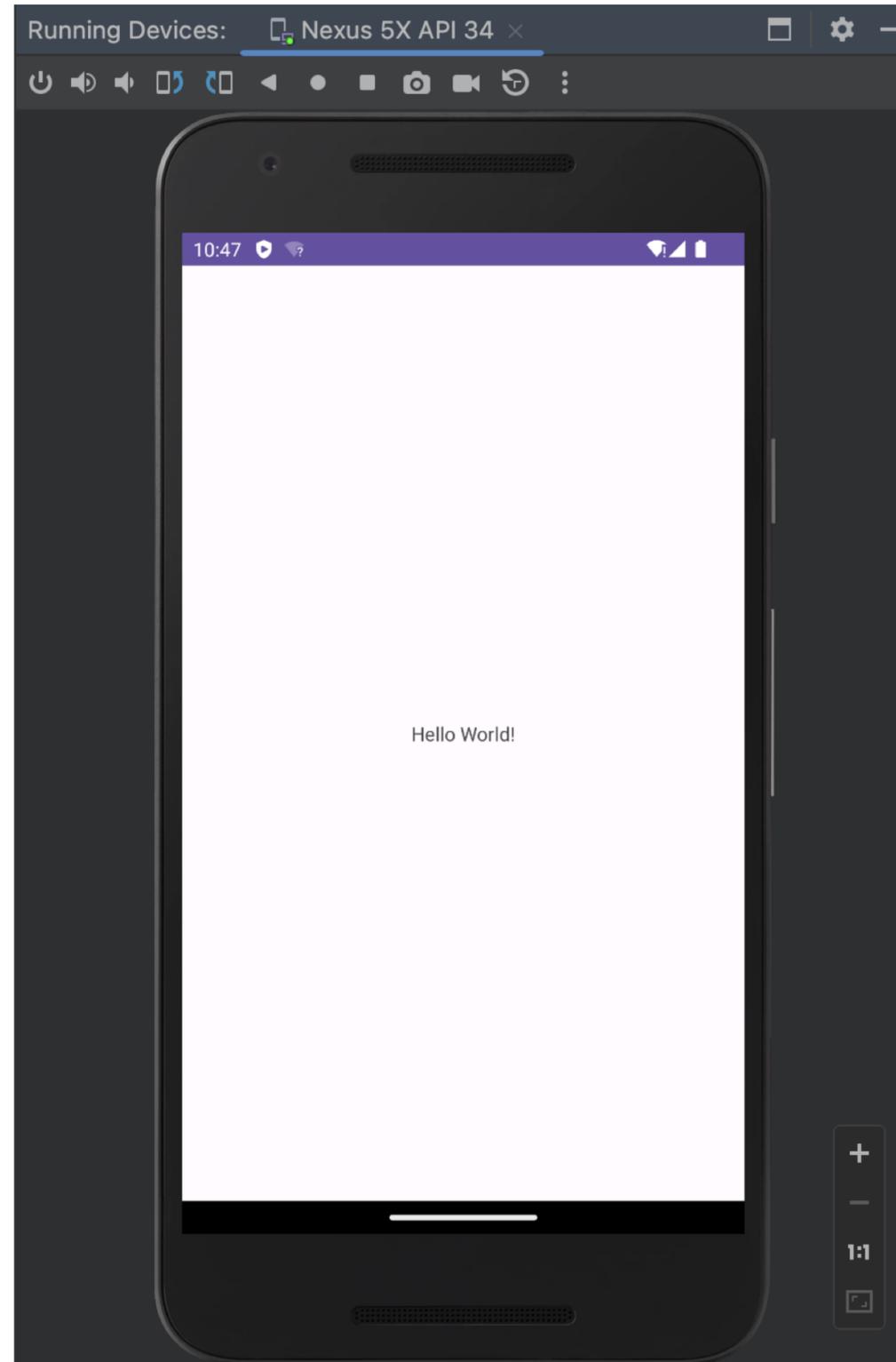


配置虚拟设备

- 选择虚拟硬件设备类型 (Nexus 5X)
- 选择系统版本，尽量选择可以支持新版本Android的系统



运行虚拟设备



配置实体设备

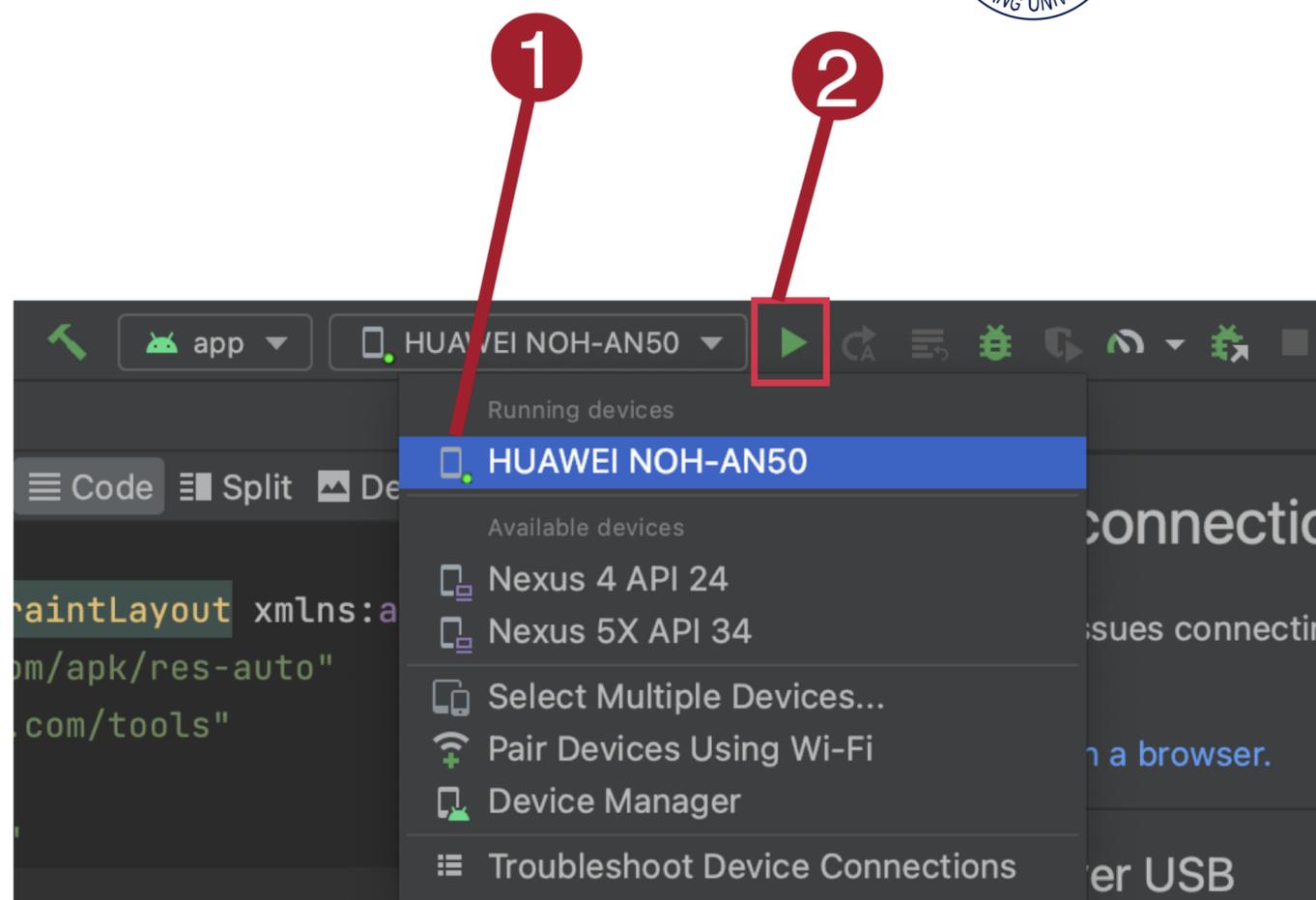
1. 打开开发者选项：
 - a. 不同品牌安卓手机略有不同，可自行查询
2. 打开USB 调试：
 - a. 设置 > 开发者选项 > USB 调试
3. 用数据线连接手机和电脑

Windows/Linux 额外步骤:

- Using Hardware Devices

Windows 驱动:

