浙江大学
ZHEJIANG UNIVERSITY

Android应用基础

陶煜波

课程目录

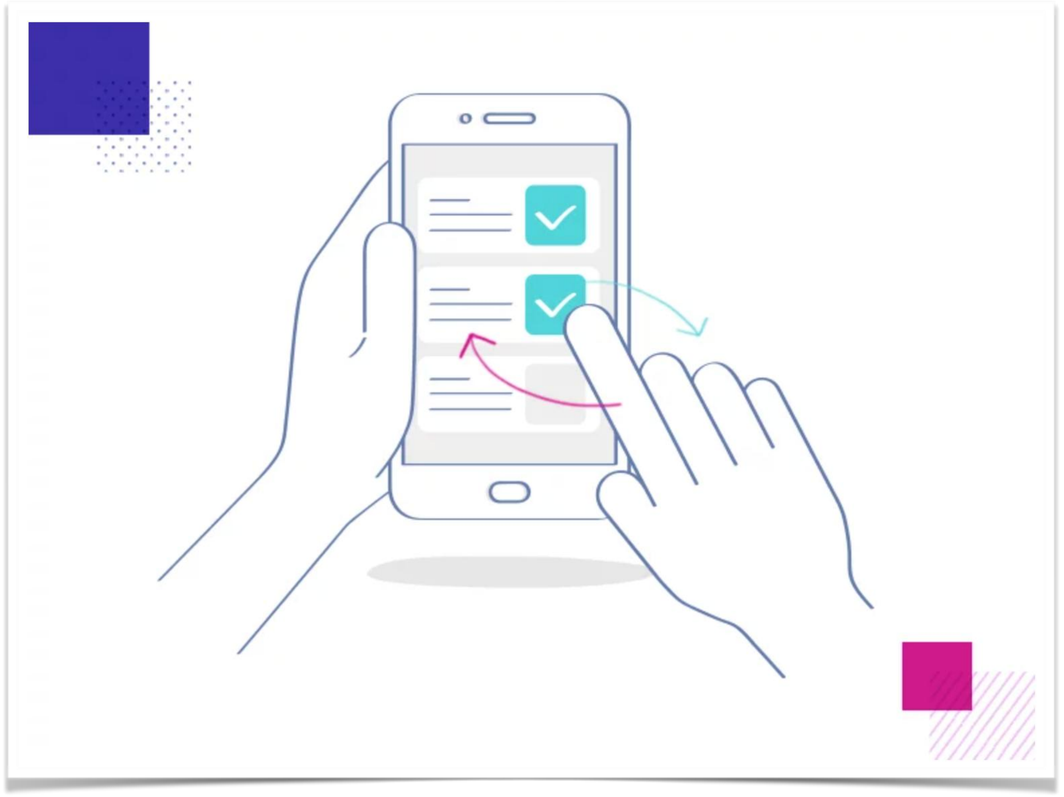
浙江大学 ZHEJIANG UNIVERSITY

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
- 截至2025年，Android是全球移动设备市场最主流的操作系统，应用生态呈现“数量收缩、质量提升”的趋势
- 截止2025年Q3，Android占据了移动OS市场的79%份额；Google Play上Android应用数量为180万
- 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：布局（线性布局、约束布局、帧布局、表格布局、相对布局等），菜单（选项式、上下文式、上下文操作栏式等）……



Android的传感器

- Andorid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

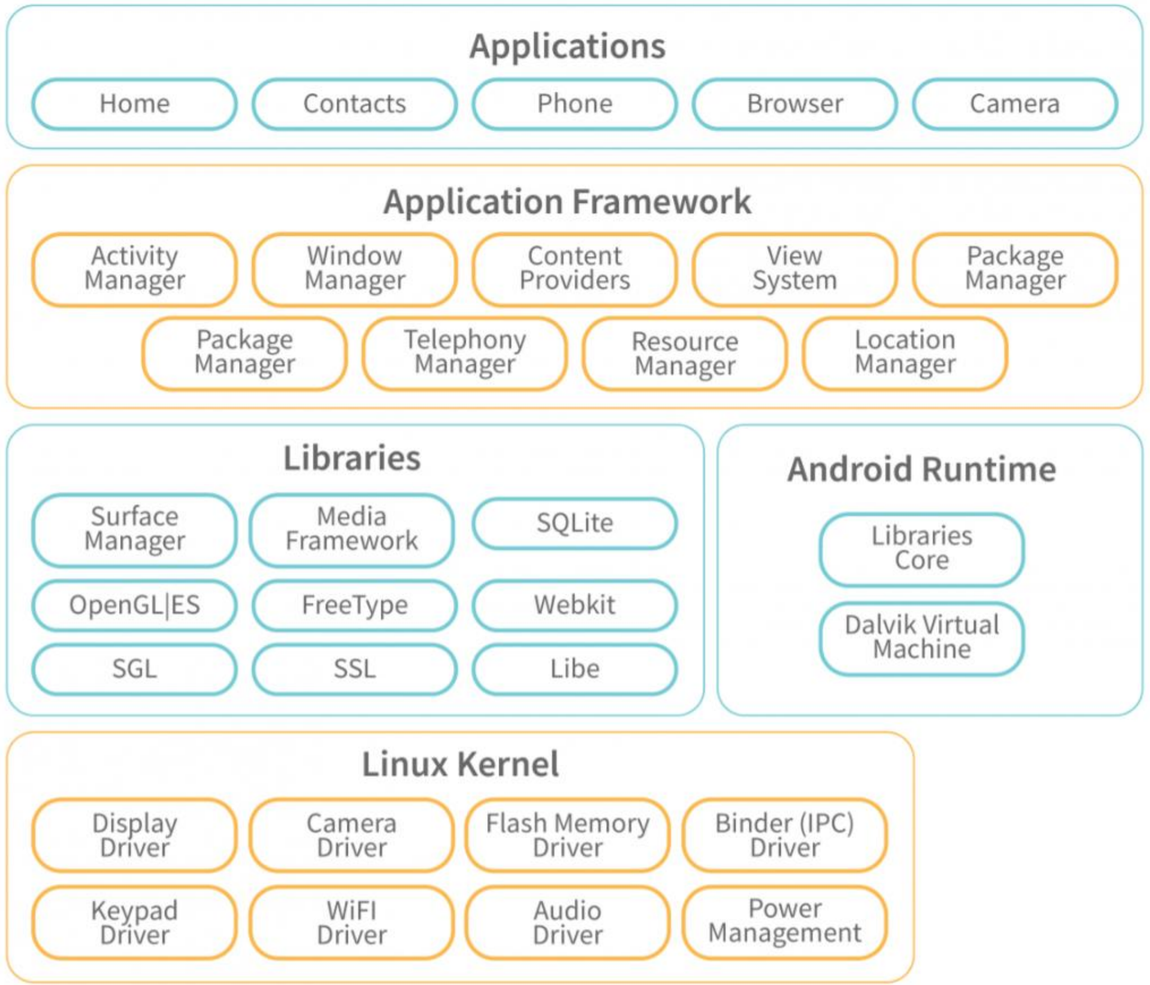
Android操作系统

- 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

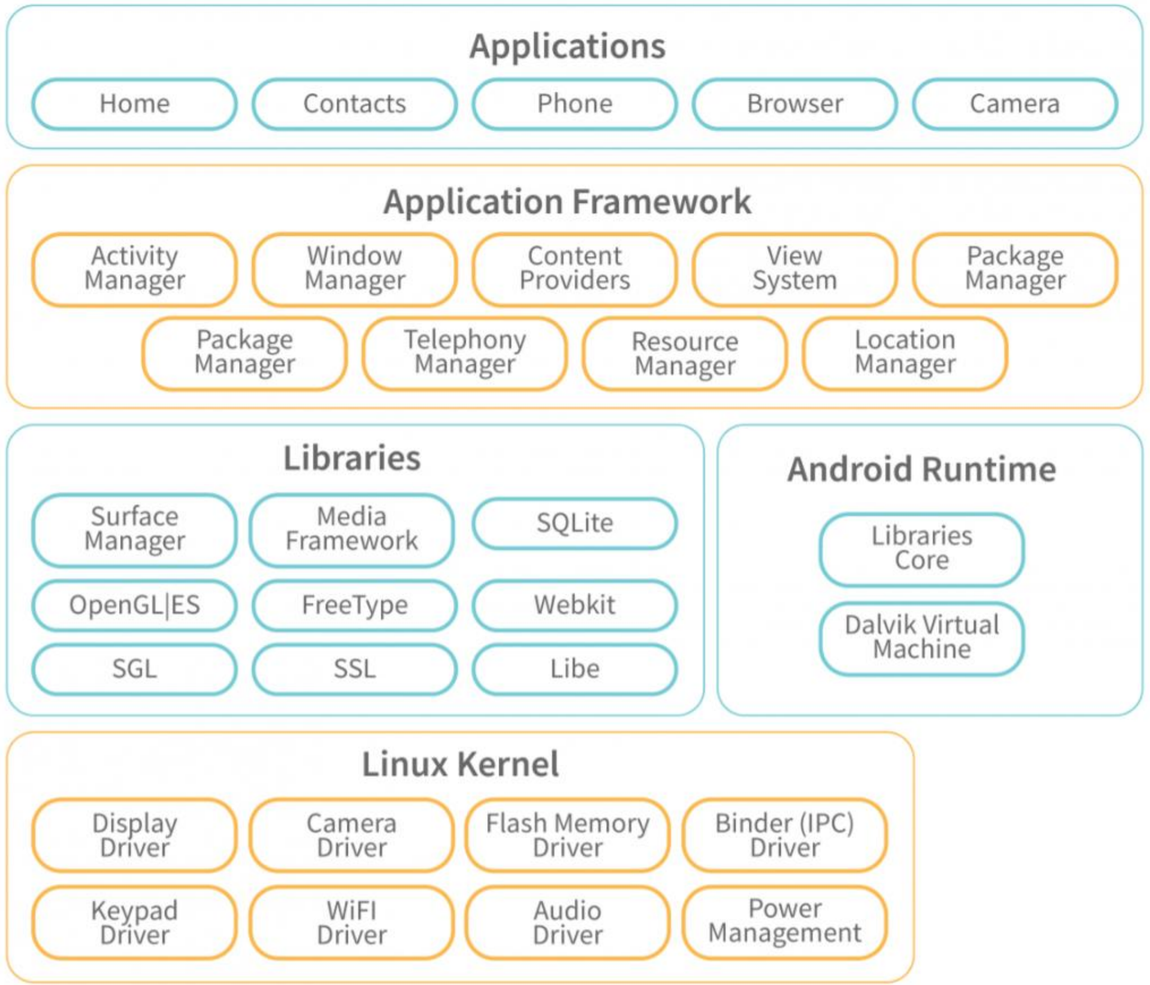
- Linux Kernal：Android平台的基础是Linux内核。上层依赖于Linux内核的底层功能，如内存管理、进程管理、网络栈、多任务。使用Linux内核使Android能够天然具备一些安全特性，同时设备制造商可以基于开源的Linux内核来开发他们的驱动程序