- OEM USB Drivers



- 1 选择安卓手机
- 2 将工程安装至手机并运行

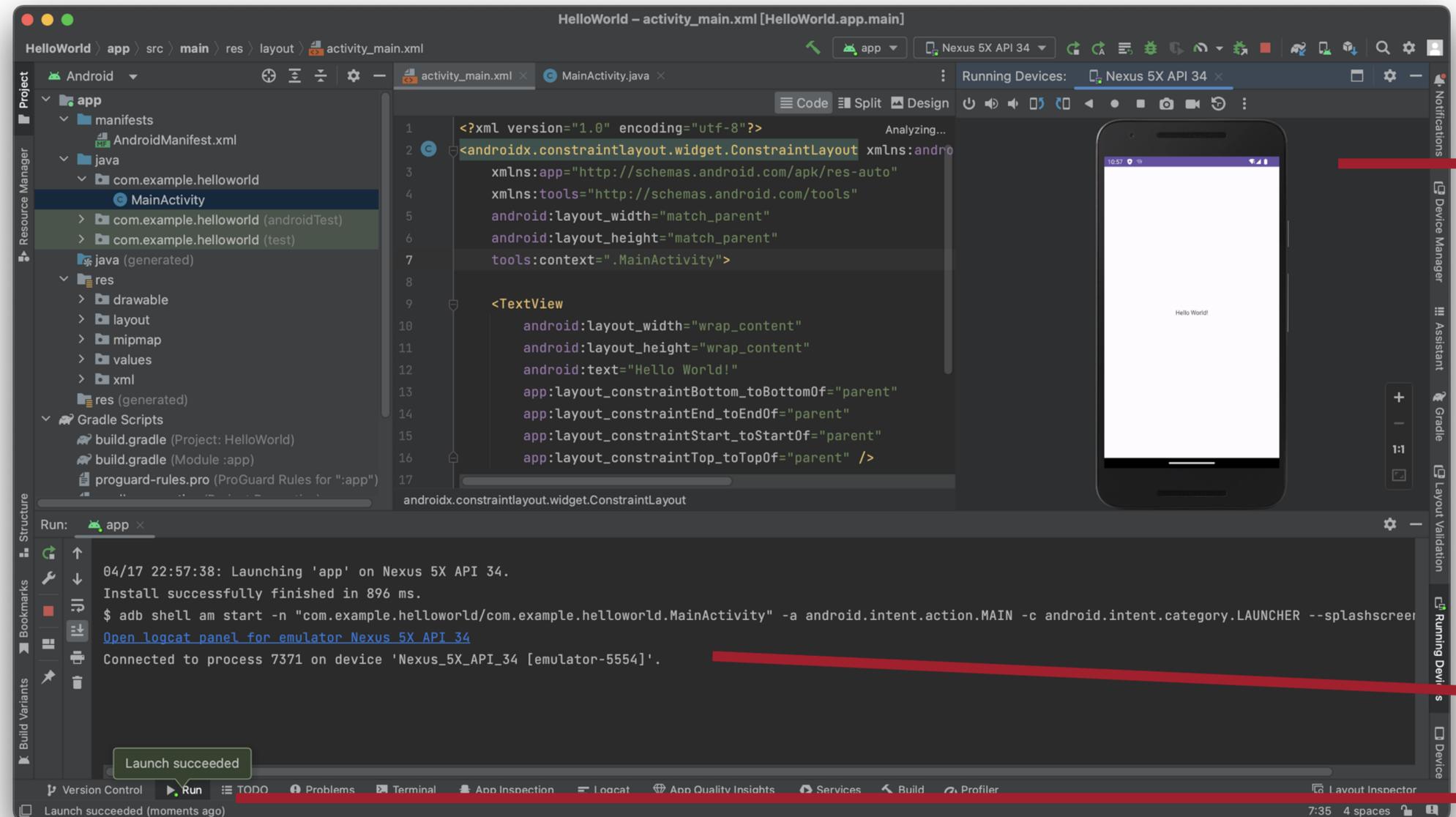


在APP运行时获得反馈

1 模拟器运行APP

2 运行面板

3 运行选项卡：用来打开或者关闭运行面板



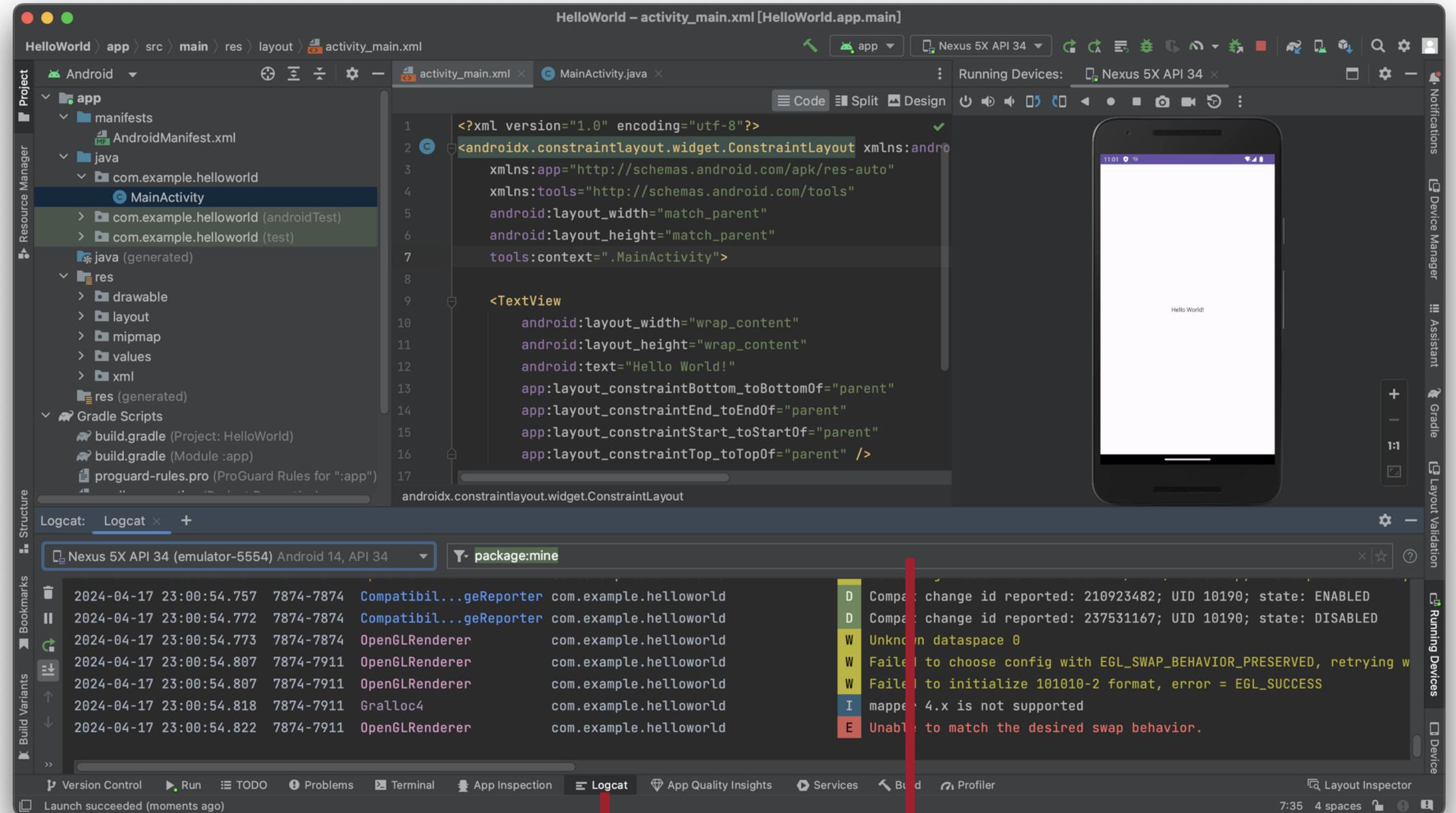
在APP中加入日志

- App运行时，**Logcat**面板显示**日志信息**
- 在源代码中加入日志（log）的代码，会在 Logcat 面板中显示
- 在 Logcat 面板中加入**过滤器**，以关注对你而言最重要的日志
- 使用 **tag** 进行搜索



Logcat

- 1 点击 Logcat 选项卡以显示 Logcat 面板
- 2 Log 级别菜单



源码中加入Log

- Log的种类：ERROR, WARN, INFO, DEBUG, VERBOSE
- 骆驼式命名法(Camel Case):
TextView, MainActivity, setText, ...

```
import android.util.Log;

// Use class name as tag
private static final String TAG =
    MainActivity.class.getSimpleName();

// Show message in Android Monitor, logcat pane
// Log.<log-level>(TAG, "Message");
Log.d(TAG, "Creating the URI..");
```

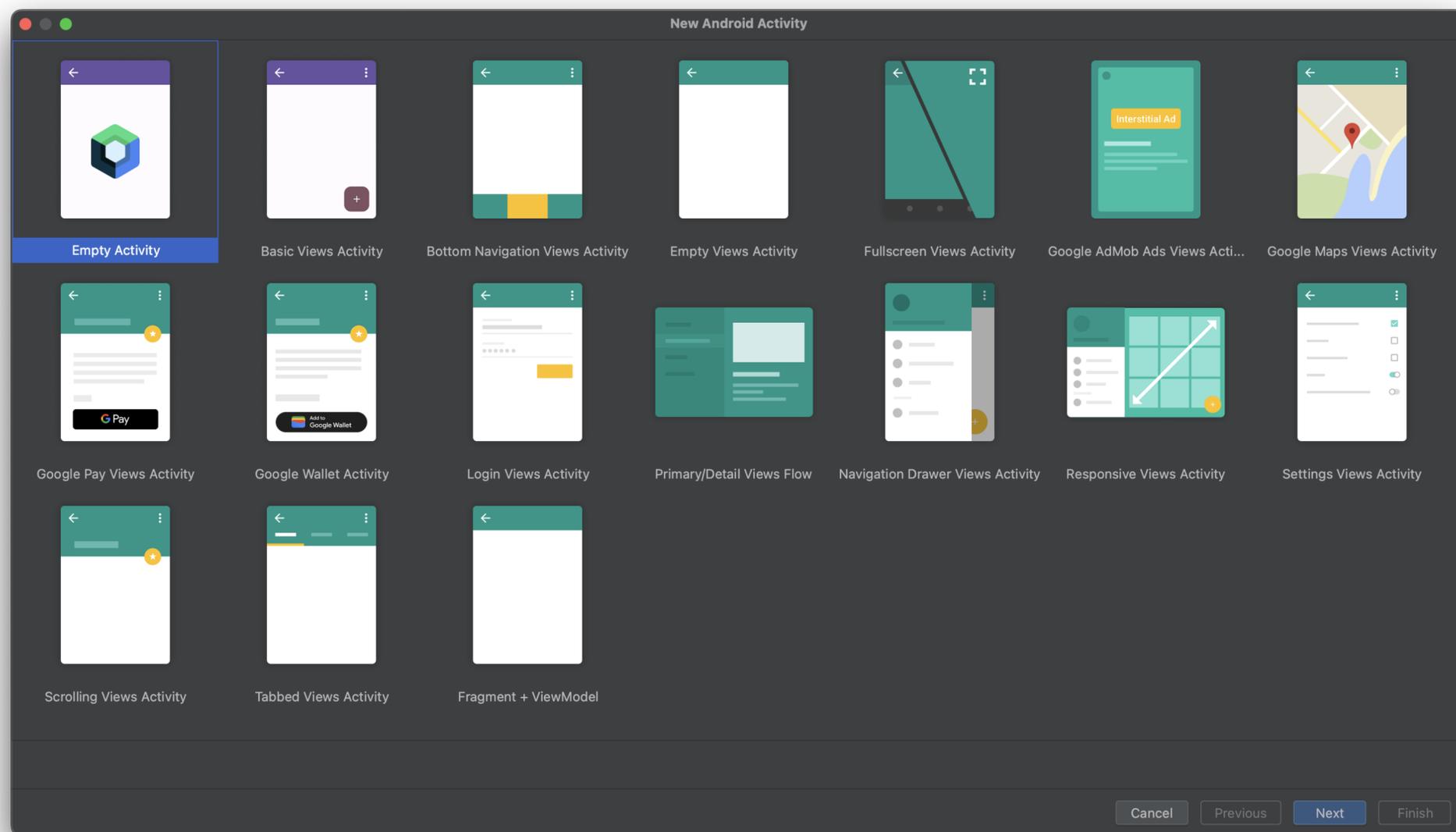


- **应用组件**是 Android 应用的**基本构建基块**。每个组件都是一个不同的功能，系统可以通过组件进入应用，并非所有组件都是用户的实际入口点。有些组件相互依赖，但每个组件都以独立实体形式存在，并发挥特定作用
- Android有四种核心组件：**活动（Activity）**，**服务**，**内容提供程序**，**广播接收器**



应用组件：Activity

- Activity表示具有用户界面的单一屏幕，**Activity子类**
- Activity类是Android应用程序的重要组成部分，**活动的启动和组合方式**是平台应用程序模型的基本组成部分
- Android系统通过调用对应于其**生命周期特定阶段**的回调函数，在Activity实例中**启动代码**
- Android中所有活动都是与用户交互，因此Activity类负责创建一个**包含UI的窗口**



一个 APP 的所有 Activity



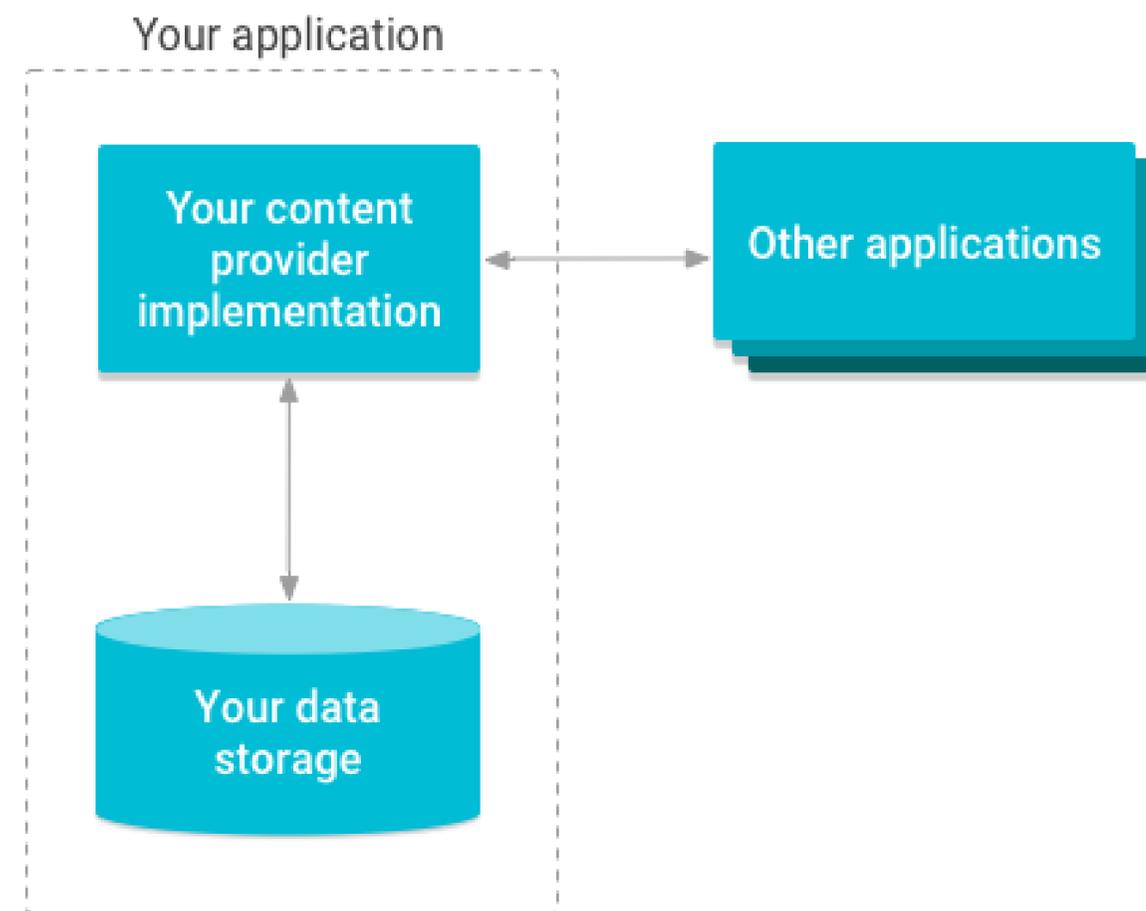
应用组件：服务

- 服务是一种在**后台运行**的组件，用于执行长时间运行的操作，或为远程进程执行作业
- 服务**不提供用户界面**。例如，当用户位于其他应用中时，服务可能在后台播放音乐
- 服务作为 **Service子类**实现



应用组件：内容提供程序

- 内容提供程序**管理**一组共享的应用数据
- **其他应用**可以通过内容提供程序**查询**数据，甚至**修改**数据（如果内容提供程序允许）
- 例如，Android 系统可提供管理用户联系人信息的内容提供程序
- 内容提供者也适用于读取和写入应用中不共享的私有数据。例如，记事本示例应用适用内容提供者来保存笔记
- **ContentProvider**子类实现，并且必须实现让其他应用能够执行事务的一组标准API



内容提供程序如何管理存储空间访问的概览图



应用组件：广播接收器

- 广播接收器是一种用于响应系统范围广播通知的组件
- 许多广播都是由系统发起的。例如，通知屏幕已关闭、电池电量不足的广播
- 尽管广播接收器不会显示用户界面，但它们可以创建状态栏通知，在发生广播事件时提醒用户
- 广播接收器更常见的用途只是作为通向其他组件的“通道”，设计用于执行极少量的工作
- 广播接收器作为 **BroadcastReceiver** 子类实现，并且每条广播都作为 **Intent** 对象进行传递



应用组件的特点

- Android系统设计的独特之处在于，任何应用都可以启动其他应用的组件
- 例如，如果让用户使用设备的相机拍摄照片
 - 如果有另一个相机应用可以执行该操作
 - 可以利用这个相机应用的Activity，而不用自行开发一个
 - 完成拍摄时，系统会将照片返回你的应用。对用户而言，就好像相机真正是你的应用的组成部分



应用组件的特点

- 当系统启动某个组件时，会启动该应用的进程（如果尚未运行），并实例化该组件所需的类
- 例如，如果你的应用启动相机应用中拍摄照片的 **Activity**，则该 **Activity** 会在属于相机应用的进程中运行
- 因此，与大多数其他系统上的应用不同，**Android** 应用并没有**单一入口点**（例如，没有 `main()` 函数）



应用组件的特点

- Android系统在单独的进程中运行每个应用，且其文件权限会限制对其他应用的访问
- 因此应用无法直接启动其他应用中的组件，但 Android 系统却可以
- 要想启动其他应用中的组件，应用必须向系统传递一则消息，说明想启动特定组件的**Intent**。系统随后便会为你的应用启动该组件



Intent

- 四种组件类型中的三种 — **Activity**、**服务和广播接收器**，通过名为**Intent**的**异步消息**进行启动
- **Intent**会在运行时将各个组件相互绑定，无论组件属于你的应用还是其他应用
- 可以将**Intent**视为从其他组件请求操作的**信使**



Intent在Activity和服务

- Intent 定义要**执行的操作**。例如，“查看”或“发送”某个内容
- Intent可以指定要执行操作的**数据的URI**（以及正在启动的组件可能需要了解的信息）
- Intent中可以**返回结果**