Android开发架构

- 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

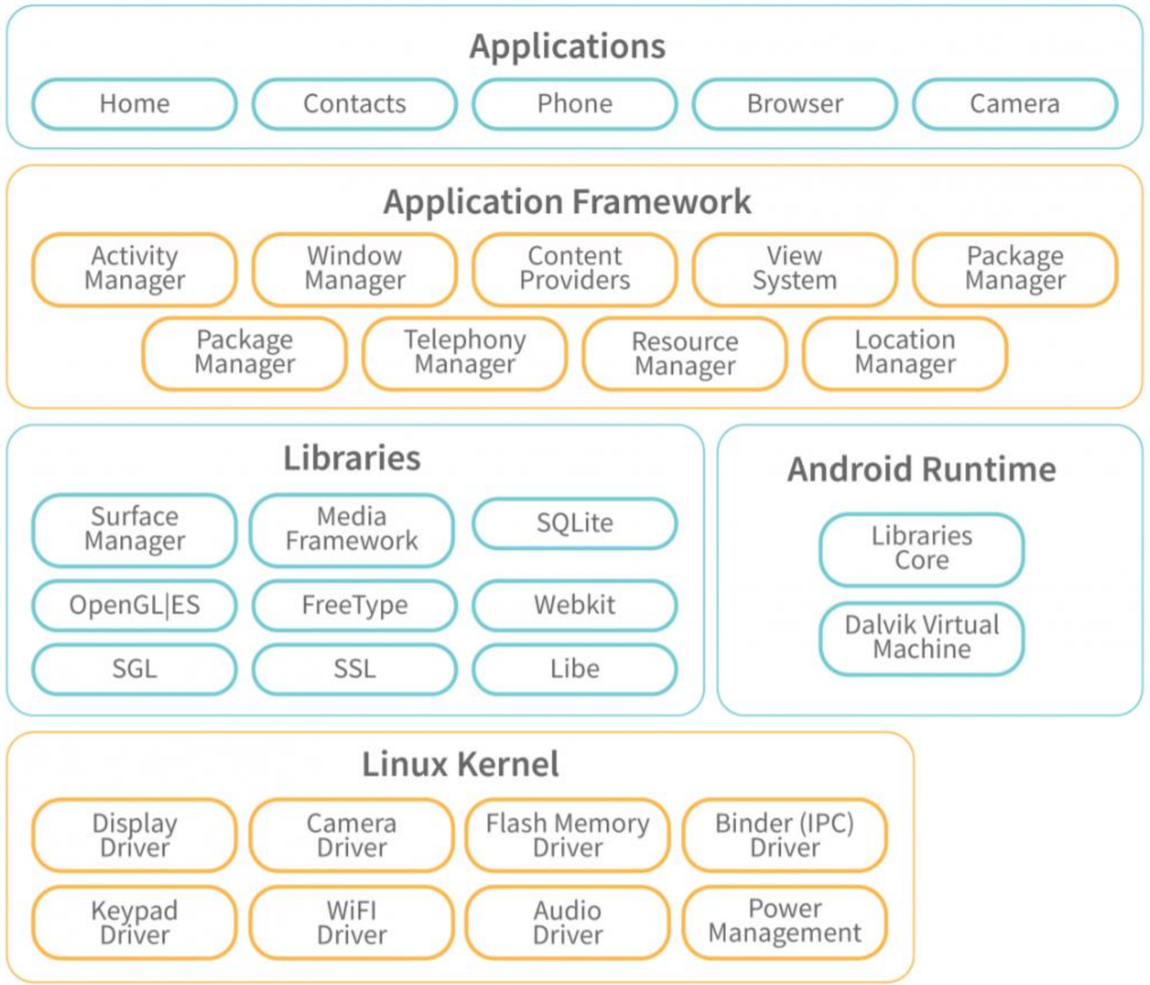
- Android Runtime：包含核心类库和虚拟机，每个应用程序都在自己的进程中运行，并拥有自己的Android Runtime实例



Android开发架构

- Application Framework：通过Java/Kotlin语言编写的应用程序编程接口（API），同时包含了硬件抽象层（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

- Applications：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



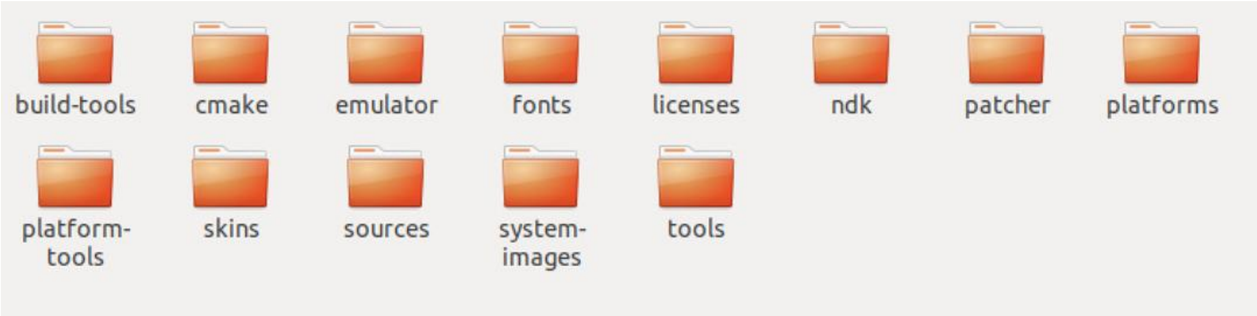
Android应用

- 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一个.apk 后缀的文件中
- apk文件是Android 系统内安装包程序

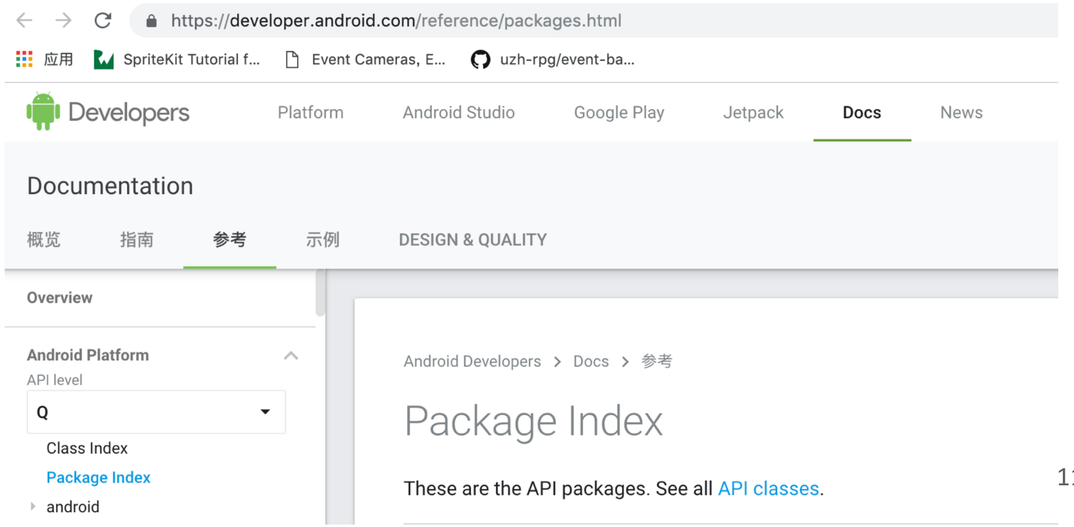
Android Software Development

- 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应用时，我们需要通过引入该开发工具包来使用Android相关的API

- Android SDK官方文档地址：developers.android.com/docs



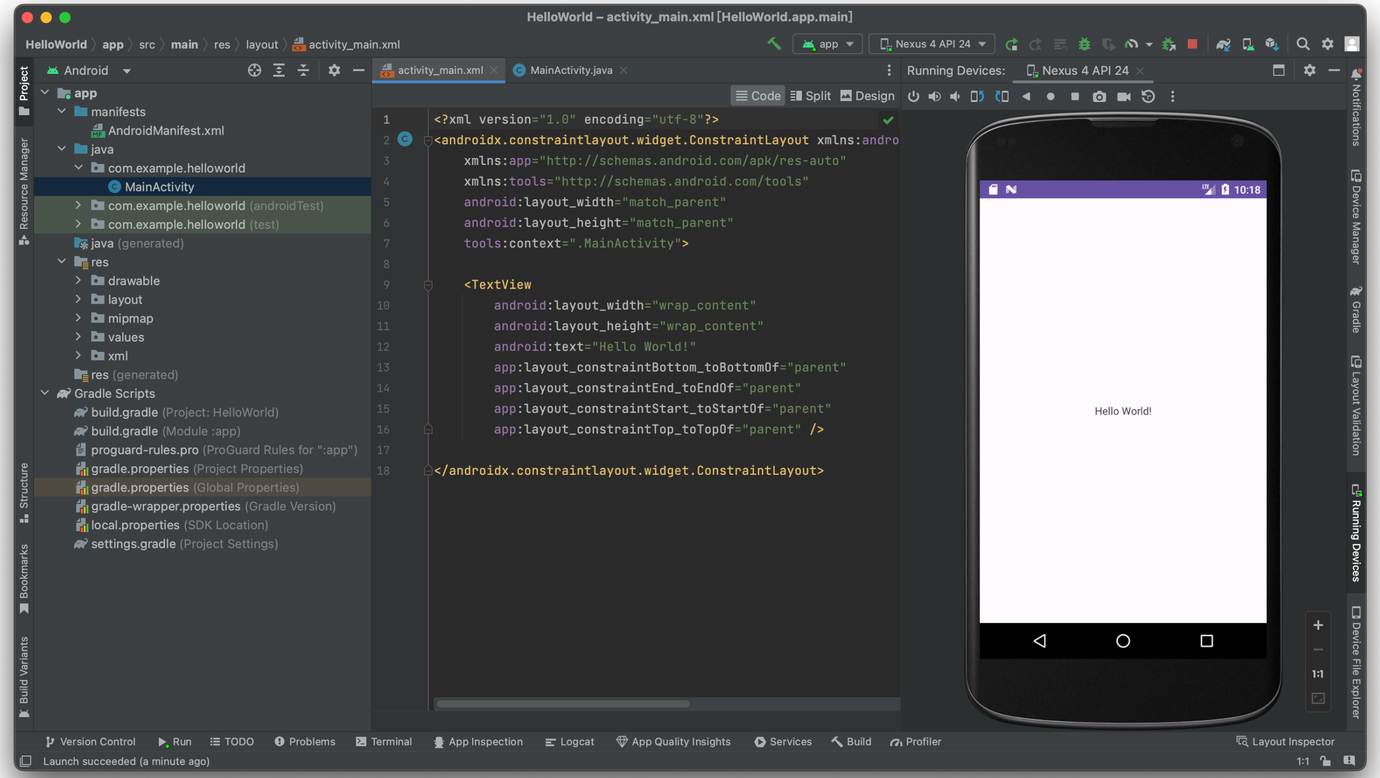
Android SDK 解压后包含的文件、文件夹



Android Studio

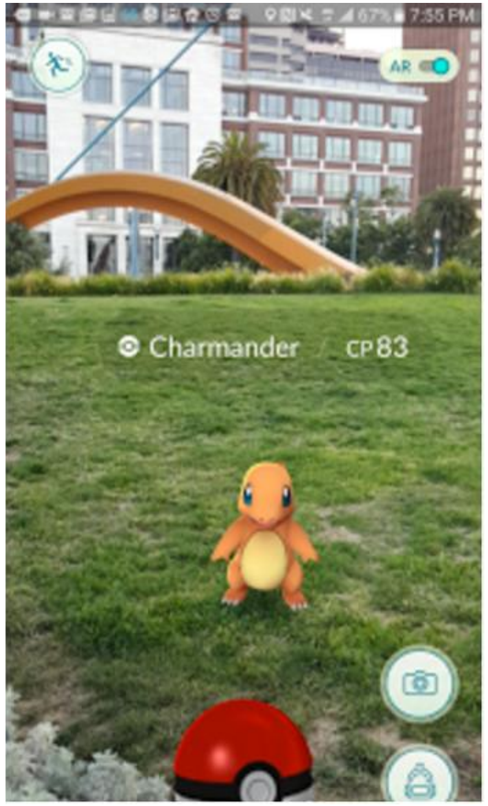
- Android Studio是用于Google Android操作系统的官方集成开发环境（IDE），基于JetBrains的IntelliJ IDEA软件构建，并且专门为Android开发而设计