通过 intent 调用其他应用 activity 查看 ppt



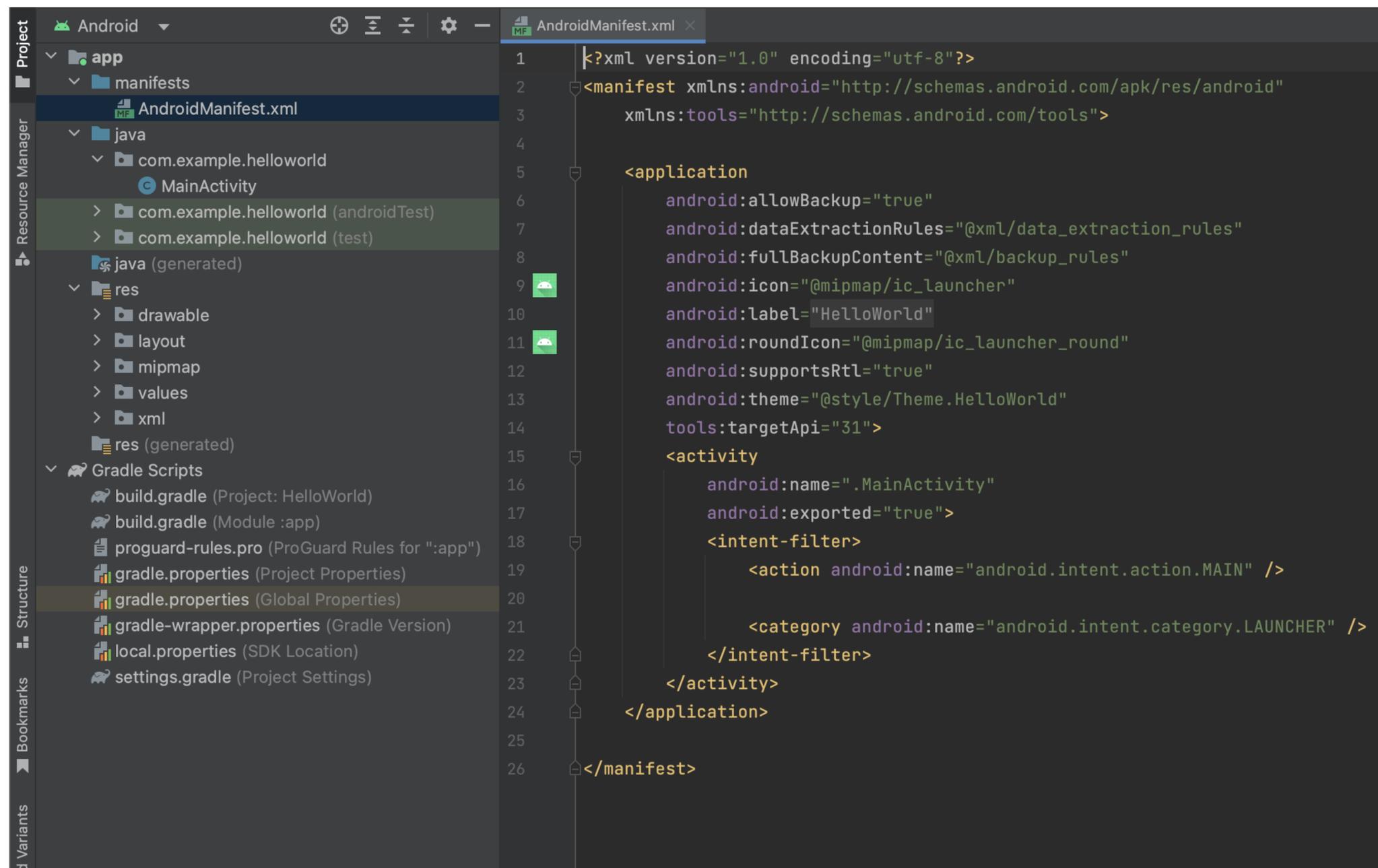
Intent在广播服务器

- Intent只会定义要广播的通知
- 例如，指示设备电池电量不足的广播只包括指示“电池电量不足”的已知操作字符串



清单文件

- 在 Android 系统启动应用组件之前，系统必须通过读取对应 **AndroidManifest.xml** 文件（“清单”文件）确认组件存在
- 应用必须在此文件中 **声明其所有组件**，该文件必须位于应用项目的根目录中



```
1 |<?xml version="1.0" encoding="utf-8"?>
2 | <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3 |   xmlns:tools="http://schemas.android.com/tools">
4 |
5 |   <application
6 |     android:allowBackup="true"
7 |     android:dataExtractionRules="@xml/data_extraction_rules"
8 |     android:fullBackupContent="@xml/backup_rules"
9 |     android:icon="@mipmap/ic_launcher"
10 |    android:label="HelloWorld"
11 |    android:roundIcon="@mipmap/ic_launcher_round"
12 |    android:supportsRtl="true"
13 |    android:theme="@style/Theme.HelloWorld"
14 |    tools:targetApi="31">
15 |     <activity
16 |       android:name=".MainActivity"
17 |       android:exported="true">
18 |       <intent-filter>
19 |         <action android:name="android.intent.action.MAIN" />
20 |
21 |         <category android:name="android.intent.category.LAUNCHER" />
22 |       </intent-filter>
23 |     </activity>
24 |   </application>
25 |
26 | </manifest>
```



清单文件的功能

- 清单文件还有许多其他作用，例如：
 - 确定应用需要的**用户权限**
 - 根据应用使用的**API**，声明应用所需的**最低 API 级别**
 - 声明应用使用或需要的**硬件和软件功能**，如相机、蓝牙服务或多点触摸屏
 - 应用需要链接的**API 库**（Android 框架 API 除外），如 Google 地图库

```
<manifest ... >  
  <uses-permission android:name="android.permission.INTERNET" />  
  
  <uses-feature android:name="android.hardware.camera.any"  
    android:required="true" />  
  
  <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="19" />  
  ...  
</manifest>
```

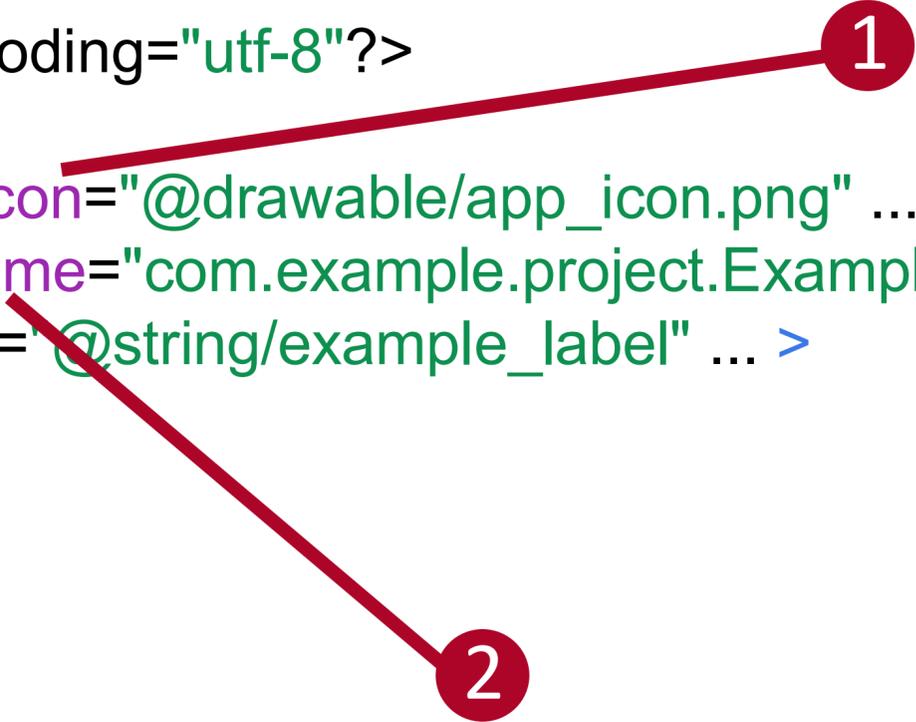


清单文件的功能

1 **android:icon** 属性指向标识应用的图标所对应的资源

2 **android:name** 属性指定 Activity 子类的完全限定类名

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
  <application android:icon="@drawable/app_icon.png" ... >
    <activity android:name="com.example.project.ExampleActivity"
      android:label="@string/example_label" ... >
    </activity>
  ...
</application>
</manifest>
```



清单文件声明组件

- 必须通过以下方式声明所有应用组件
 - Activity 的 `<activity>` 元素
 - 服务的 `<service>` 元素
 - 内容提供程序的 `<provider>` 元素
 - 广播接收器的 `<receiver>` 元素
- 在源代码中，但未在清单文件中声明的Activity、服务和内容提供程序对系统不可见，也永远不会运行



清单文件声明组件

- 可以使用 **Intent** 启动 Activity、服务和广播接收器
- 在应用的清单文件中声明Activity时，可以选择性地加入声明Activity功能的**Intent过滤器**，以便响应来自其他应用的Intent
- Android通过将接收到的 Intent 与设备上的其他应用的清单文件中提供的 Intent 过滤器进行比较来确定可以响应Intent的组件



清单文件声明组件

- 如果一个email应用包含一个用于撰写新电子邮件的Activity，则可以像右边声明一个Intent过滤器来响应“send”Intent（以发送新电子邮件）

```
<manifest ... >
...
<application ... >
  <activity android:name="com.example.project.ComposeEmailActivity">
    <intent-filter>
      <action android:name="android.intent.action.SEND" />
      <data android:type="*/*" />
      <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
  </activity>
</application>
</manifest>
```



清单文件声明组件

- 基于 Android 系统的设备多种多样，硬件和功能各不相同。为防止将应用安装在缺少应用所需特性的设备上，必须在清单文件中**声明设备和软件要求**
- 其中大多数声明只是提供信息，**系统不会读取它们**，但 **Google Play 等外部服务会读取它们**，以便当用户在其设备中搜索应用时为用户提供过滤功能



清单文件声明组件

- 如果应用需要相机，并使用 Android 2.1（API 级别 7）中引入的 API，在清单文件中应该这样声明

```
<manifest ... >  
    <uses-feature android:name="android.hardware.camera.any"  
                android:required="true" />  
    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="19" />  
    ...  
</manifest>
```



小结

- Android Studio功能
- 应用组件：活动（Activity），服务，内容提供程序，广播接收器
 - Activity - UI，服务 - 后台应用，内容提供程序 - 数据管理，广播接收器 - 消息
 - Activity、服务和广播接收器通过Intent 的异步消息进行启动
- 清单文件
 - 组件申明，Activity的Intent过滤器
 - 用户权限，硬件与软件功能
 - 最低API级别，外部API库



课程目录



1

Android 简介

2

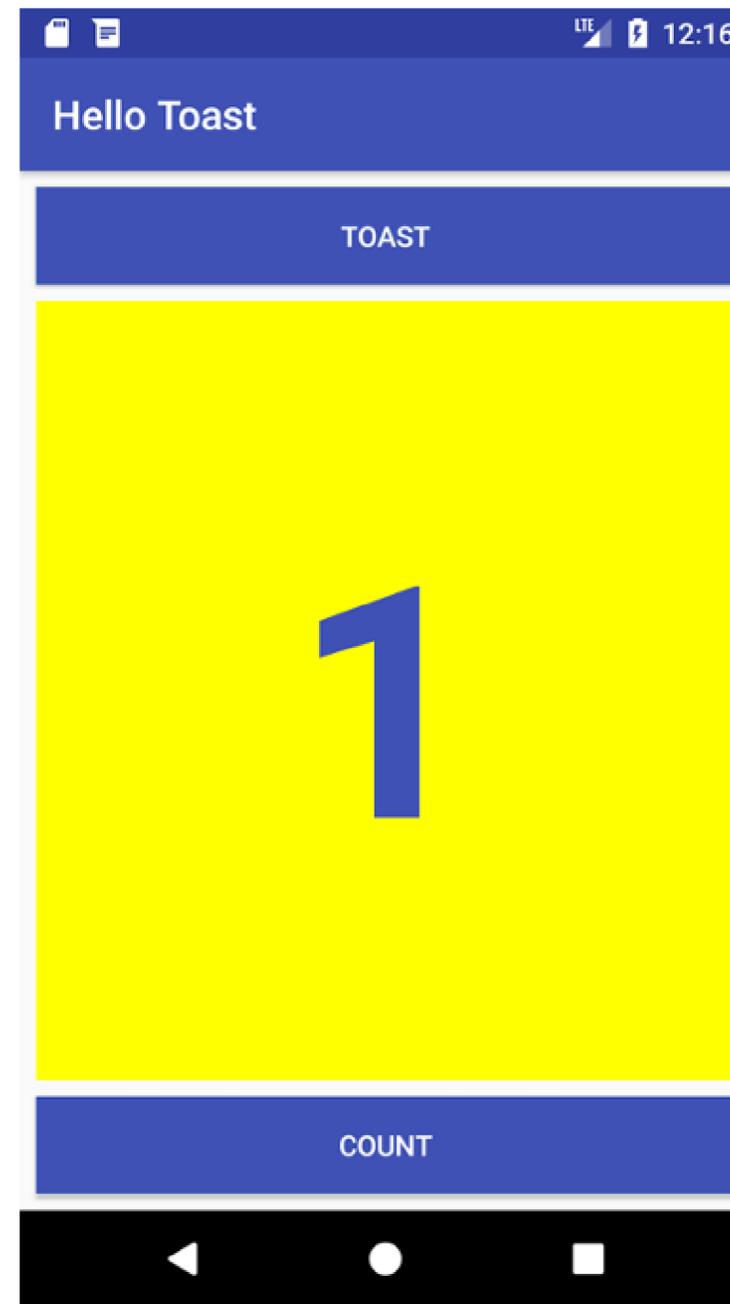
第一个Android App

3

视图, 布局和资源

什么是视图？

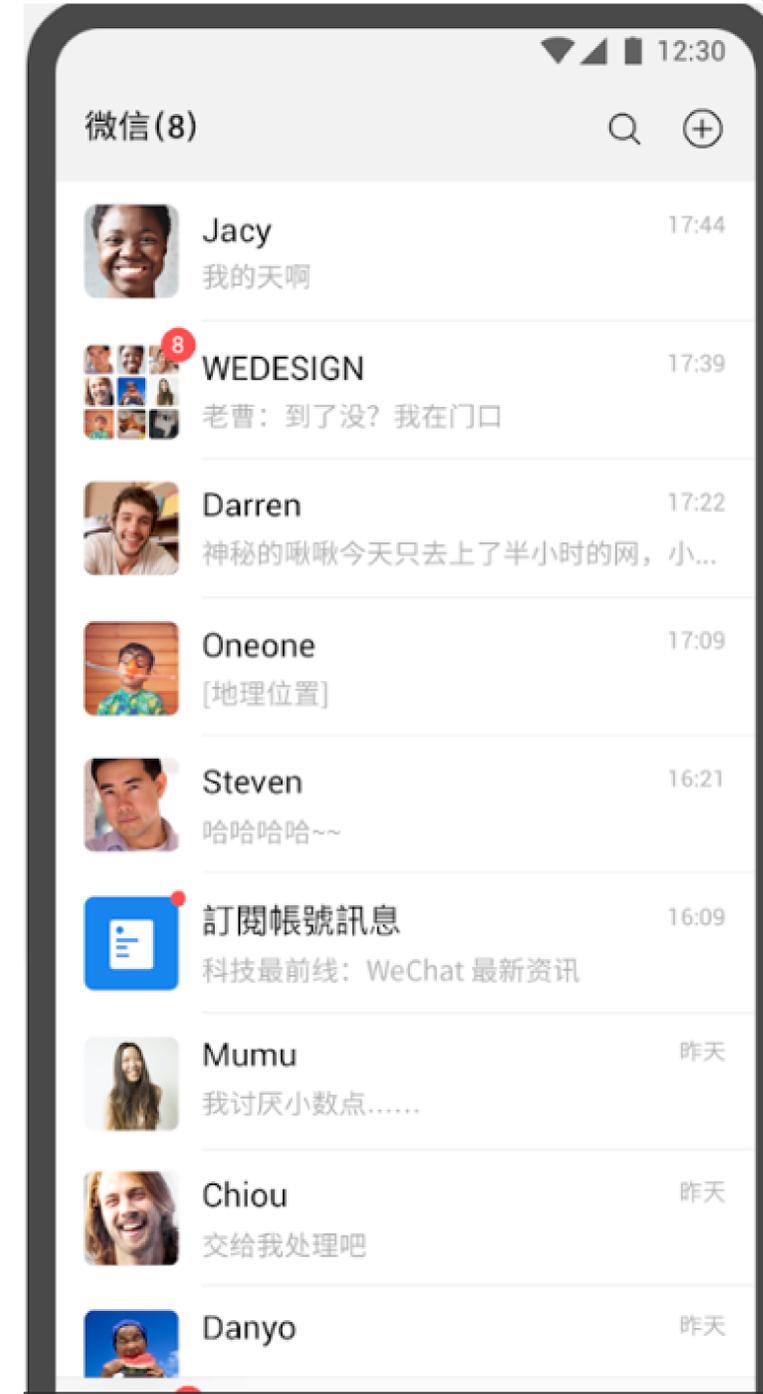
- 简单地说，打开你的手机（如右图），所有你看到的东西（**UI element**）都是视图



视图



- 视图类是构建用户界面的基本元素，是UI组件（如按钮，复选框和文本输入字段）的基类
- Android系统提供数百个预定义视图
 - 文本(TextView), 可编辑文本(EditText)
 - 按钮(Button), 菜单等交互式视图
 - 滑动组件(ScrollView, RecyclerView)
 - 显示图像(ImageView)
 - 视图组(ConstraintLayout, LinearLayout)

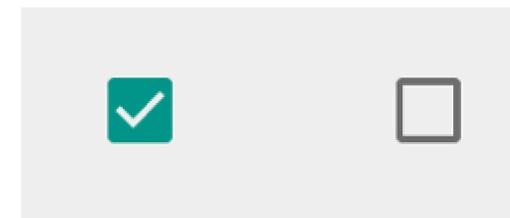


一些常用的视图

Button



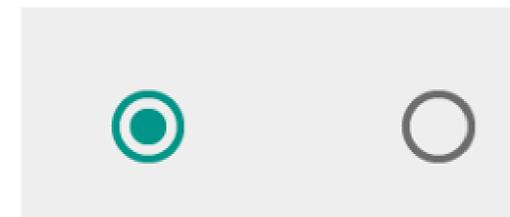
CheckBox



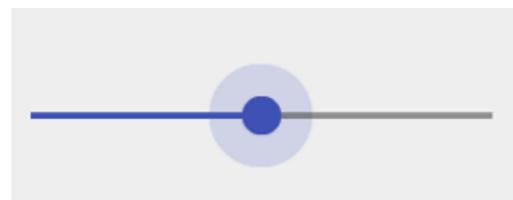
EditText



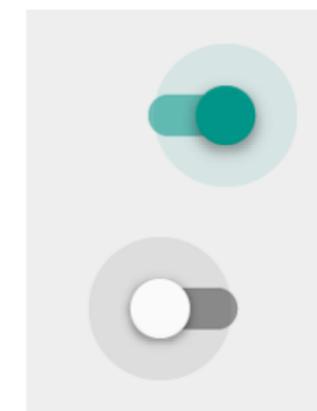
RadioButton



Slider



Switch



视图属性

- 颜色，尺寸，位置
- 一些视图有焦点（点击后接收用户输入）
- 互动式（响应用户点击）
- 是否可见
- 与其他视图之间的联系



创建视图/布局的方式

- 方法一：Android Studio **Layout Editor**（类似xcode的可视化编辑）
- 方法二：通过编写**XML代码**（类似HTML）
- 方法三：通过**Java/Kotlin代码**创建



Android Studio Layout Editor

① XML布局文件

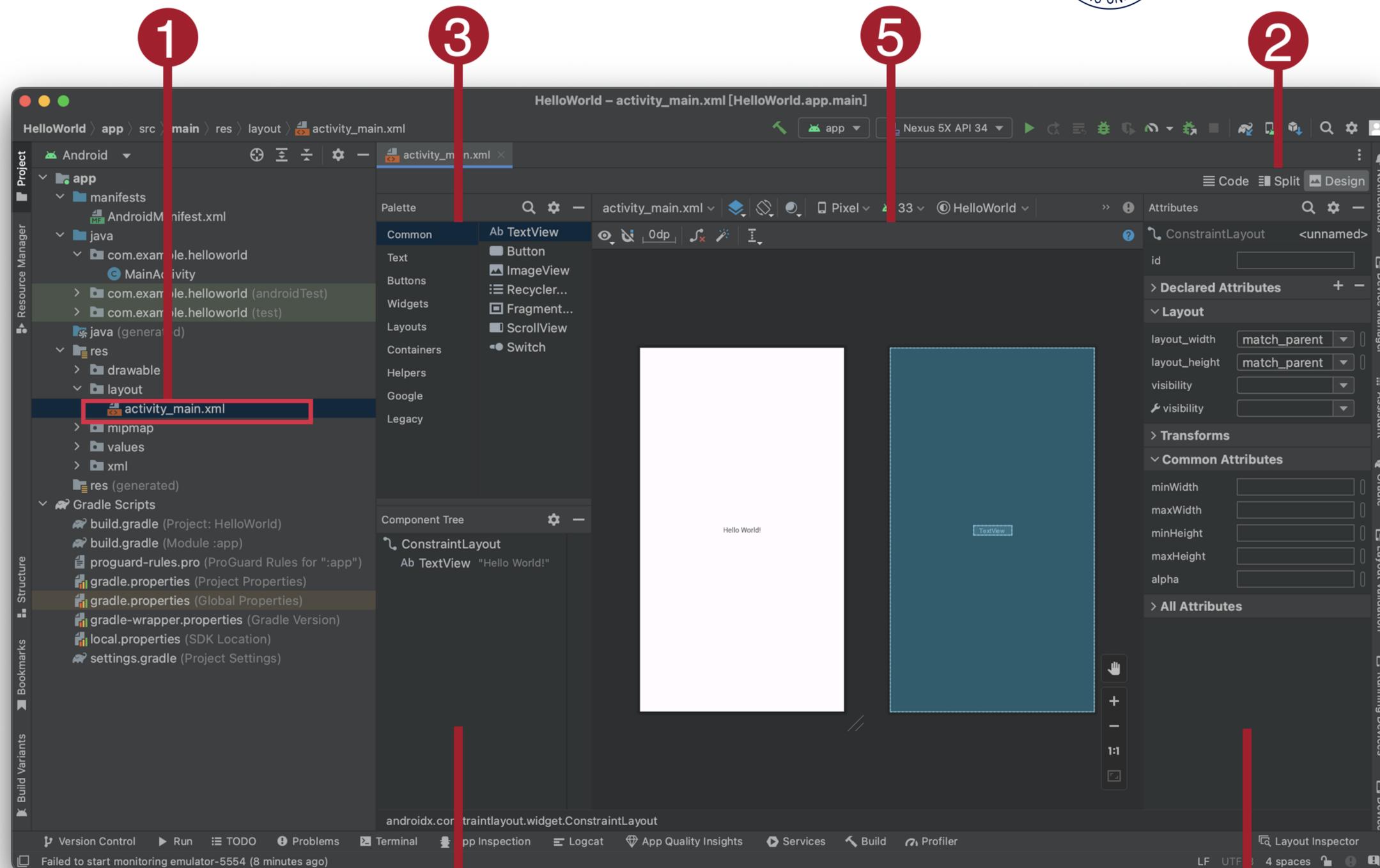
② 设计和文本选项卡

③ 调色板窗格

④ 组件树

⑤ 设计和蓝图窗格

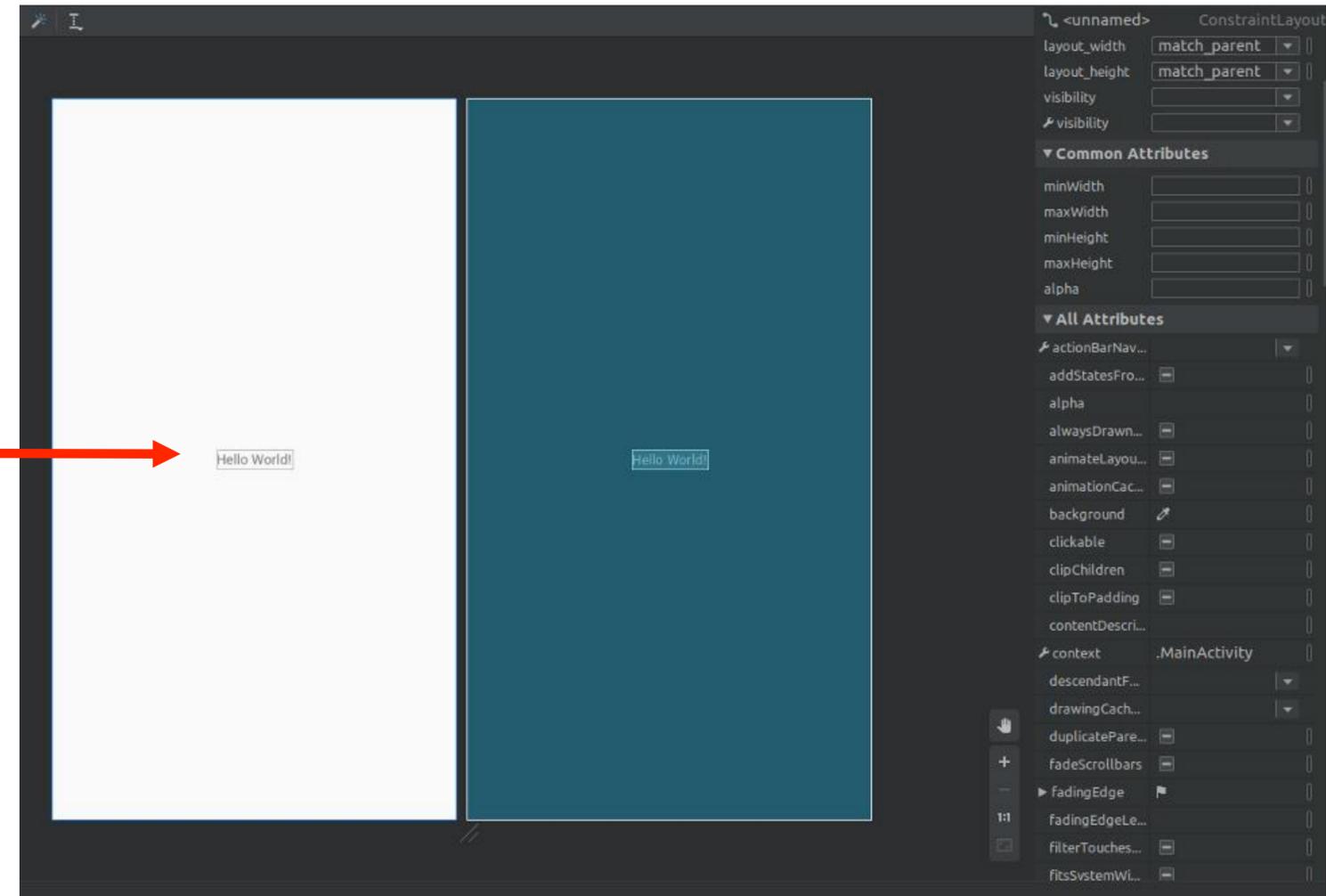
⑥ 属性选项卡



Android Studio Layout Editor



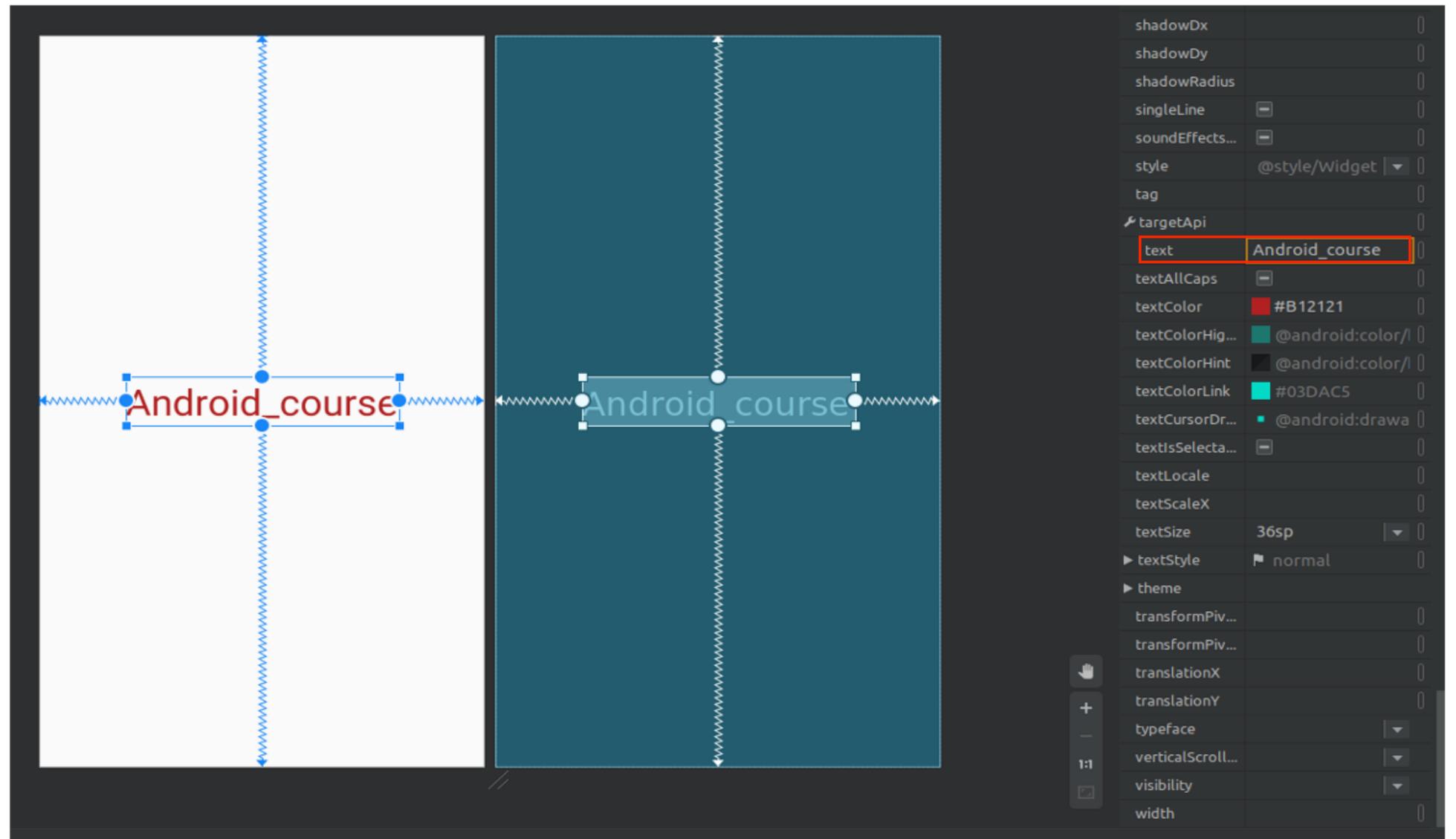
```
<TextView  
android:layout_width="wrap_content"  
android:layout_height="wrap_content"  
android:text="Hello World!"  
app:layout_constraintBottom_toBottomOf="parent"  
app:layout_constraintLeft_toLeftOf="parent"  
app:layout_constraintRight_toRightOf="parent"  
app:layout_constraintTop_toTopOf="parent" />
```



Android Studio Layout Editor



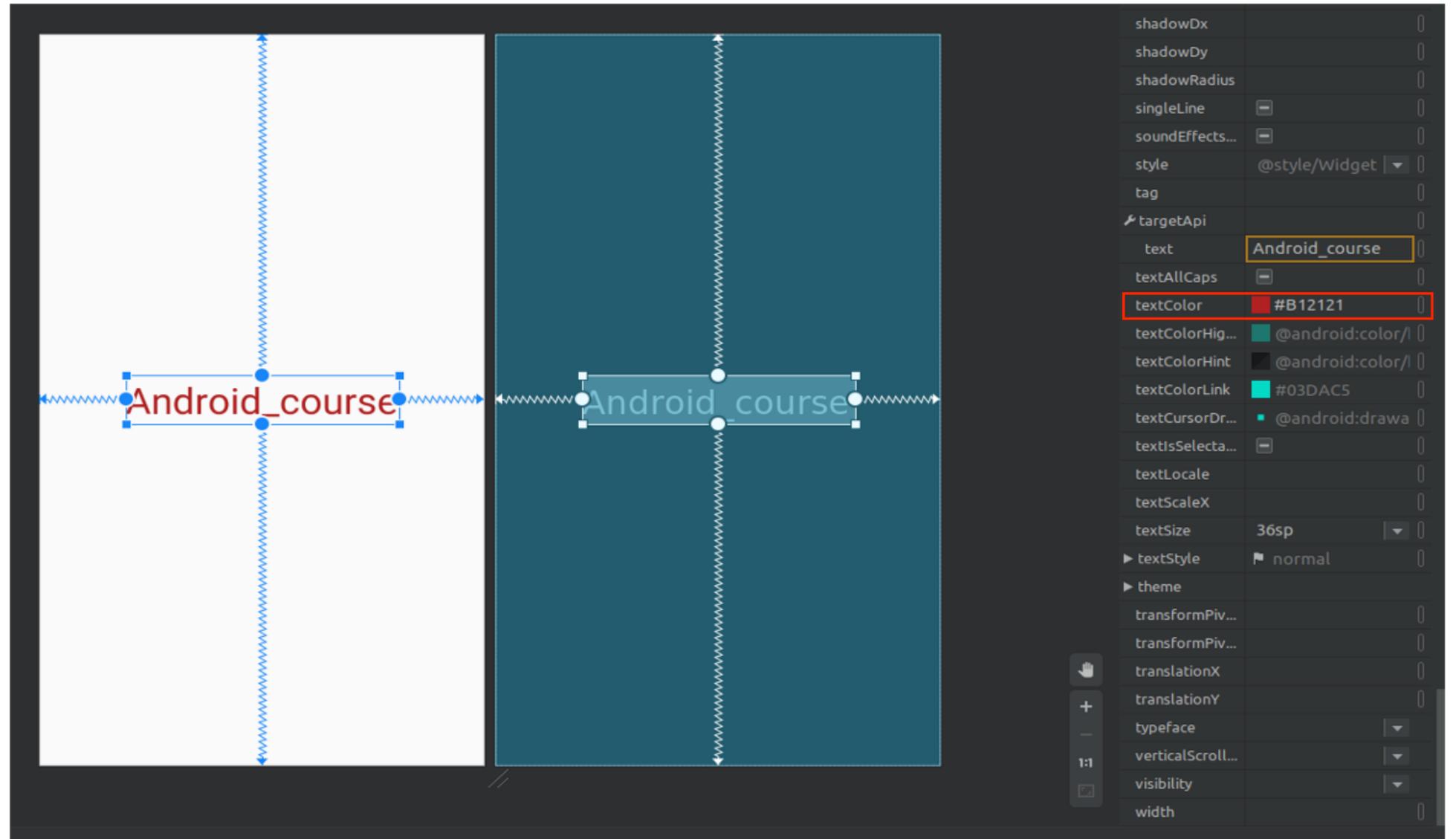
```
<TextView  
    ...  
    android:text="Android course"  
    ...  
>
```



Android Studio Layout Editor



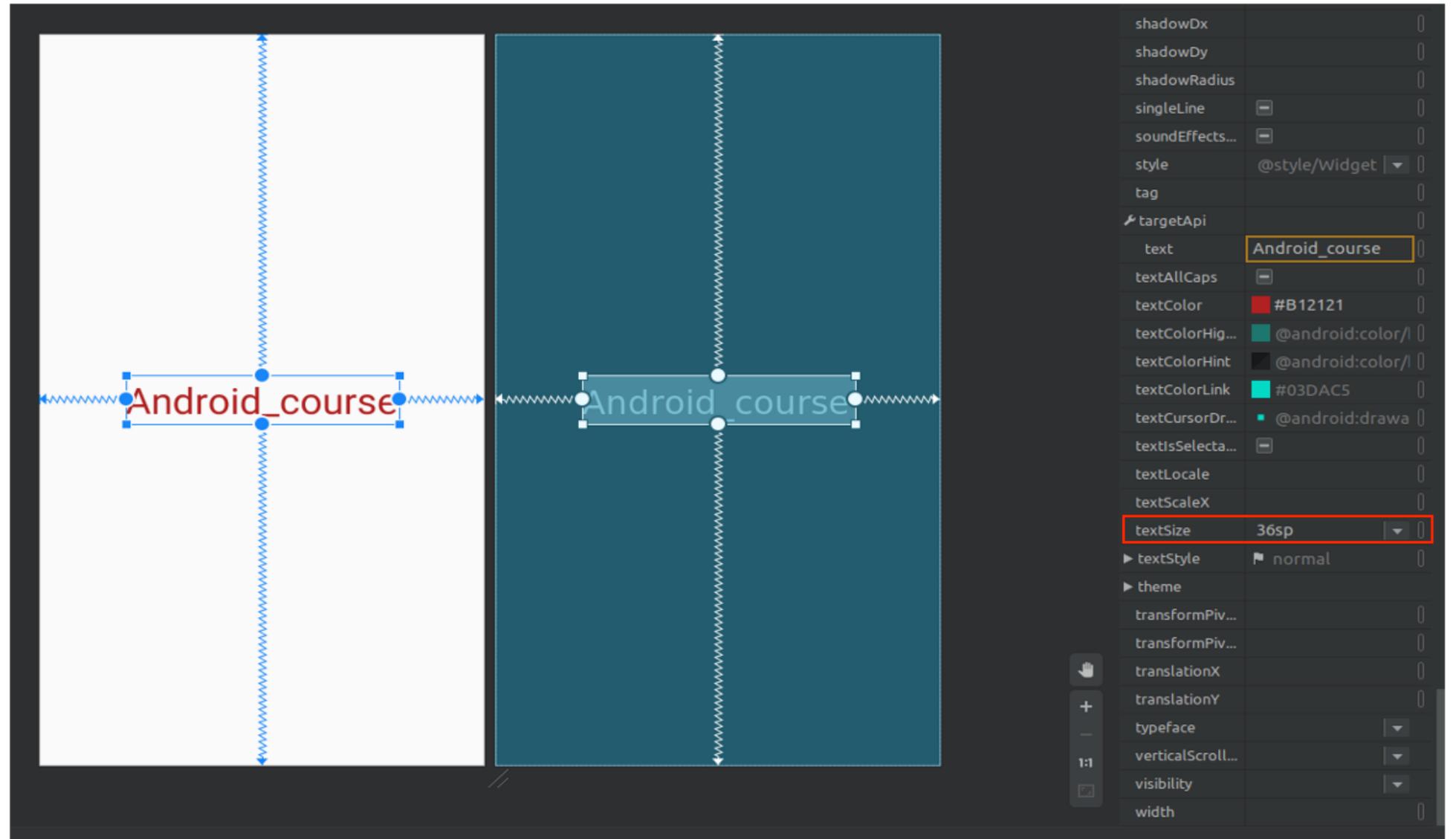
```
<TextView  
    ...  
    android:textColor="#B12121"  
    ...  
>
```



Android Studio Layout Editor