- 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

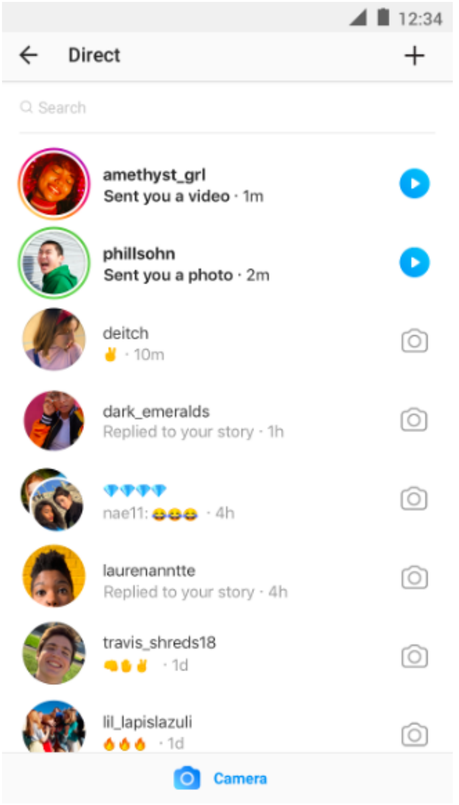


Some Android Apps

Wechat



Pokemon Go



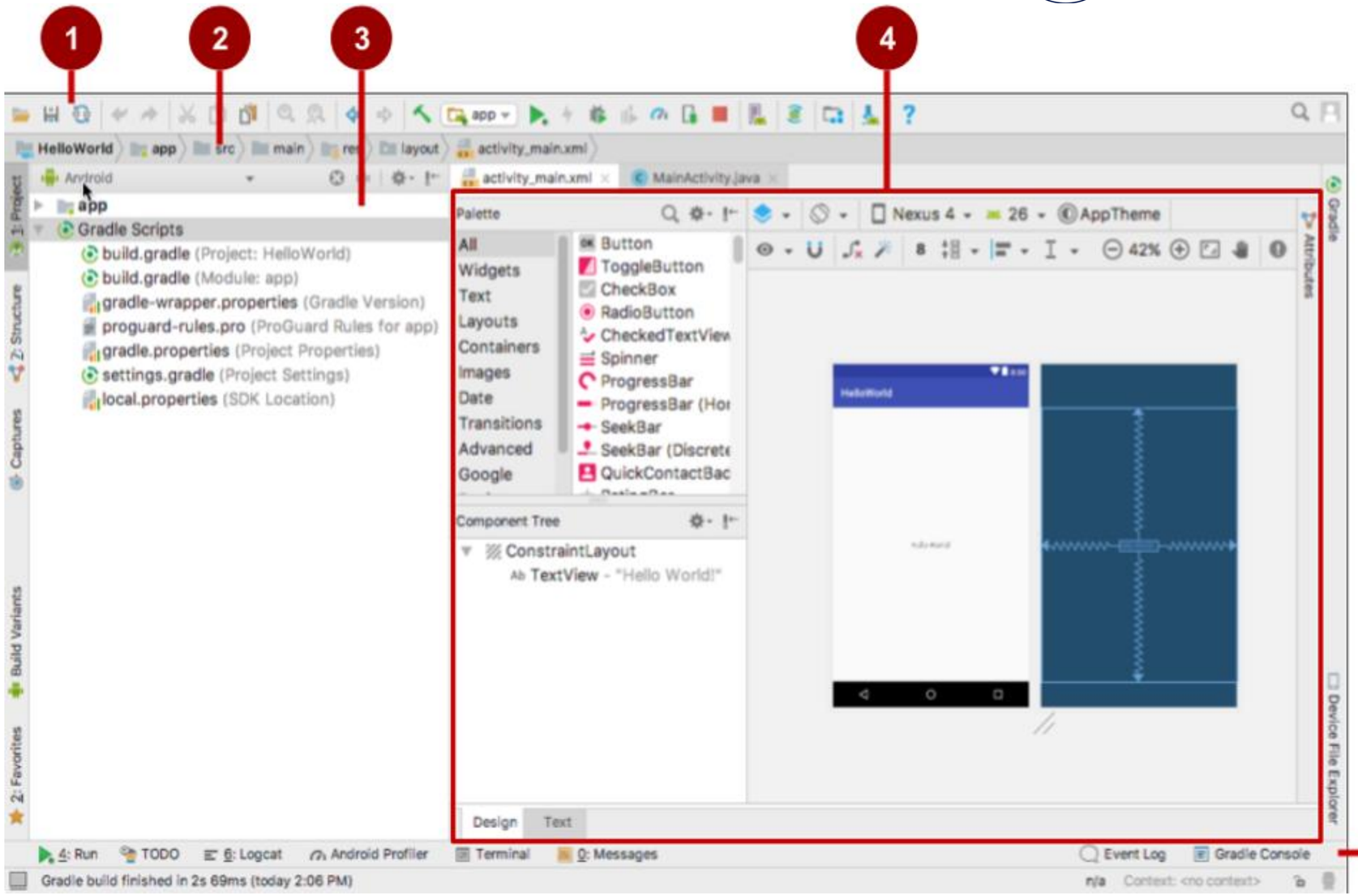
Instgram

课程目录

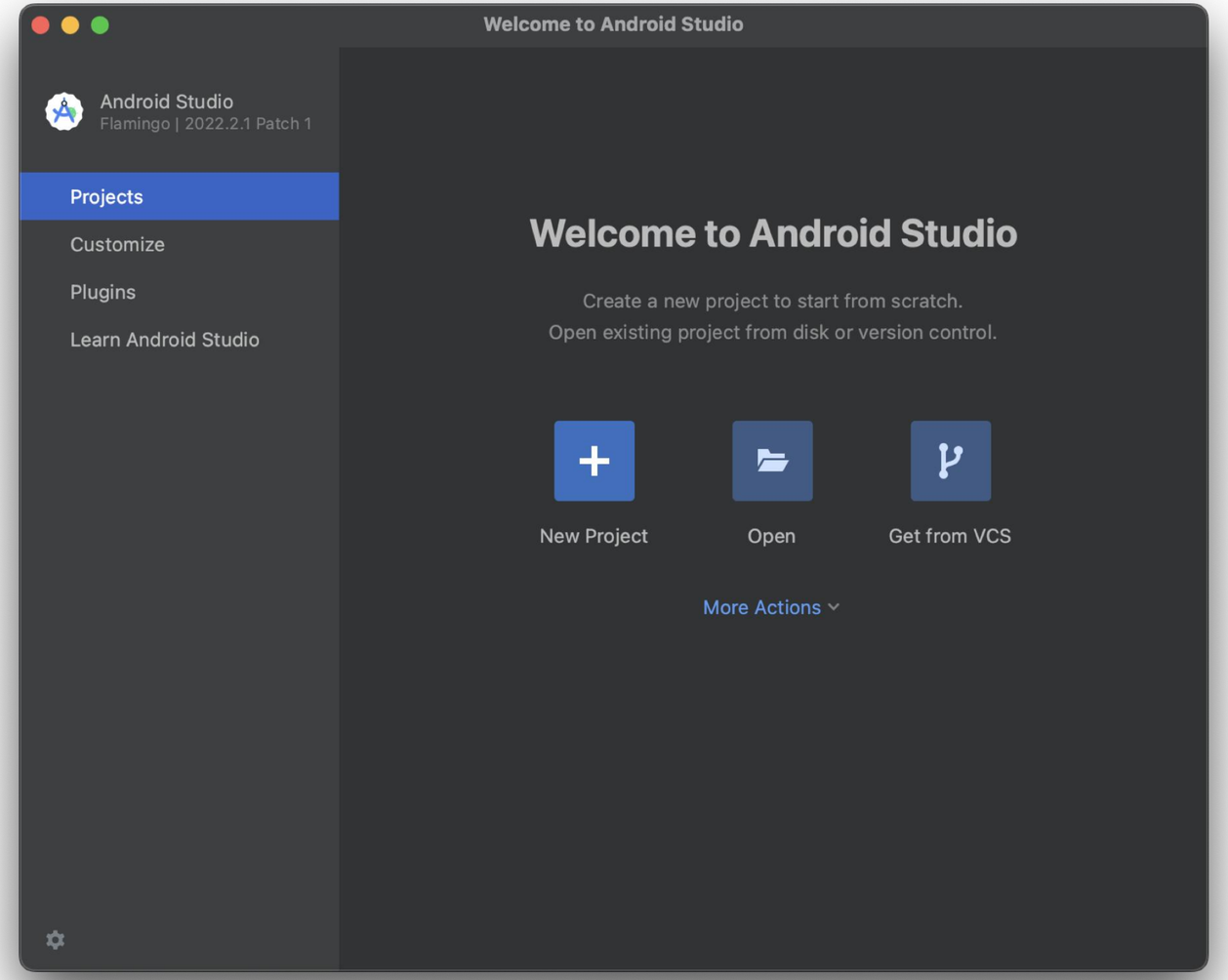
浙江大学 ZHEJIANG UNIVERSITY

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

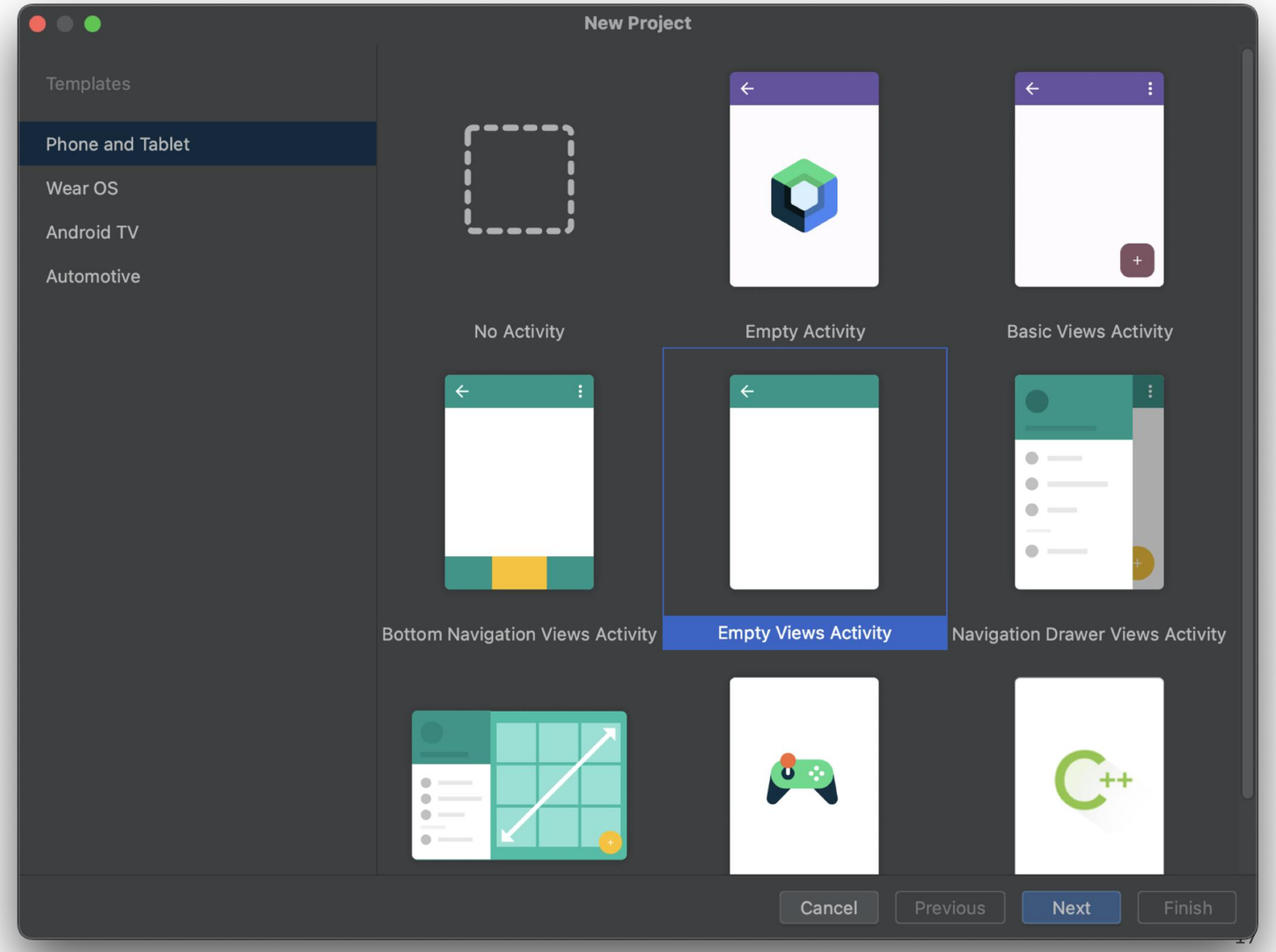


打开Android Studio



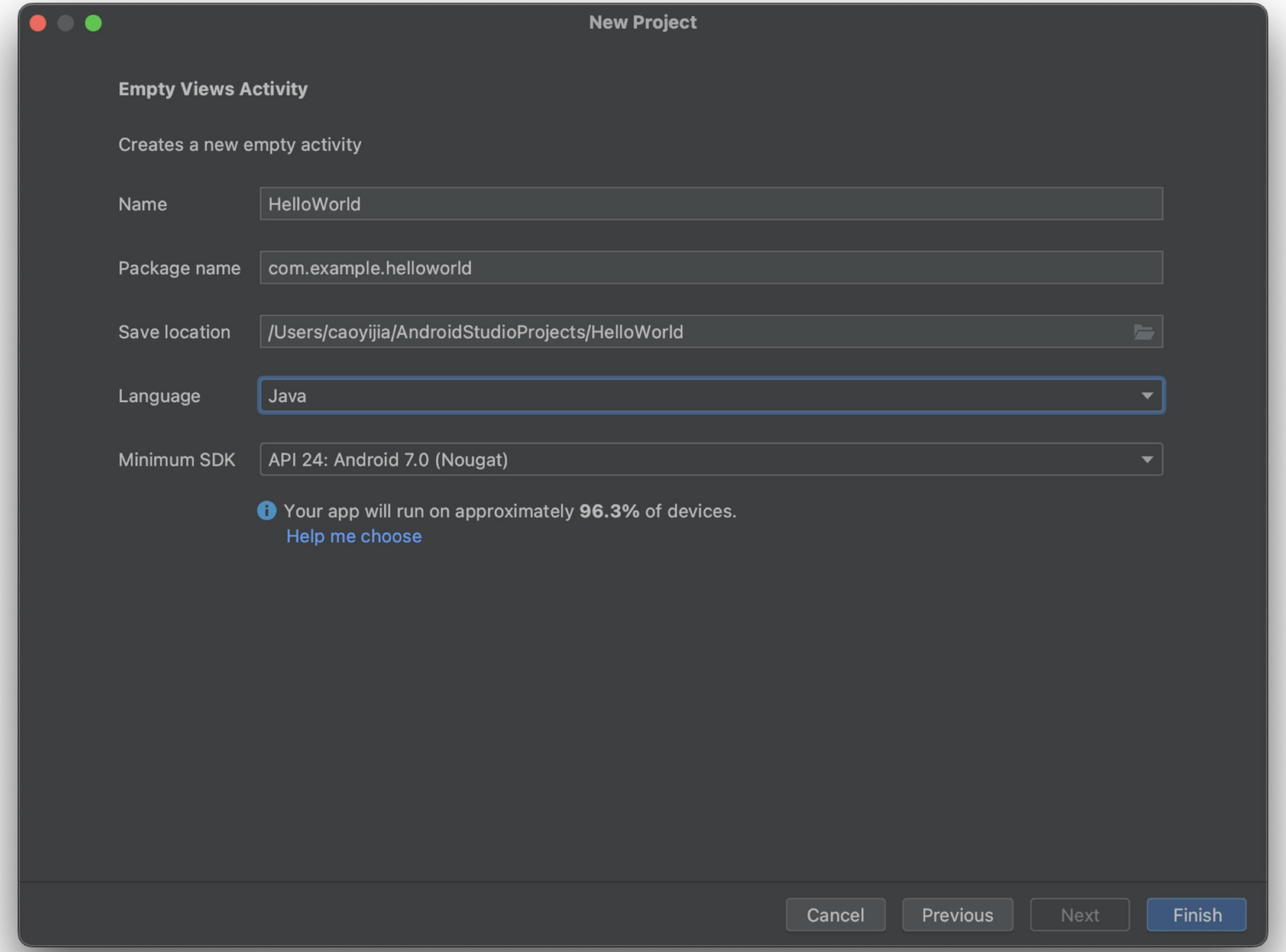
选择项目类型

- 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Empty Views Activity



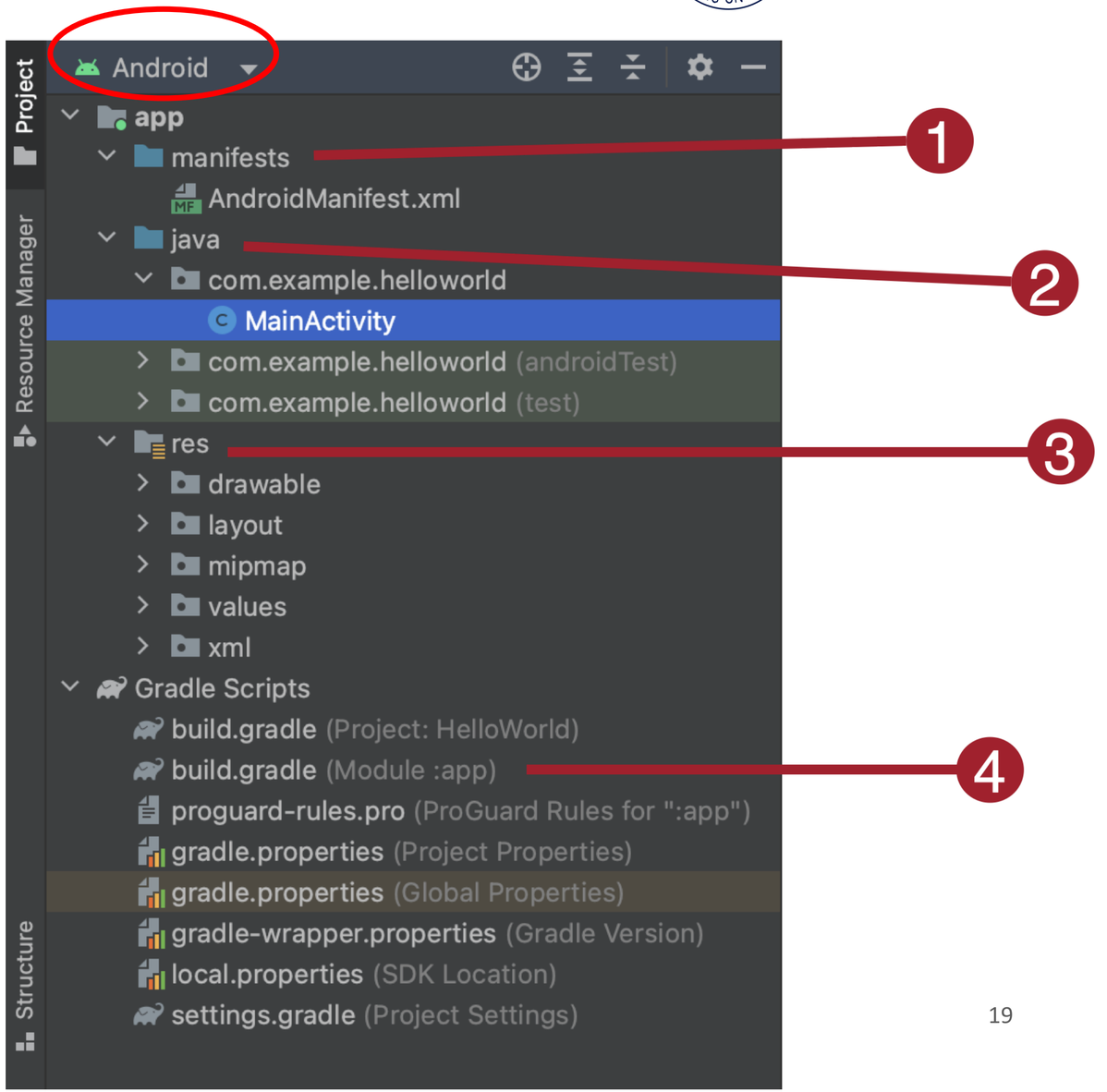
输入项目信息

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



项目文件夹

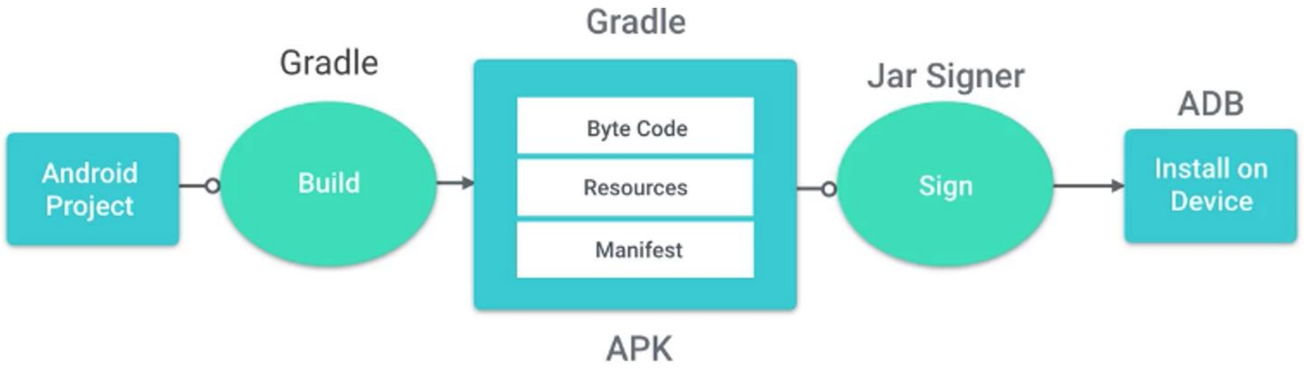
1. manifests — 清单文件（Manifest file）
2. java — Java或Kotlin源代码
3. res — 资源（格式为XML）包括布局，字符串，图片，颜色
4. Gradle Scripts — Gradle 构建文件



Gradle

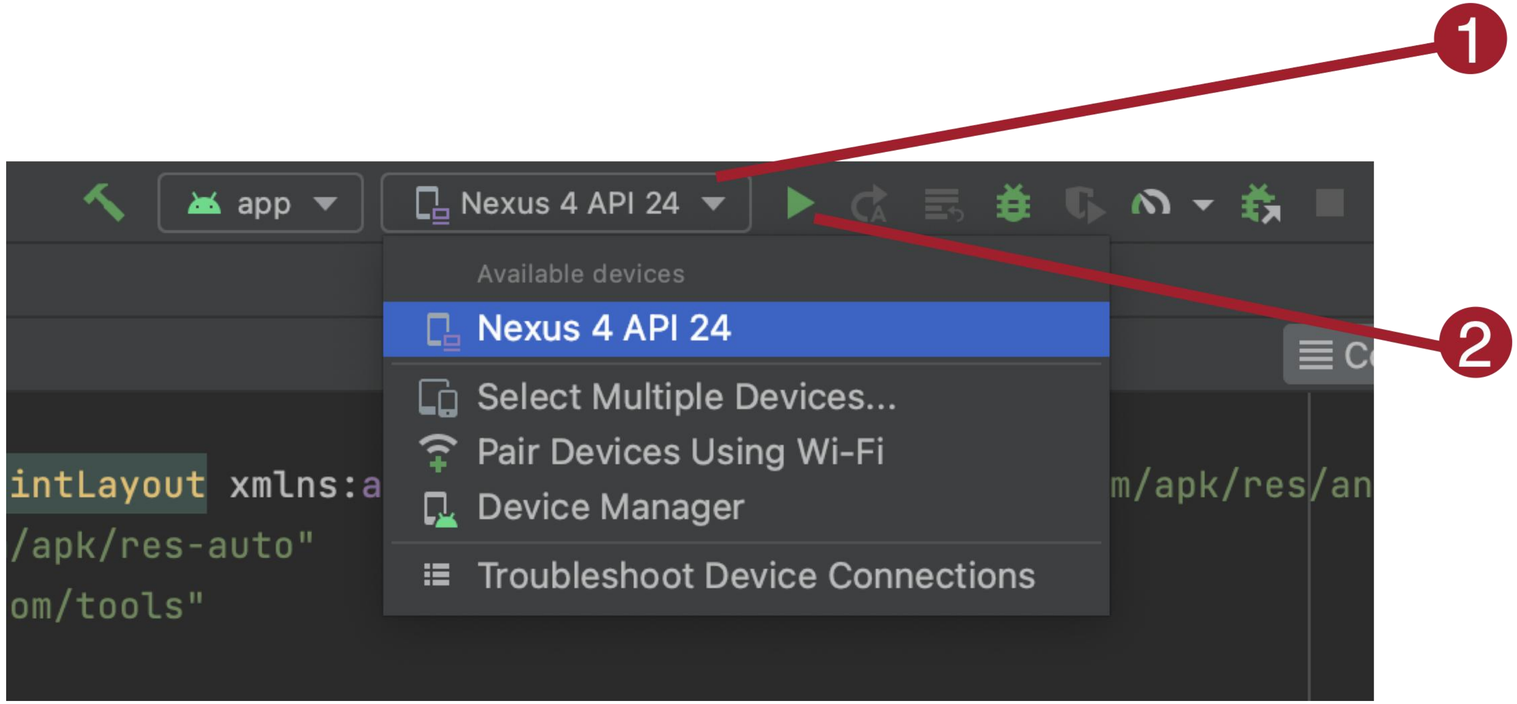
- Gradle是用于多语言软件开发的构建自动化工具。 它控制从编译和打包到测试，部署和发布的任务中的开发过程。 支持的语言包括Java（ Kotlin，Groovy，Scala），C/C ++和JavaScript

- 一个安卓项目中有三个.gradle文件：
  build.gradle(Project)，
  build.gradle(Module)，
  settings.gradle