```
<TextView  
    ...  
    android:textSize="36sp"  
    ...  
>
```



Android Studio Layout Editor



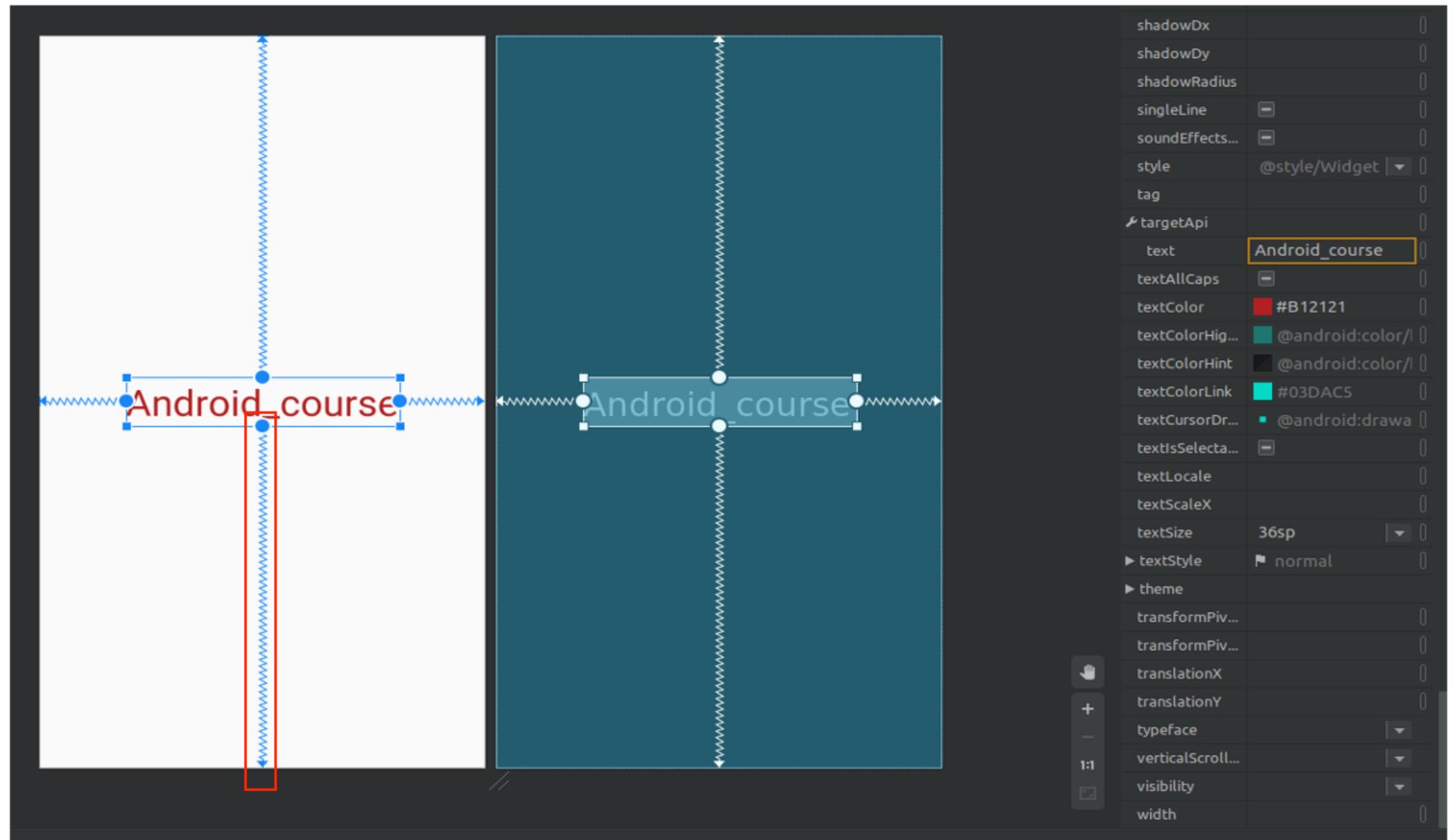
```
<TextView
```