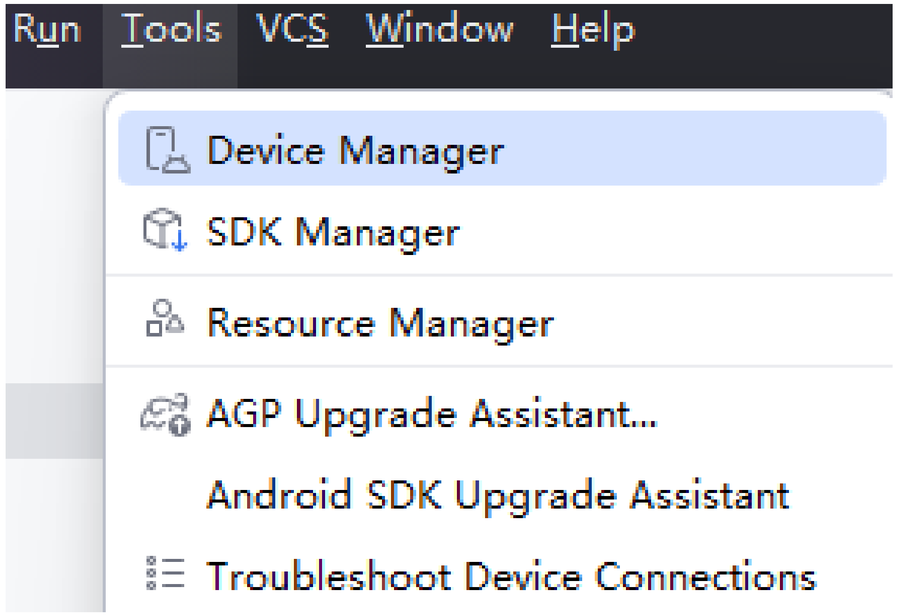
运行项目

1. 选择虚拟设备或者实体设备
2. 运行工程

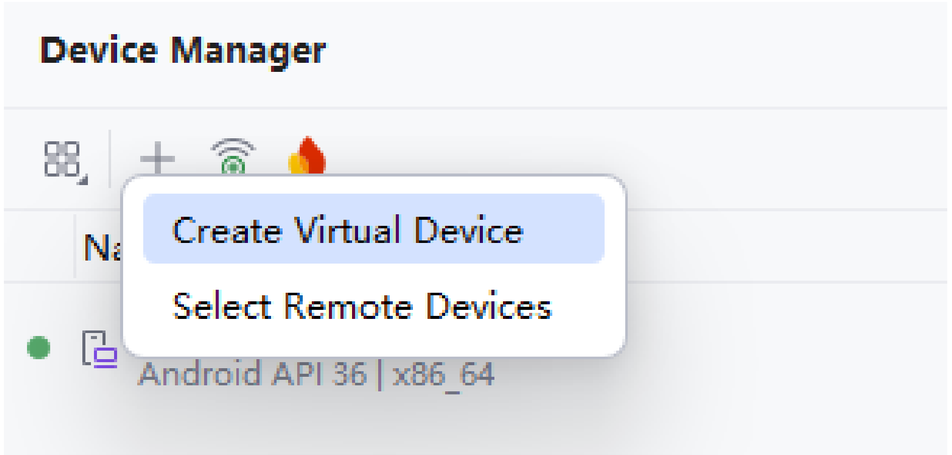


创建虚拟设备

- 点击“Tools → Device Manager”进入设备管理界面

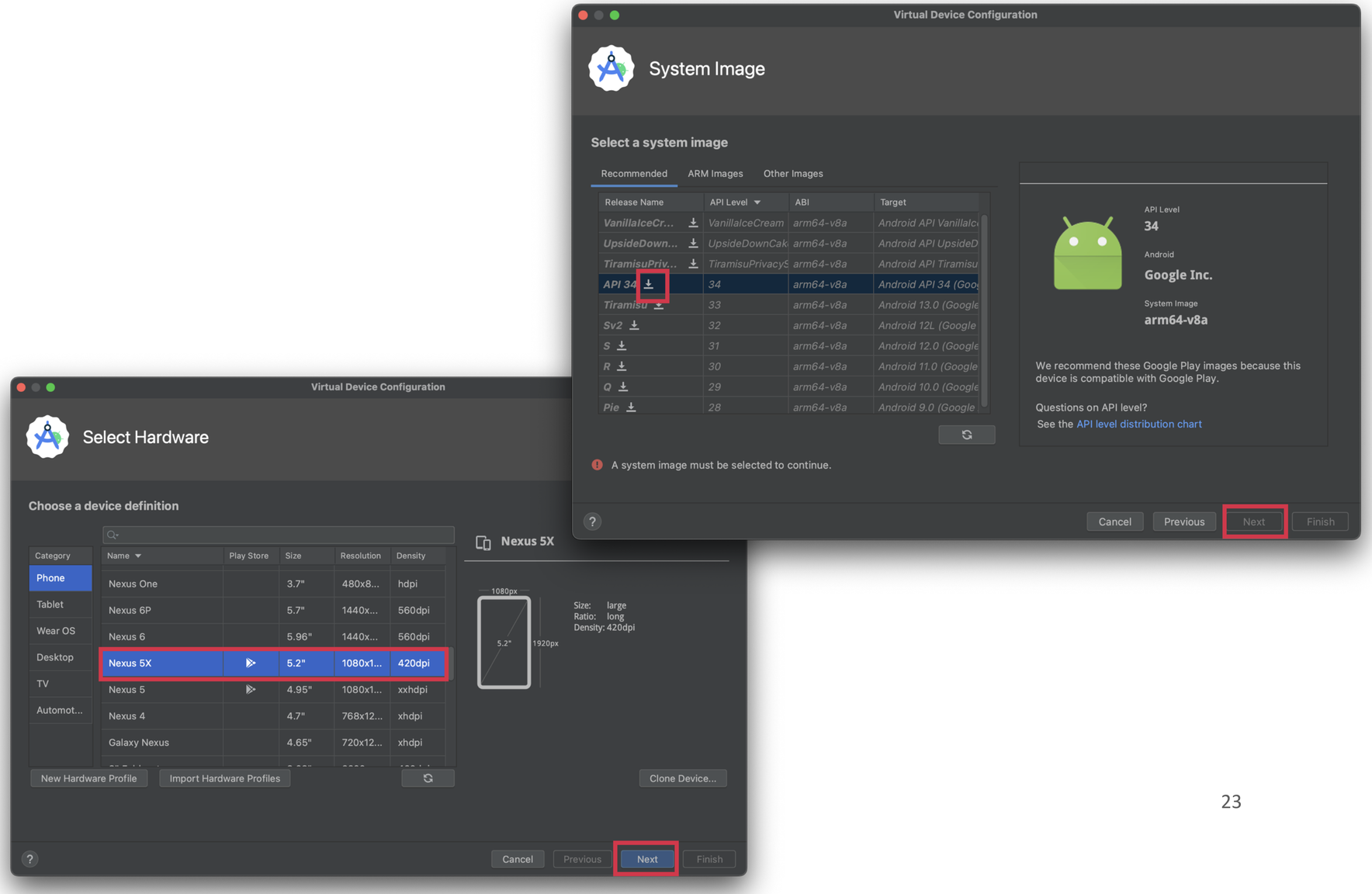


- 点击“Create Virtual Device”创建新的虚拟设备

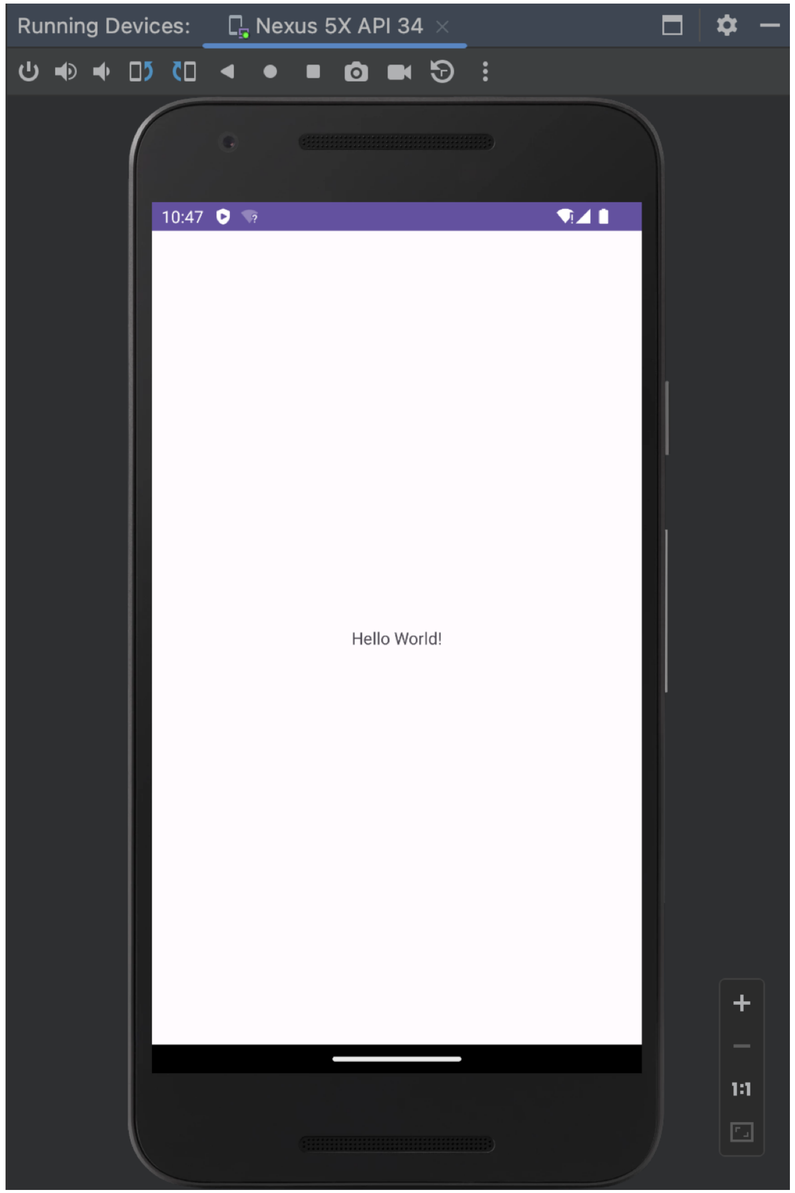


配置虚拟设备

- 选择虚拟硬件设备类型 (例如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

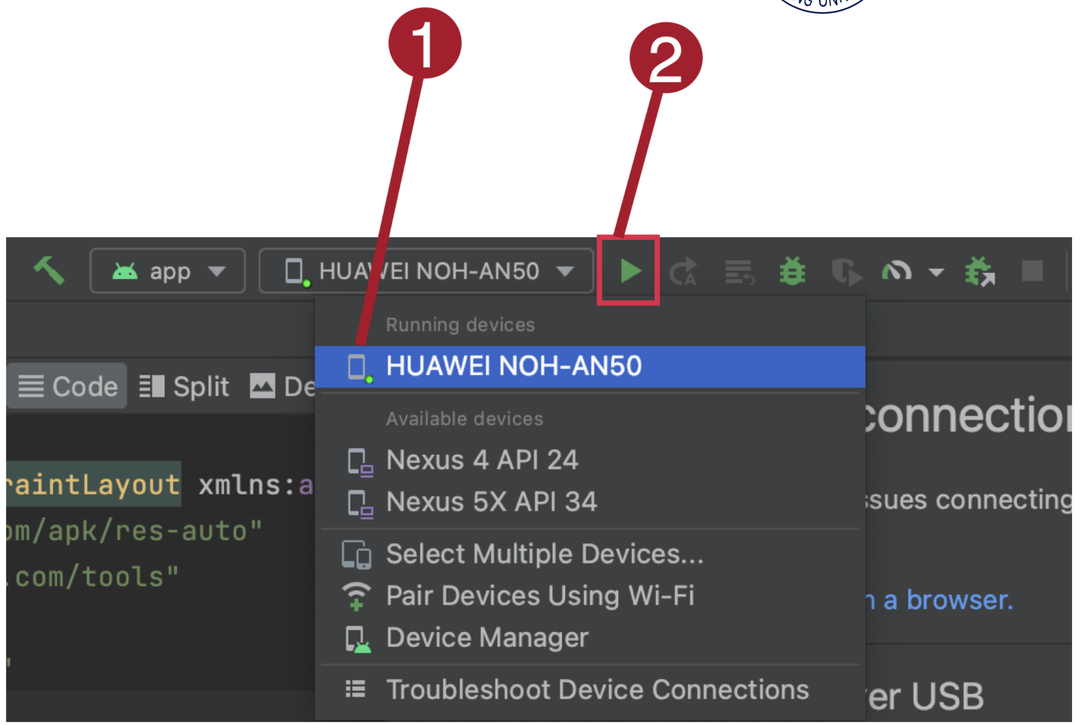


配置实体设备

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

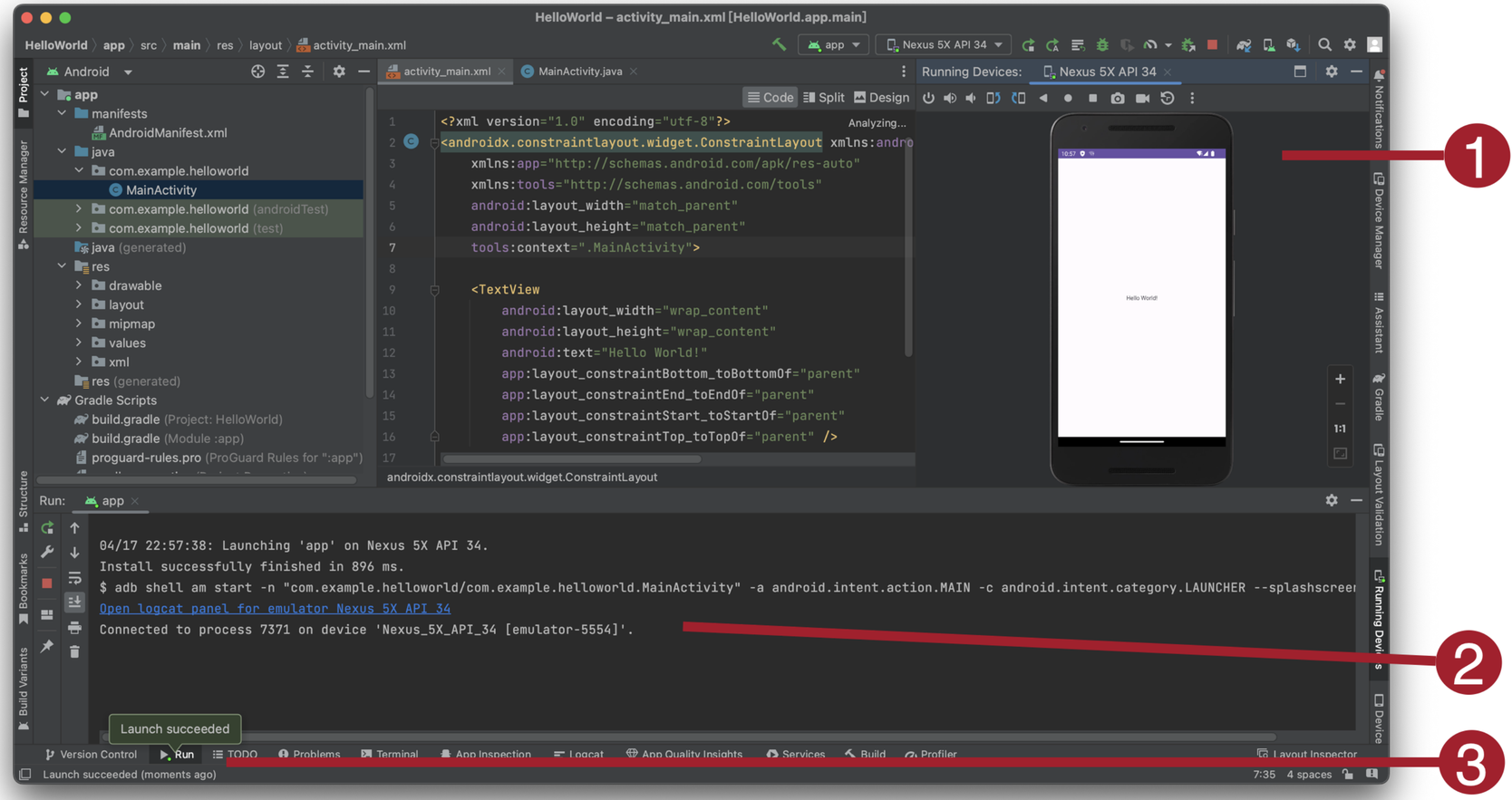
Windows 驱动:
- OEM USB Drivers



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

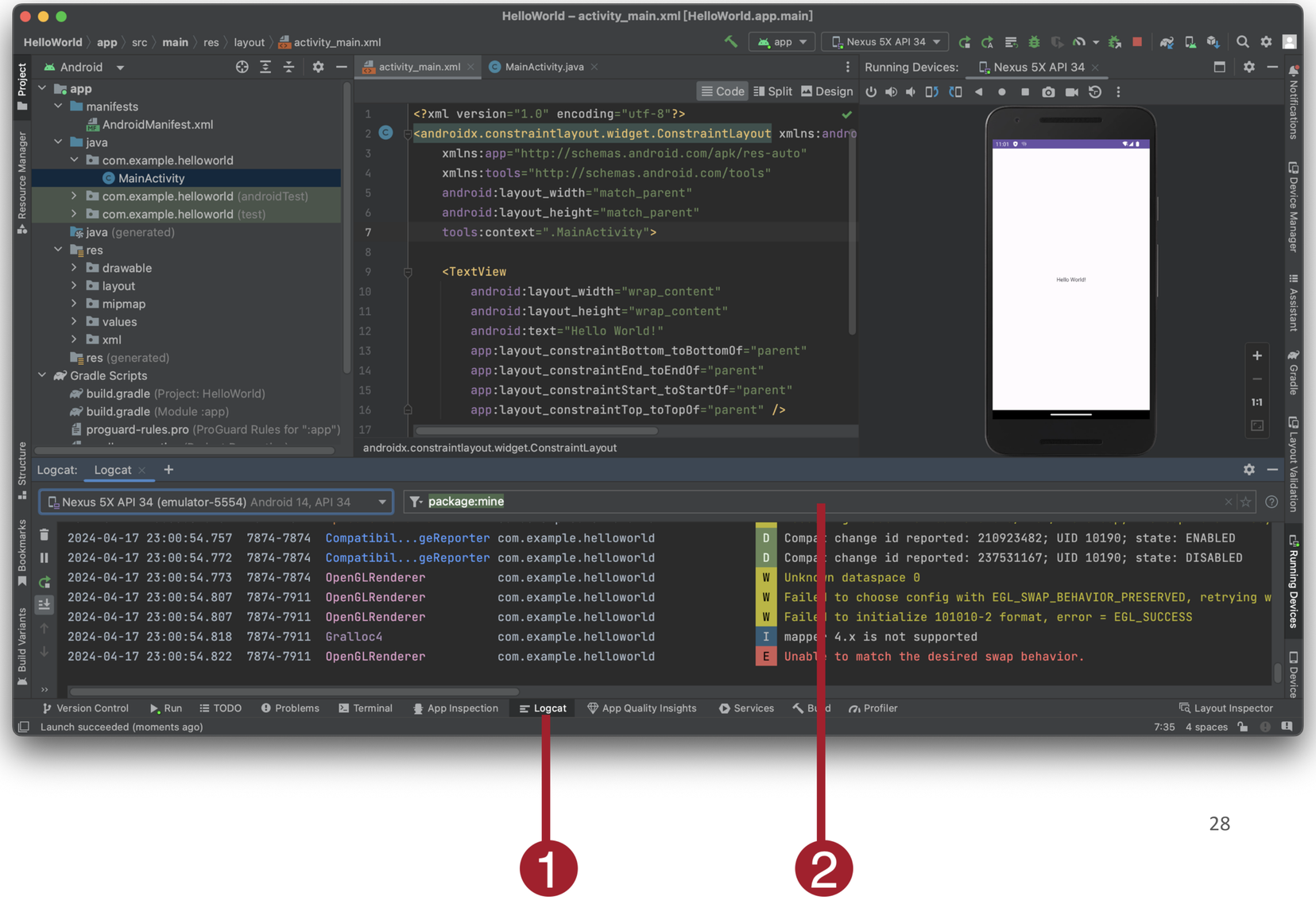


在App中加入日志

- 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…”);
```

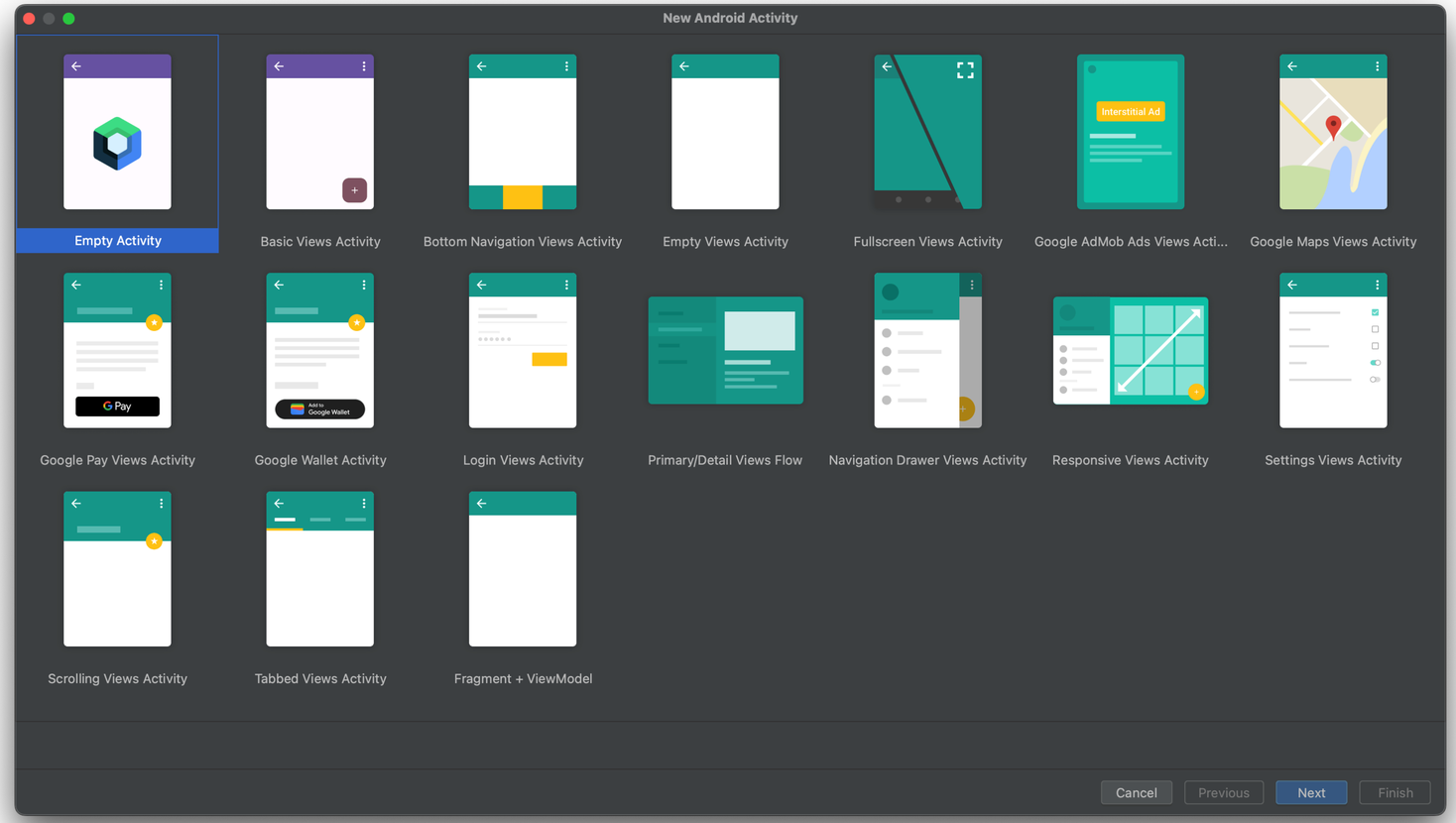
应用组件

- 应用组件是Android应用的基本构建基块。每个组件对应不同功能，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 (Service)，内容提供程序 (Content Provider)，广播接收器 (Broadcast Receiver)

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用户界面的屏幕窗口



Activity模板

一个Activity(单一屏幕) = 一个布局文件(xml) + 一个代码文件(Activity子类)

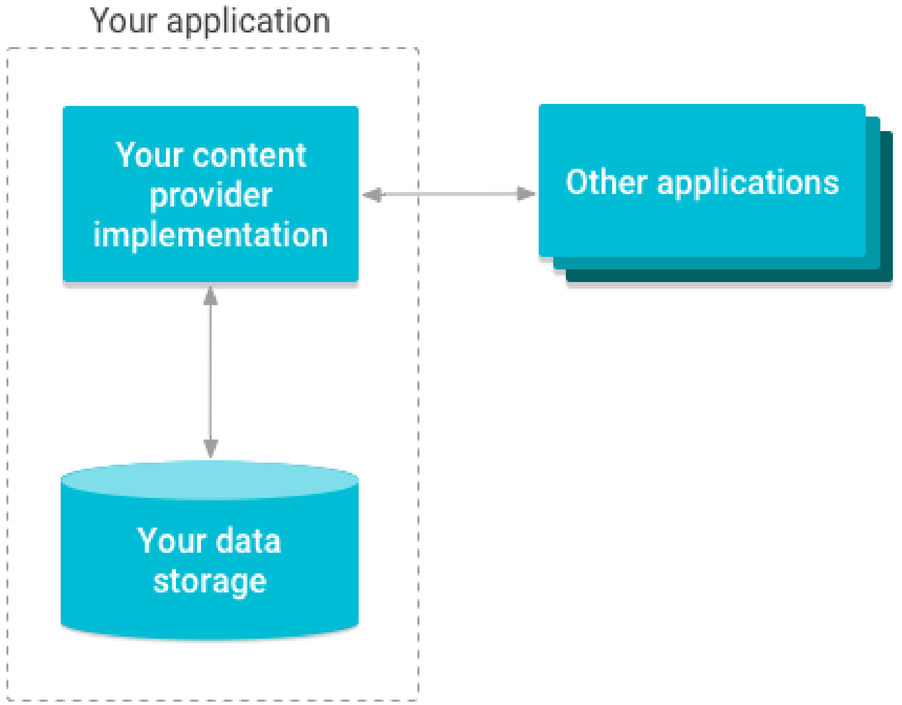