```
...
```

```
app:layout_constraintBottom_  
toBottomOf="parent"
```

```
...
```

```
>
```



Android Studio Layout Editor



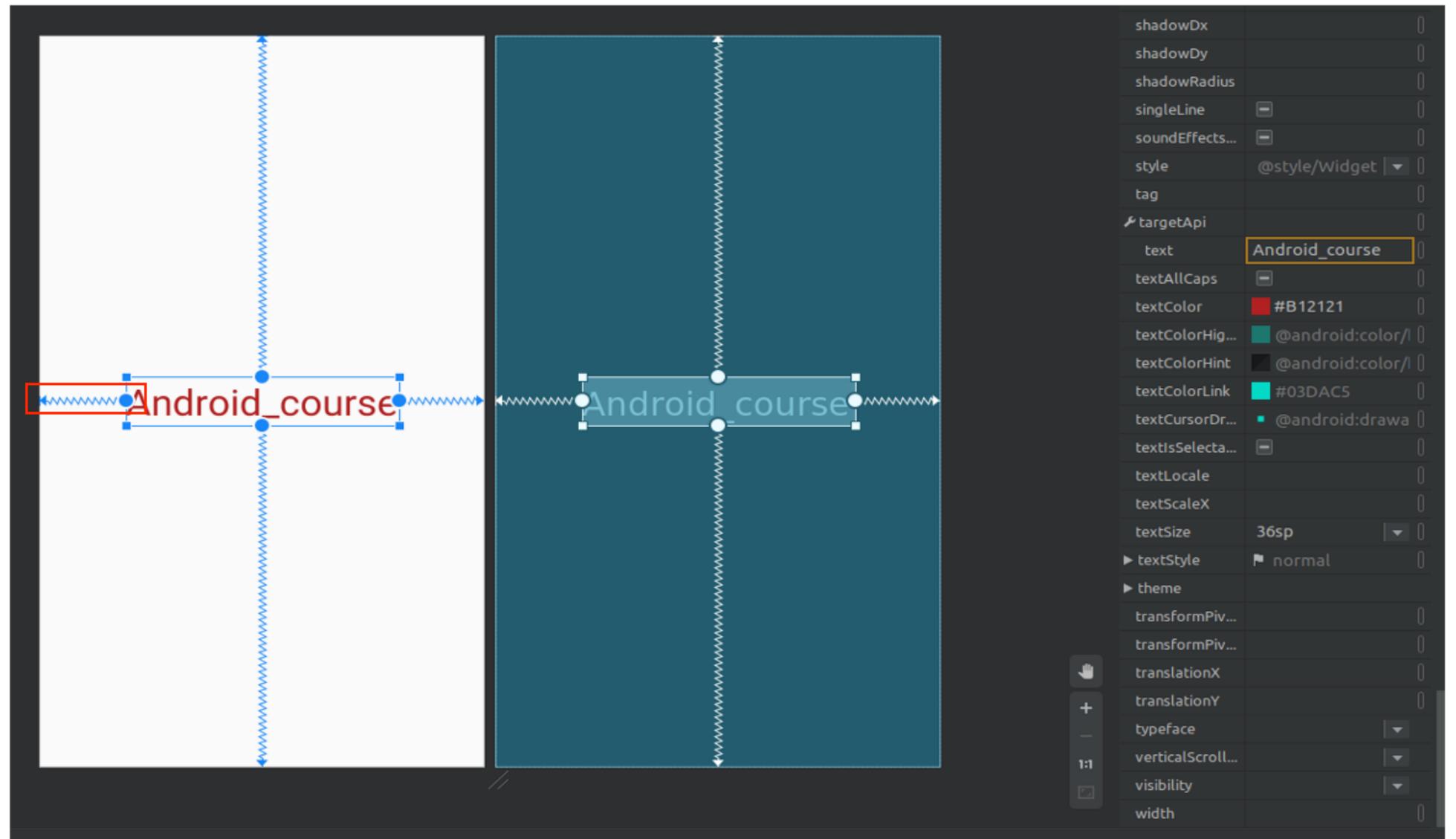
```
<TextView
```

```
...
```

```
app:layout_constraintLeft_toLeftOf="parent"
```

```
...
```

```
>
```



Android Studio Layout Editor



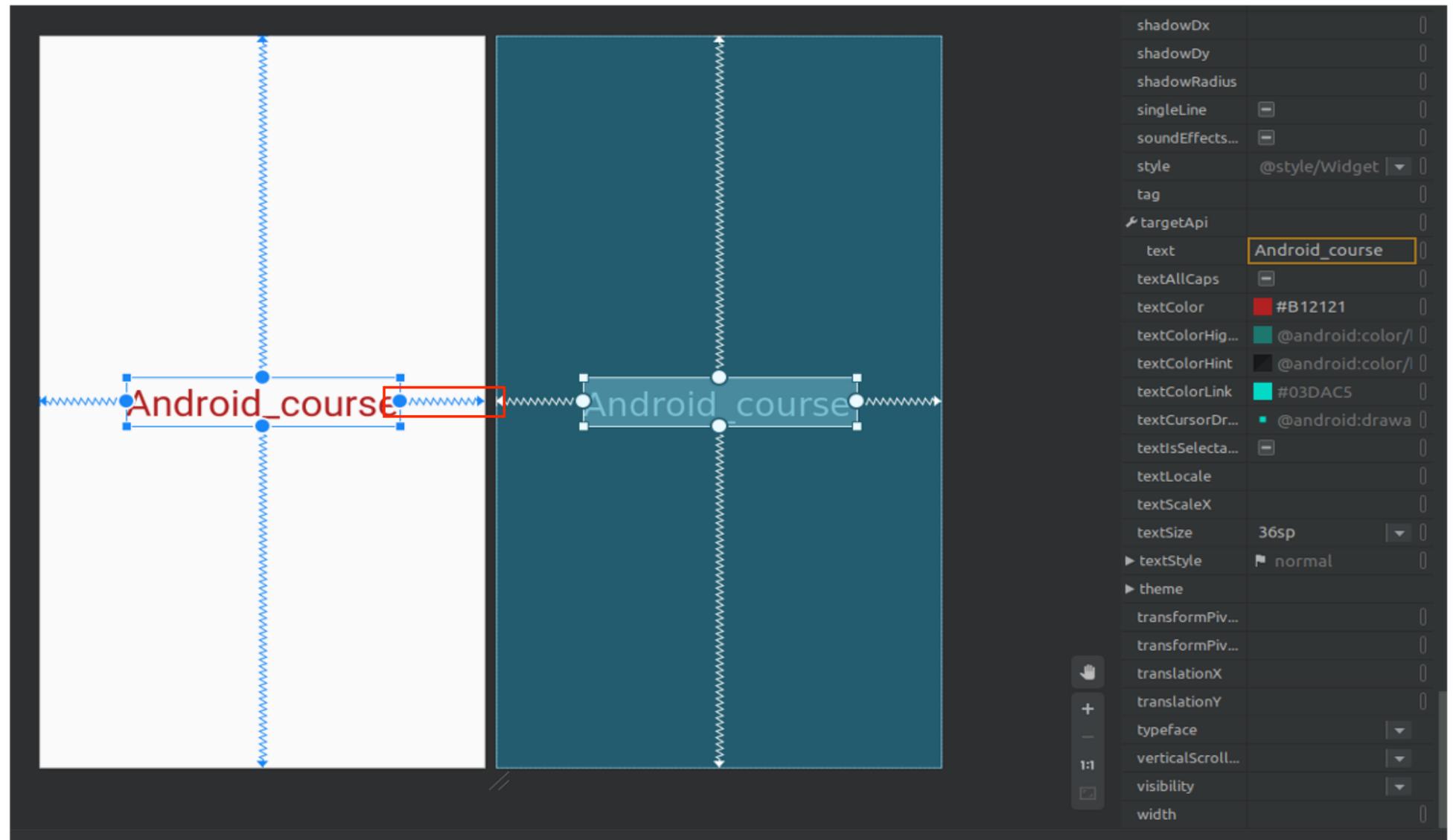
```
<TextView
```

```
...
```

```
app:layout_constraintRight_t  
oRightOf="parent"
```

```
...
```

```
/>
```



Android Studio Layout Editor



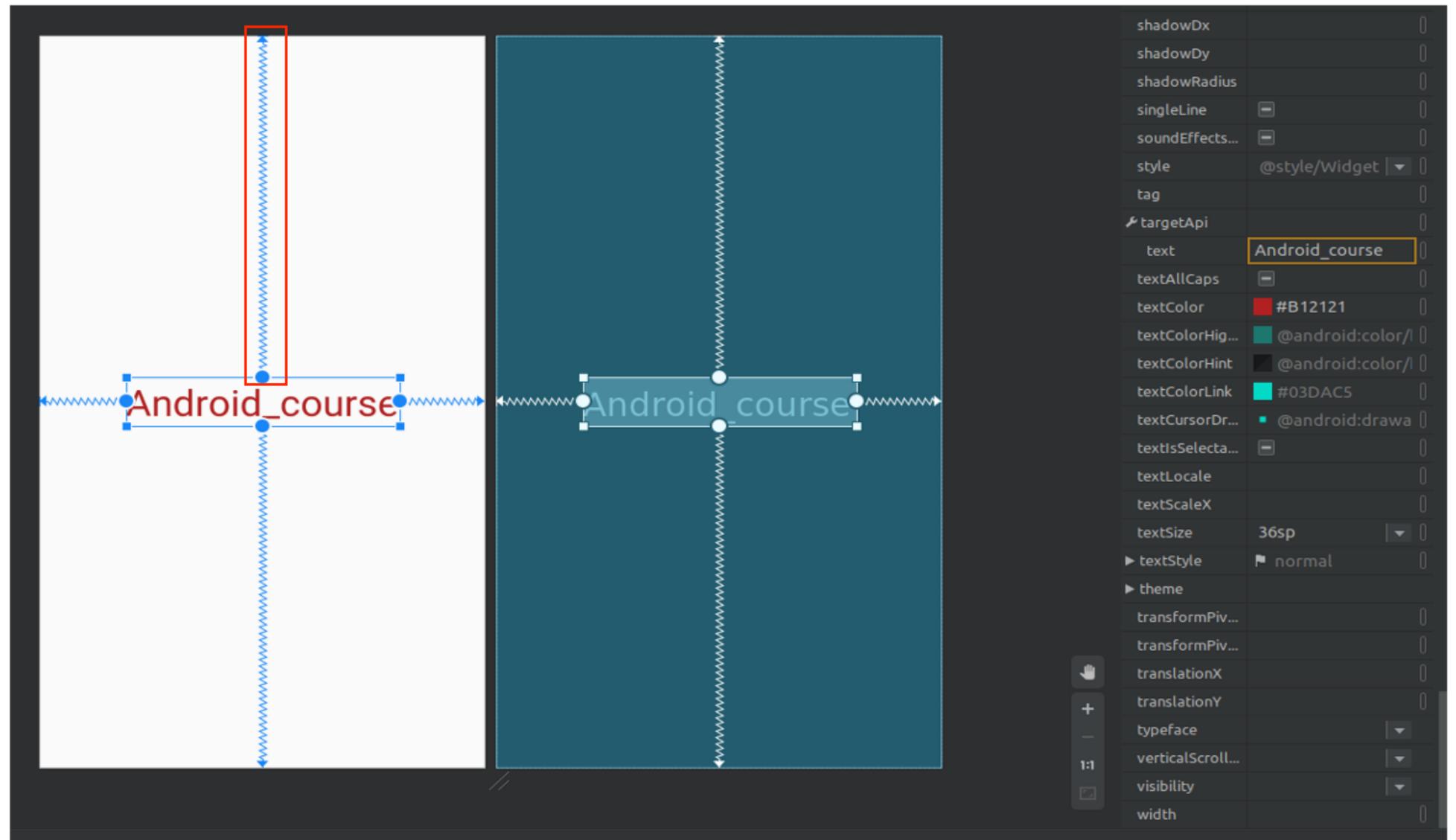
```
<TextView
```

```
...
```

```
app:layout_constraintTop_toTopOf="parent"
```

```
...
```

```
>
```



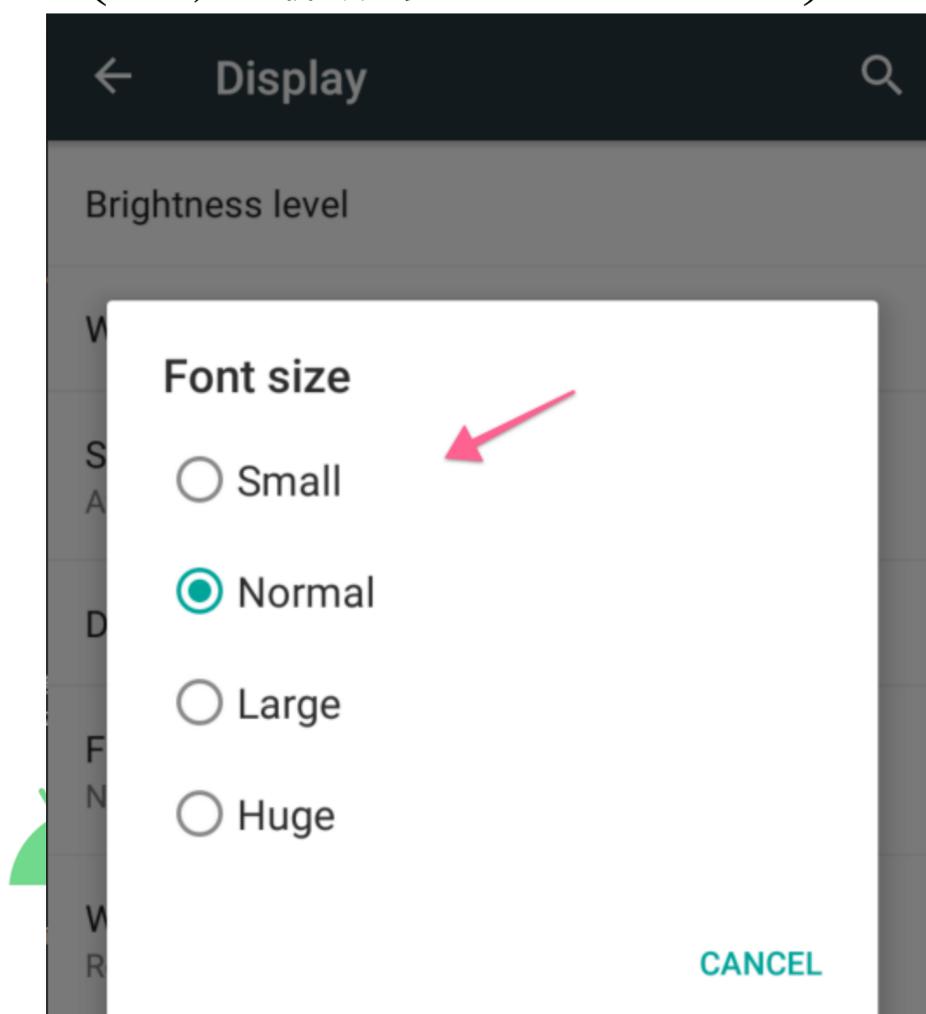
通过XML定义视图

- match_parent, wrap_content
- dp = density-independent pixels
(设备无关像素，非像素单位)
- sp = scale-independent pixels
(可缩放设备无关像素)
(用户偏好-字体大小)

```
<TextView
```

```
    android:id="@+id/show_count"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="16dp"  
    android:padding="16dp"  
    android:background="@color/myBackgroundColor"  
    android:text="@string/count_initial_value"  
    android:textColor="@color/colorPrimary"  
    android:textSize="24sp"  
    android:textStyle="bold"
```

```
/>
```



通过XML定义视图的三种格式

- android:<property_name>=“<property_value>”
 - Example: android:layout_width="match_parent"
- android:<property_name>=“@<resource_type>/resource_id”
 - Example: android:text="@string/button_label_next"
- android:<property_name>=“@+id/view_id”
 - Example: android:id="@+id/show_count"



通过Java代码创建视图



- 在Activity中编写如下代码在视图中创建一个文本框并显示“Display this text!”

```
TextView myText = new TextView(this);  
myText.setText("Display this text!");
```



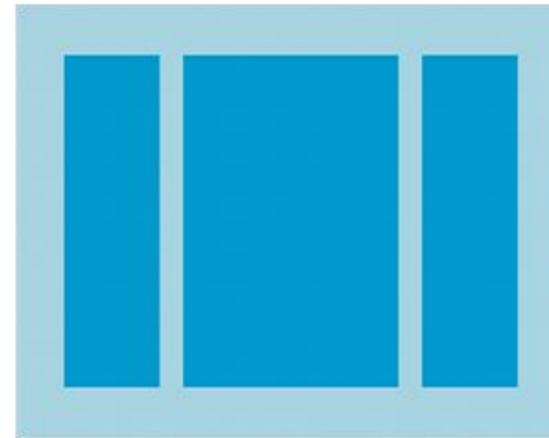
视图组和视图的层次性

- 视图组包含“子”视图，是视图的层次化表达，常见的视图组包括：
 - **ConstraintLayout**：使用约束来限制UI元素的位置。约束包括与其他元素之间或是与布局边缘之间的约束
 - **ScrollView**：包含一个元素并能滚动
 - **RecyclerView**：包含多个元素，并通过动态添加和删除元素来滚动



布局与视图组

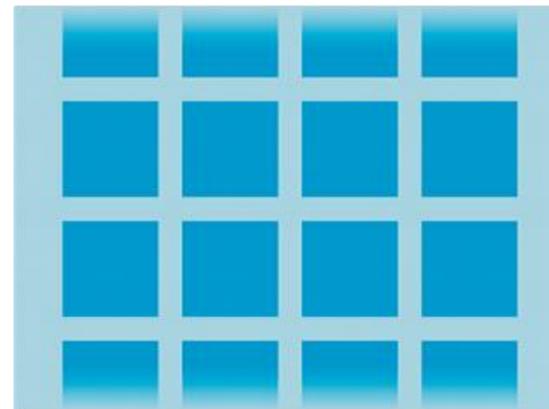
- **布局**是视图组的一种特定类型（布局类是视图组的子类），因此也包含子视图。常见的布局包括：LinearLayout, ConstraintLayout, GridLayout, TableLayout....



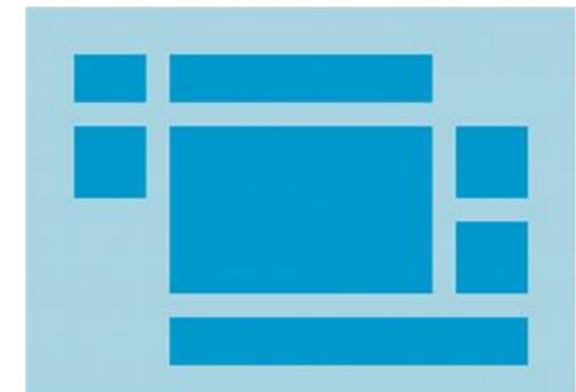
LinearLayout



ConstraintLayout



GridLayout



TableLayout

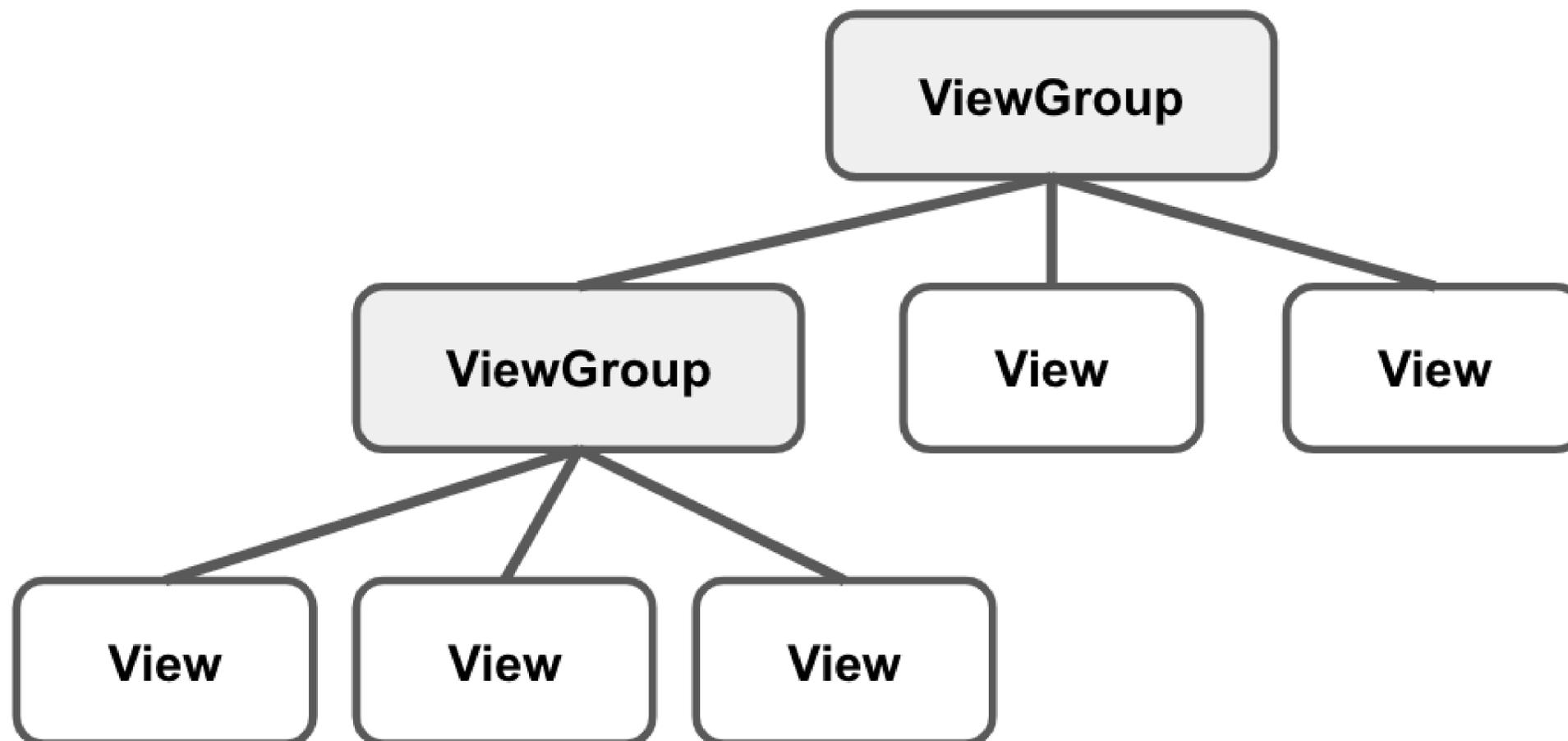


视图的层次性 vs 布局的层次性

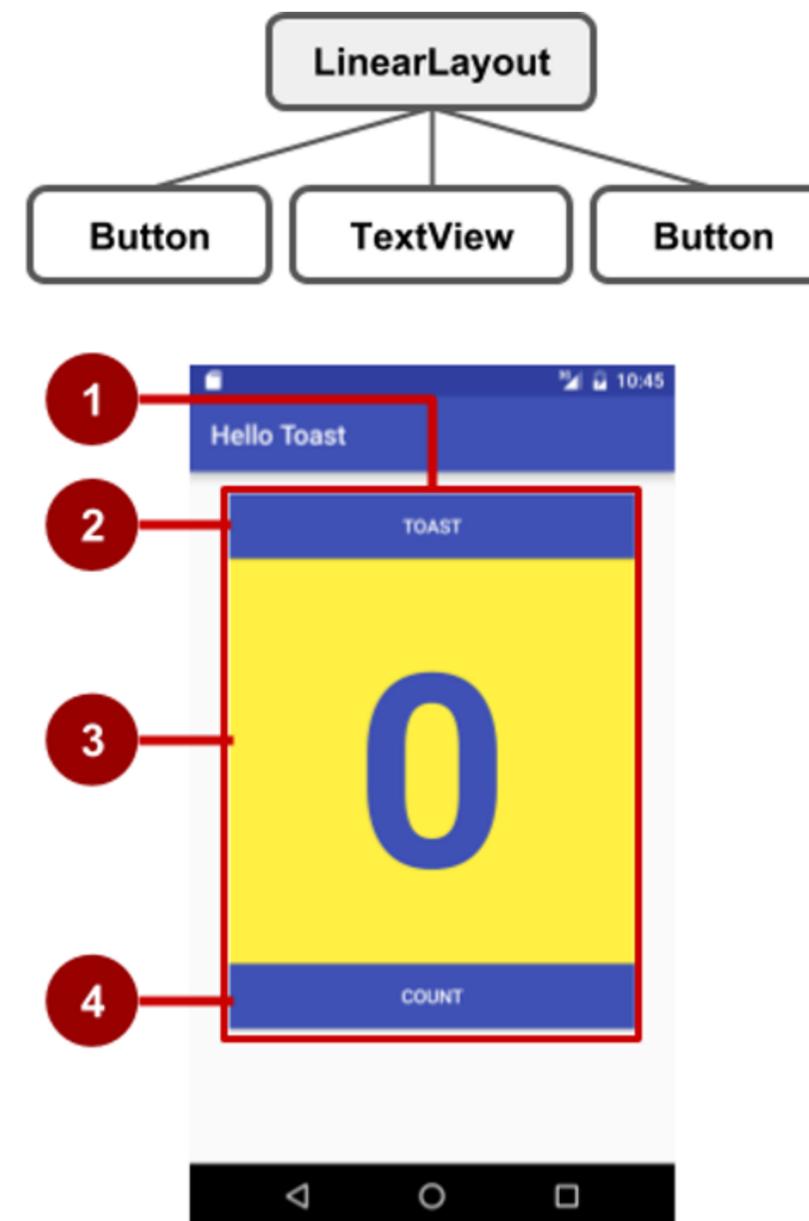
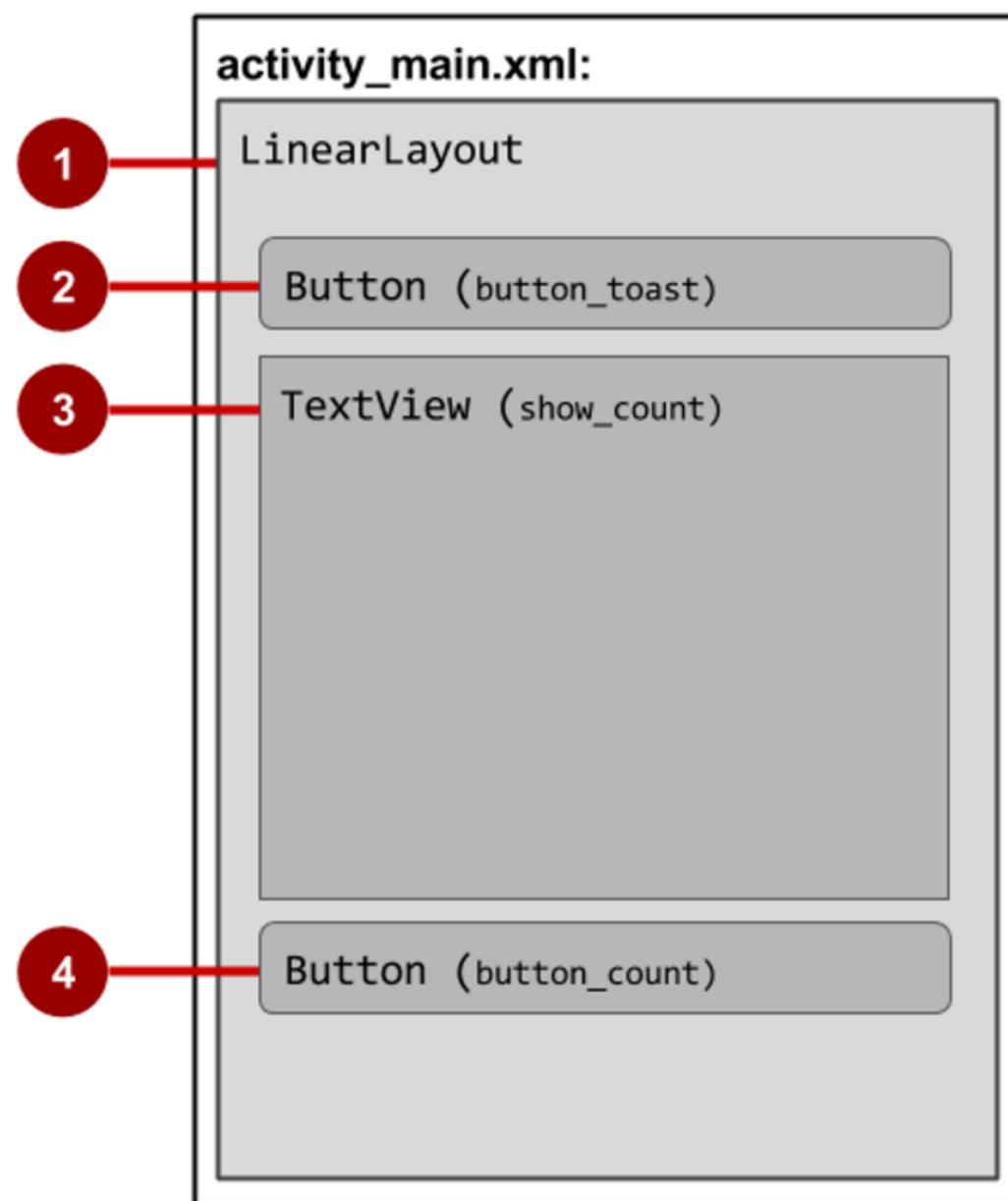
- 视图类的层次结构是标准的面向对象中的**类继承**
 - 例如, Button is-a TextView is-a View is-an Object
 - 上述是is-a的关系
- 布局的层次结构则是描述**视图如何排列**
 - 例如, LinearLayout可以包含排列在一行的多个Button
 - 上述是has-a的关系



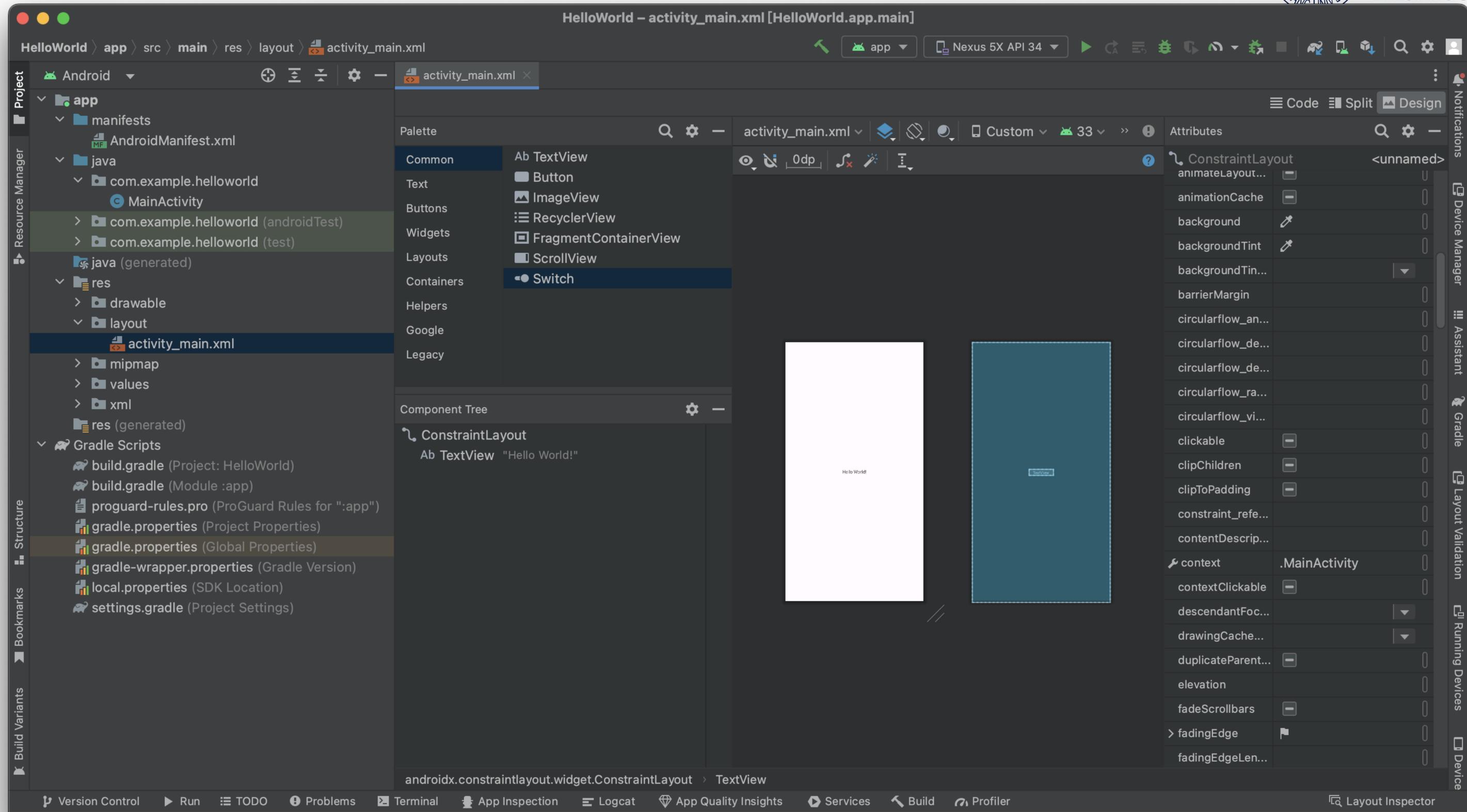
视图组和视图的层次关系



视图的层次性和屏幕布局



在Layout Editor中查看视图的层次结构



创建布局

- 使用XML