一个App = Activity [+ Activity + … + Activity] + 资源文件 + 清单文件

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，如果想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 (同一App的不同Activity之间，不同App的Activity之间)
- 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

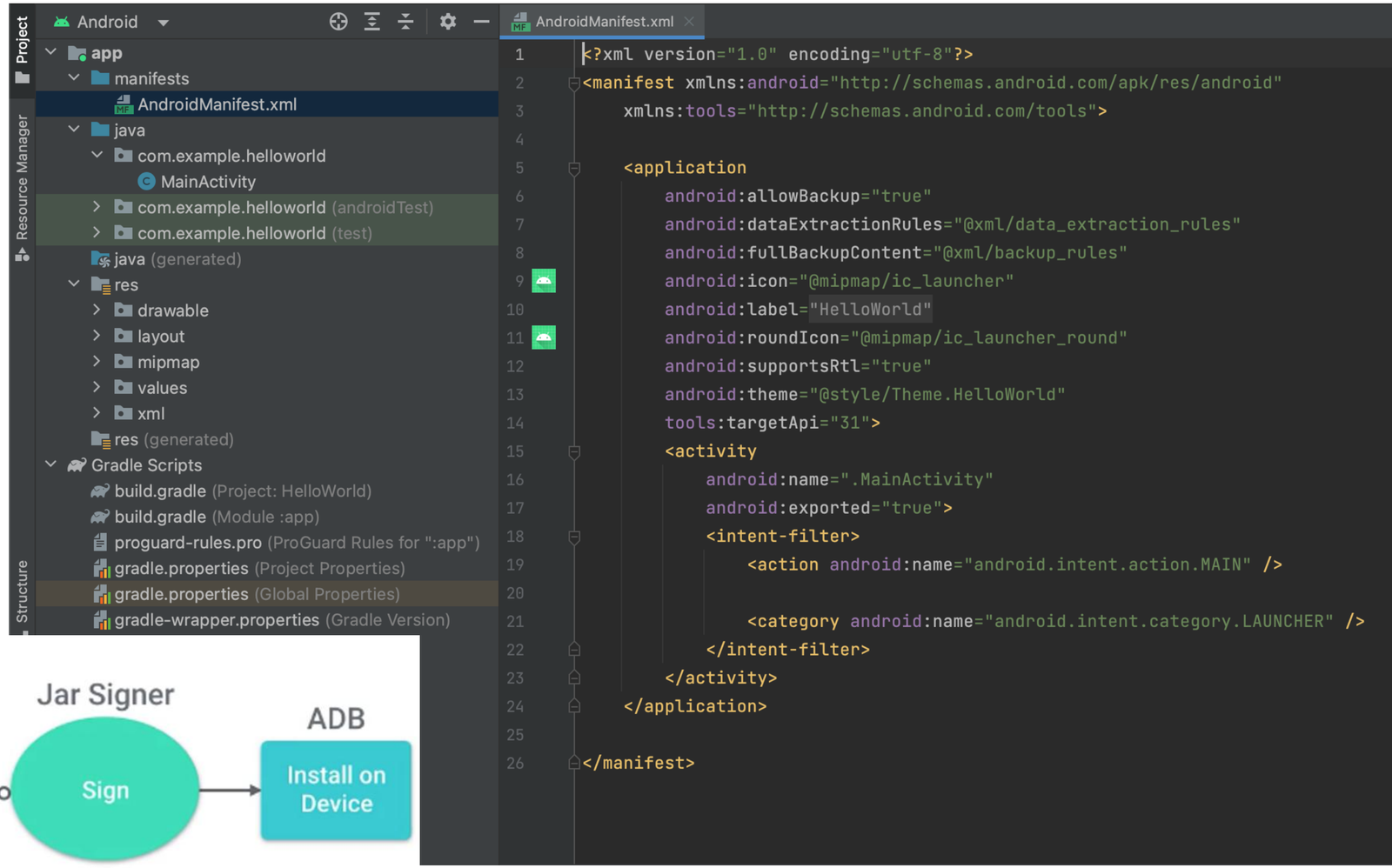
Intent在广播服务器

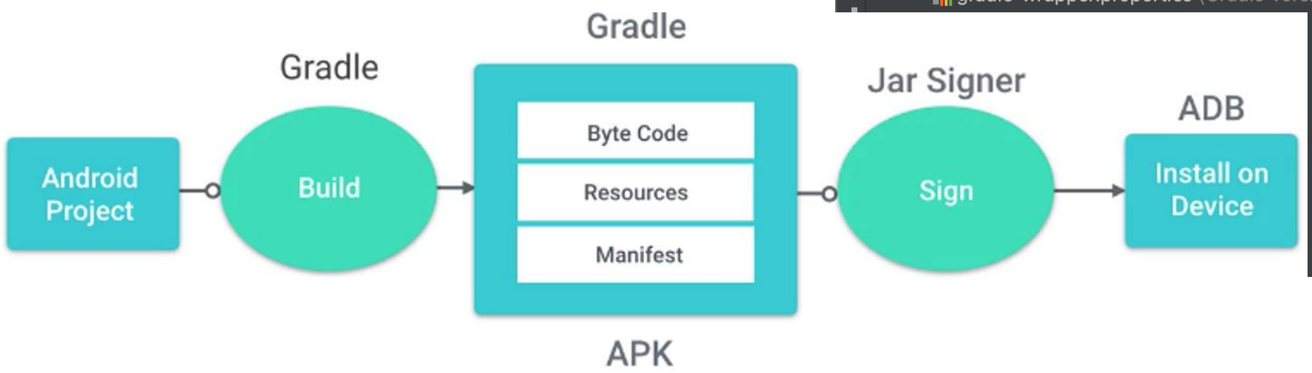
- Intent只会定义要广播的通知
- 例如，指示设备电池电量不足的广播只包括指示“电池电量不足”的已知操作字符串

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





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-sdk android:minSdkVersion="7" android:targetSdkVersion="19" />

    <uses-feature android:name="android.hardware.camera.any"
                  android:required="true" />
        ...
</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-permission android:name="android.permission.INTERNET" />

    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="19" />

    <uses-feature android:name="android.hardware.camera.any"
                  android:required="true" />
        ...
</manifest>
```

第一个Android App 小结