```
<LinearLayout
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">
  <Button
    ... />
  <TextView
    ... />
  <Button
    ... />
</LinearLayout>
```

- 使用Java

```
LinearLayout linearL = new LinearLayout(this);
linearL.setOrientation(LinearLayout.VERTICAL);
TextView myText = new TextView(this);
myText.setText("Display this text!");
linearL.addView(myText);
setContentView(linearL);
...
//Set the width and height of a view:
LinearLayout.LayoutParams layoutParams =
    new LinearLayout.LayoutParams(
        LinearLayout.LayoutParams.MATCH_PARENT,
        LinearLayout.LayoutParams.MATCH_CONTENT);
myView.setLayoutParams(layoutParams);
```

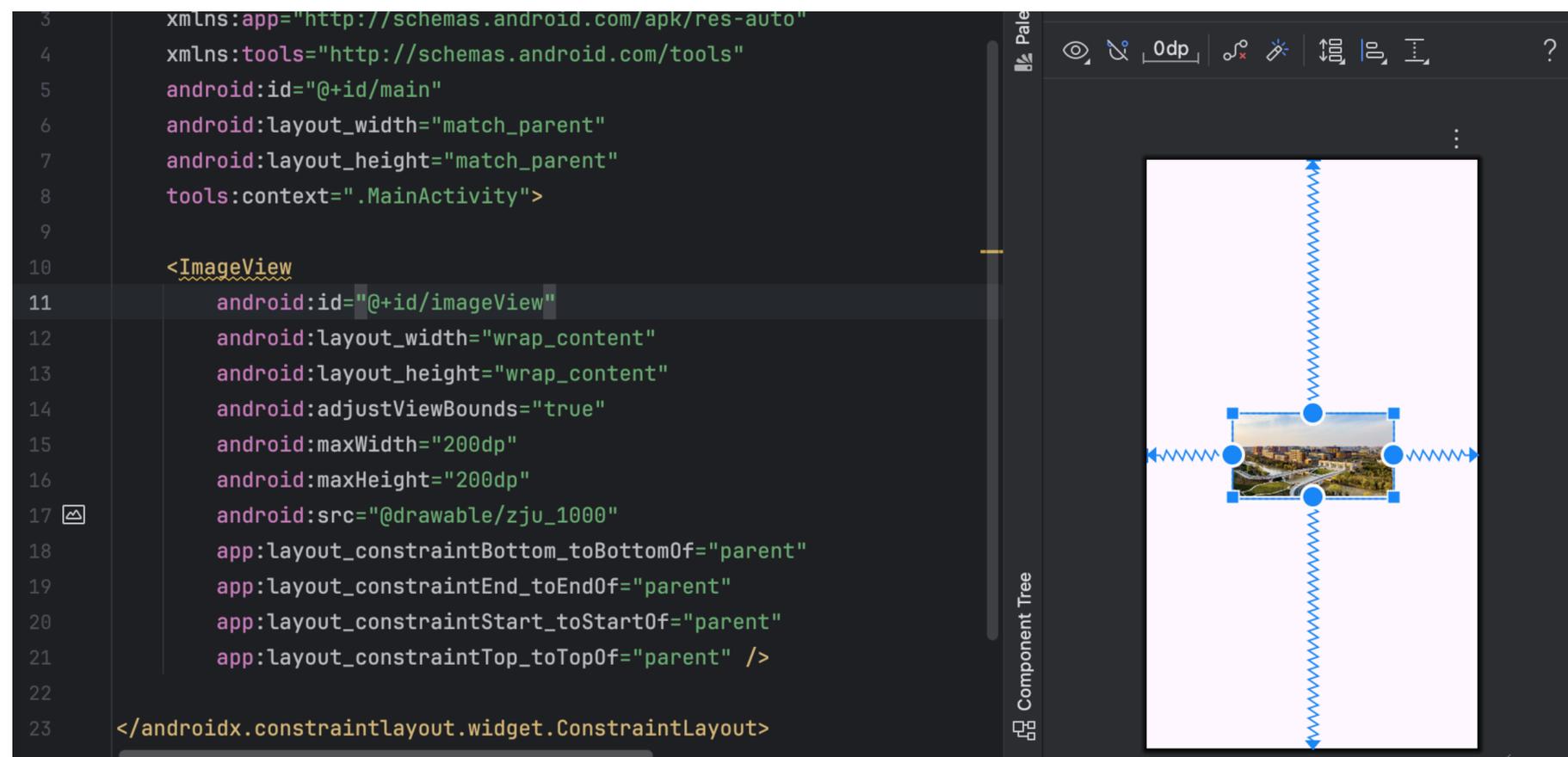


- 资源是指代码使用的附加文件和静态内容，例如位图、布局定义、界面字符串、动画说明等
- 资源主要分为两类，一类存放在`res`目录下，另一类存放在`assets`目录下



res目录下的资源

- 这类资源可以进行读写，也可以通过文件管理器直接访问，在项目内部则通过一个分配给它的id调用
- 这类资源在.xml是通过”@资源类型/资源名称”进行访问的
- 右边是一个例子：左边显示使用res资源时的.xml代码，右边则是显示的预览



res目录下的资源

- 这类资源在Java/Kotlin文件是通过**R.资源类型.资源名称**进行访问的

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        EdgeToEdge.enable($this.enableEdgeToEdge: this);  
        setContentView(R.layout.activity_main);  
        ImageView imageView = findViewById(R.id.imageView);  
        imageView.setImageResource(R.drawable.zju_1000);  
    }  
}
```



res目录下的资源



- res目录下面的资源一般可以分为以下类型
 - animator: 动画
 - anim: 渐变动画
 - color: 颜色
 - mipmap: 不同启动器图标密度的可绘制对象文件
 - drawable: 位图文件
 - layout: 用户界面布局
 - menu: 菜单
 - raw: 各类原始文件
 - values: 字符串, 整型, 颜色。特点是每一个xml都被视为一种资源类型
 - xml: 可在运行时通过调用 `Resources.getXML()` 读取的任意XML文件。各种XML配置文件
 - font: 字体



Assets目录下的资源

- 这类资源是只读的，只能通过AssetManager类来调用
- 使用String[] list(String path)来查看资源
- 使用InputStream open(String fileName, int accessMode)或者InputStream open(String fileName)来打开文件
- Assets目录下主要存放文本，网页，图像，音频视频文件



小结

- 创建视图的三种方式
 - 方法一：Android Studio **Layout Editor**（类似xcode的可视化编辑）
 - 方法二：通过编写**XML代码**（类似HTML）
 - 方法三：通过**Java/Kotlin代码**创建
- 视图组与视图的层次性
- 资源
 - **res**资源（读写）：XML“**@资源类型/资源名称**”，代码**R.资源类型.资源名称**
 - **assets**资源（只读）



一些学习资料



浙江大学
ZHEJIANG UNIVERSITY

- [Meet Android Studio](#)
- 安卓官方文档 developer.android.com
- [Create and Manage Virtual Devices](#)
- [Supporting Different Platform Versions](#)
- [Supporting Multiple Screens](#)
- [Gradle Wikipedia page](#)
- [Google Java Programming Language style guide](#)
- 在StackOverflow查找常见问题 [Stackoverflow.com](https://stackoverflow.com)





Thanks!