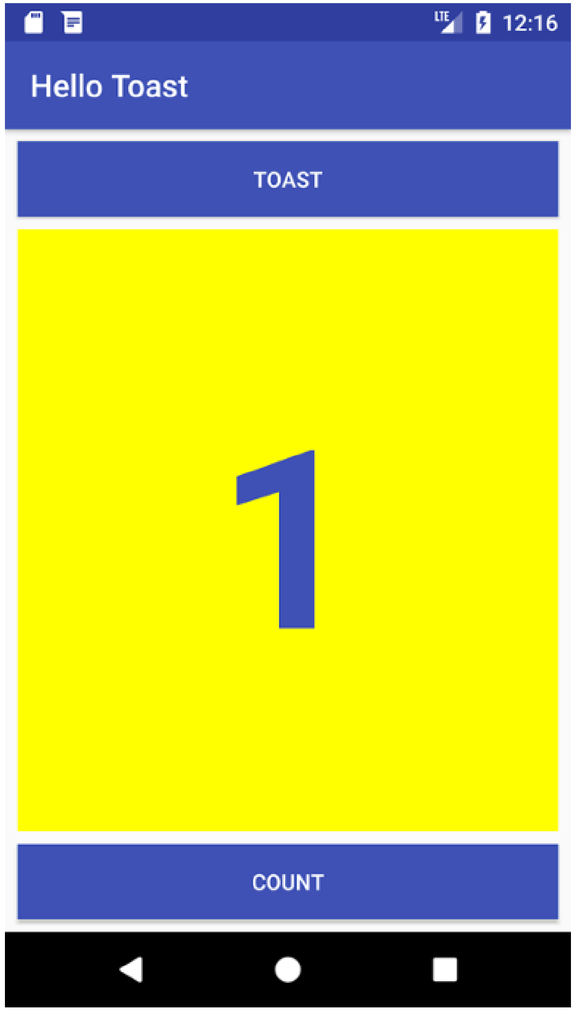
- Android Studio功能 - 创建、运行、调试
- 应用组件：活动 (Activity)，服务，内容提供程序，广播接收器
  - Activity - UI，服务 - 后台应用，内容提供程序 - 数据管理，广播接收器 - 消息
  - Activity、服务和广播接收器通过Intent的异步消息进行启动
  - Activity = Layout (xml，布局，静态) + Activity子类 (Java，功能，动态)
  - 活动App = 一个或多个Activity + 资源文件 + 清单文件
  - Intent：同一App的不同Activity，不同App的Activity、服务和广播接收器之间
- 清单文件 (manifest)
  - 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

浙江大学 ZHEJIANG UNIVERSITY

什么是视图?

- Activity = Layout (xml) + Activity子类 (Java)
- Layout = 一个层次表示的视图 (多个Views)
- 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

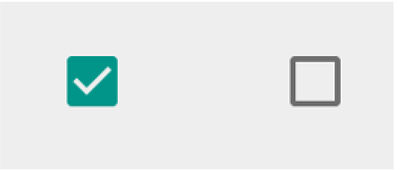
- 视图(View)类是构建用户界面的基本元素，是UI组件 (如按钮，复选框和文本输入字段)的基类

- 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, FrameLayout, TableLayout)

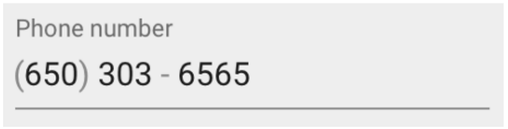
一些常用的视图

Button

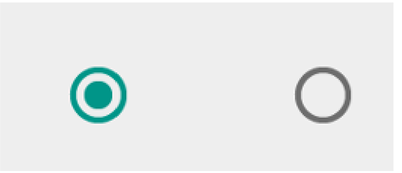
CheckBox



EditText

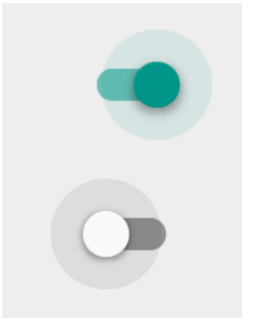


RadioButton



Slider

Switch



视图属性

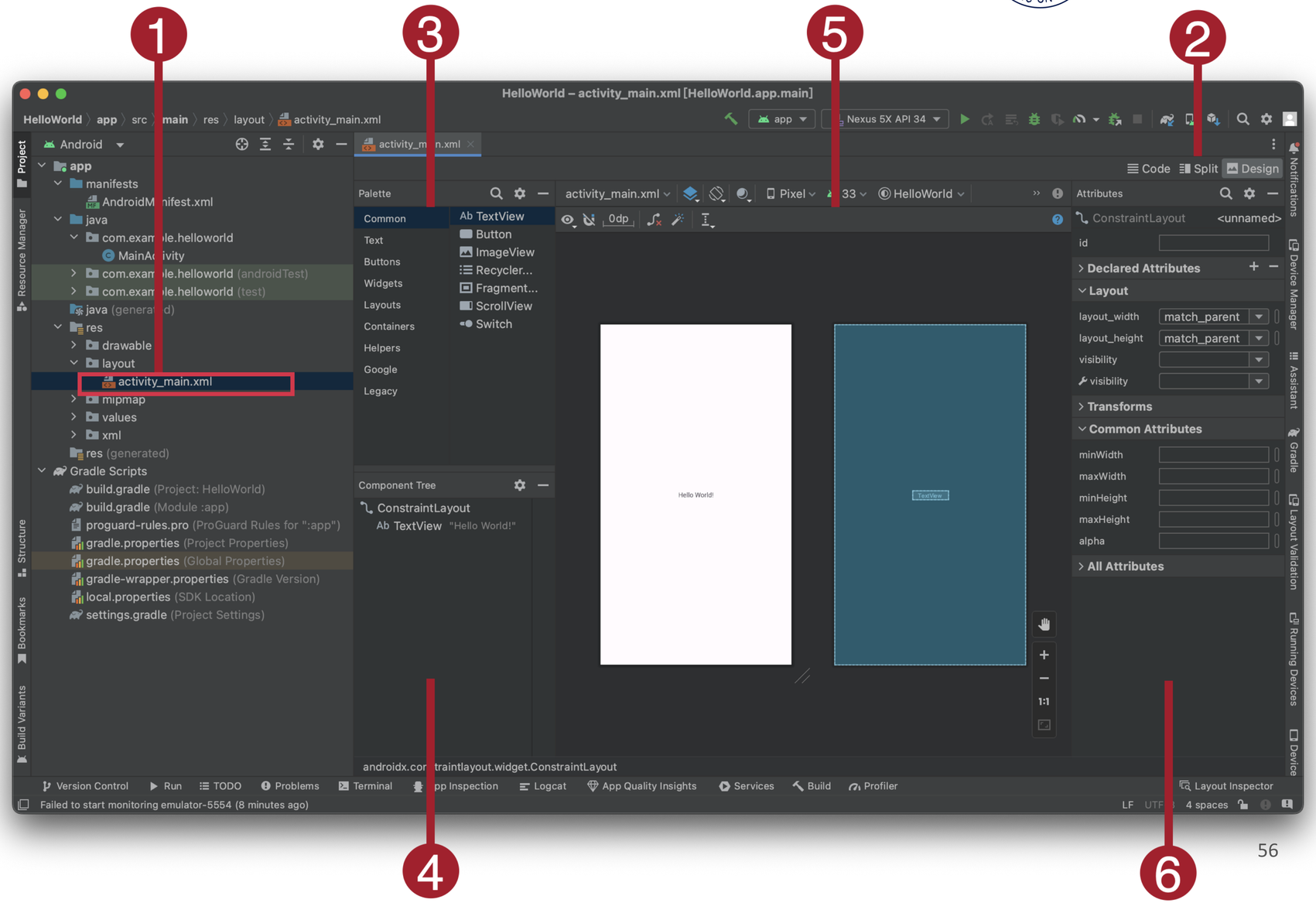
- 颜色，尺寸，位置
- 一些视图有焦点 (点击后接收用户输入)
- 互动式 (响应用户点击)
- 是否可见
- 与其他视图之间的联系

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

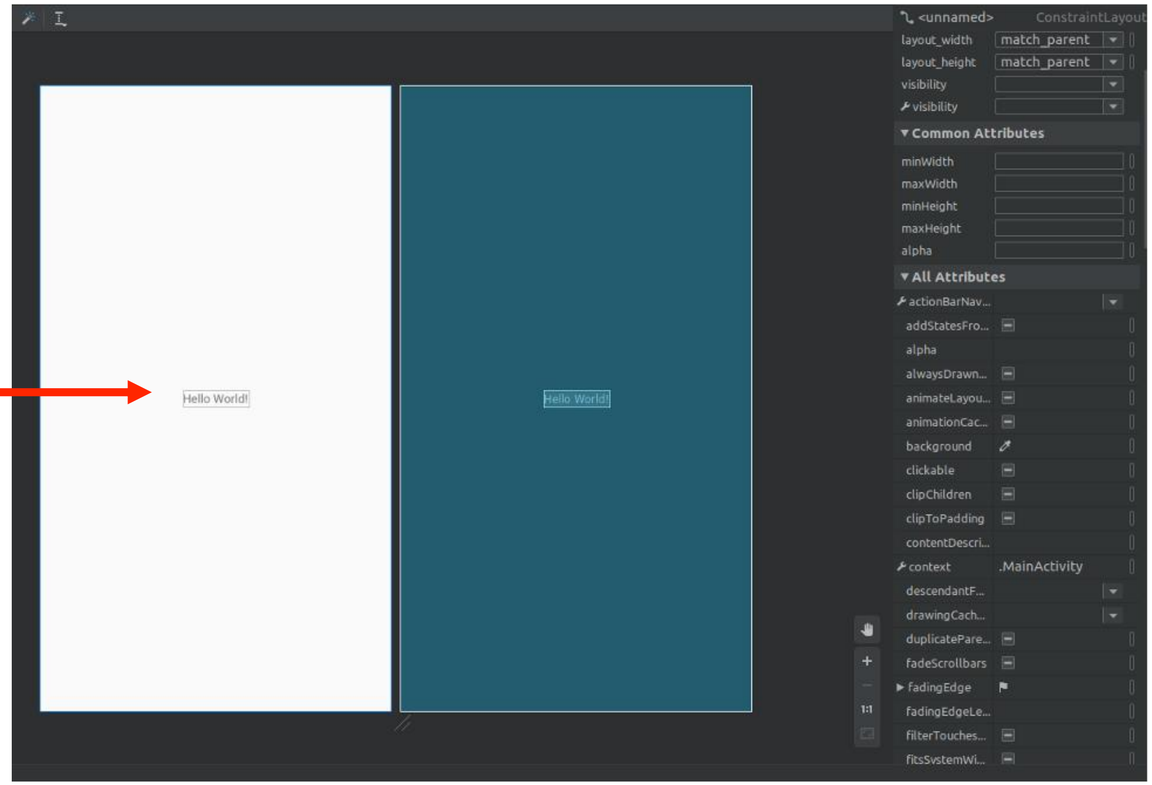
Android Studio Layout Editor（方法一）

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



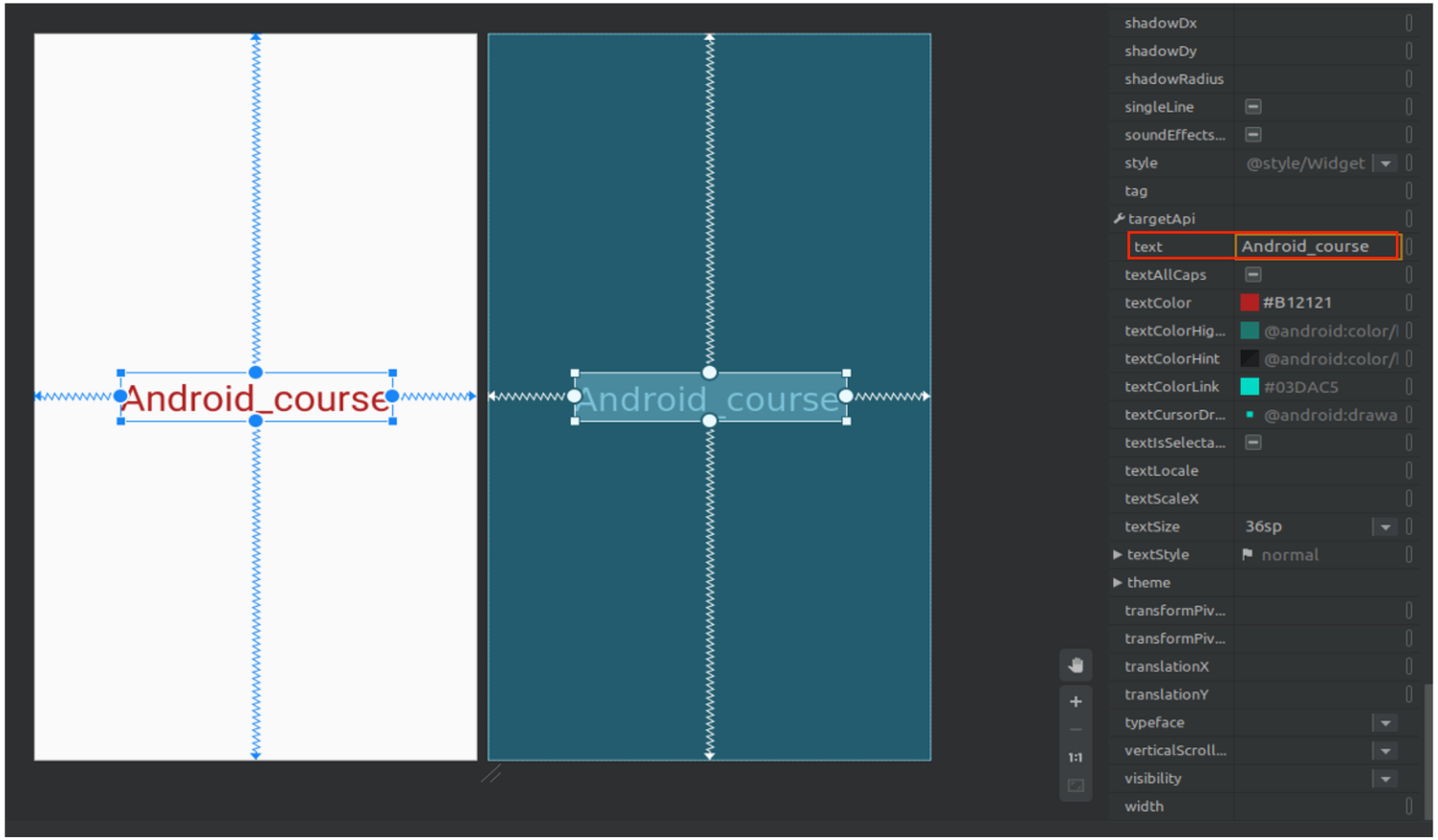
Android Studio Layout Editor（方法一）

```
<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"/>
```



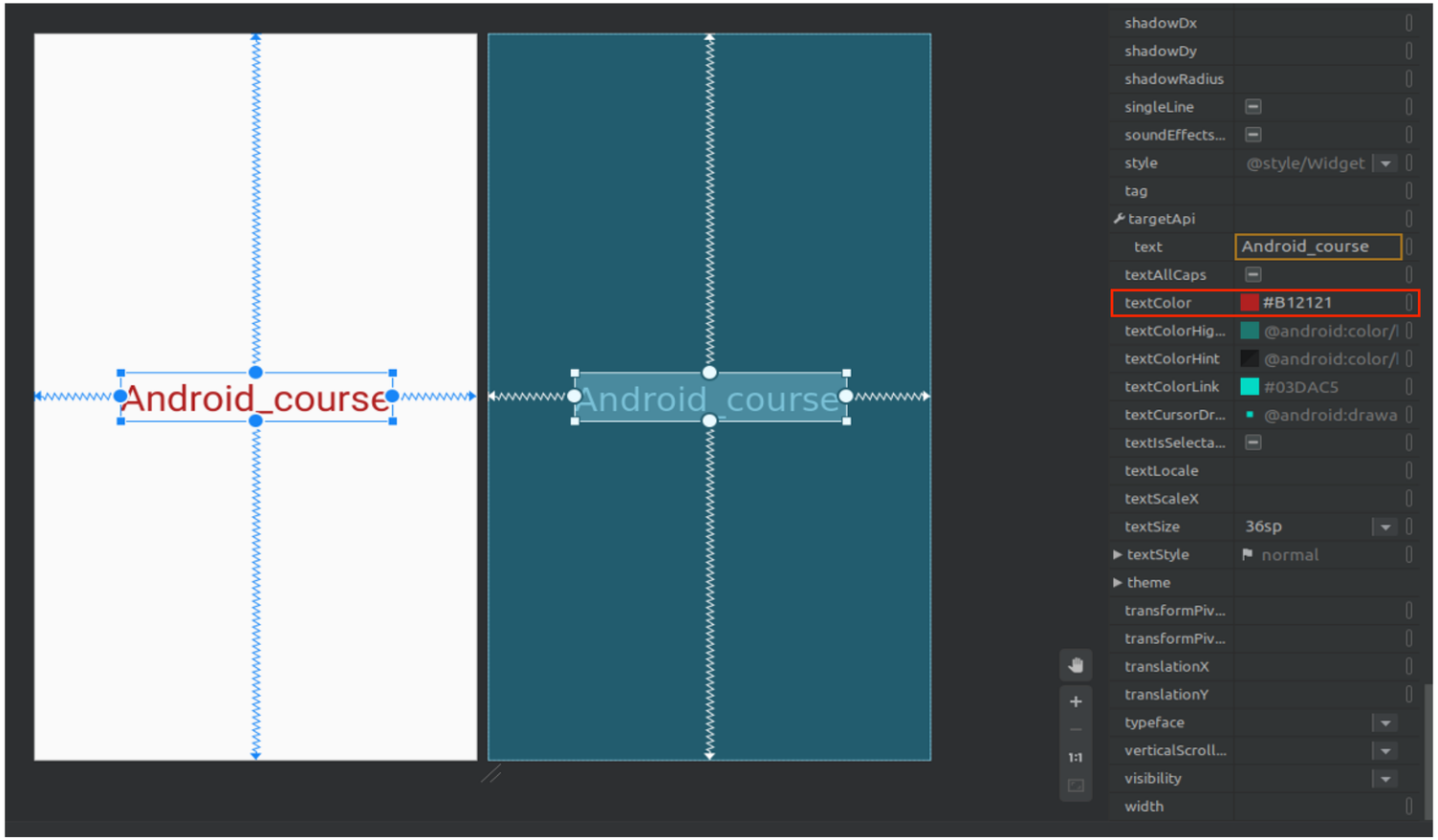
Android Studio Layout Editor（方法一）

```
<TextView
    ...
    android:text="Android_course"
    ...
/>
```



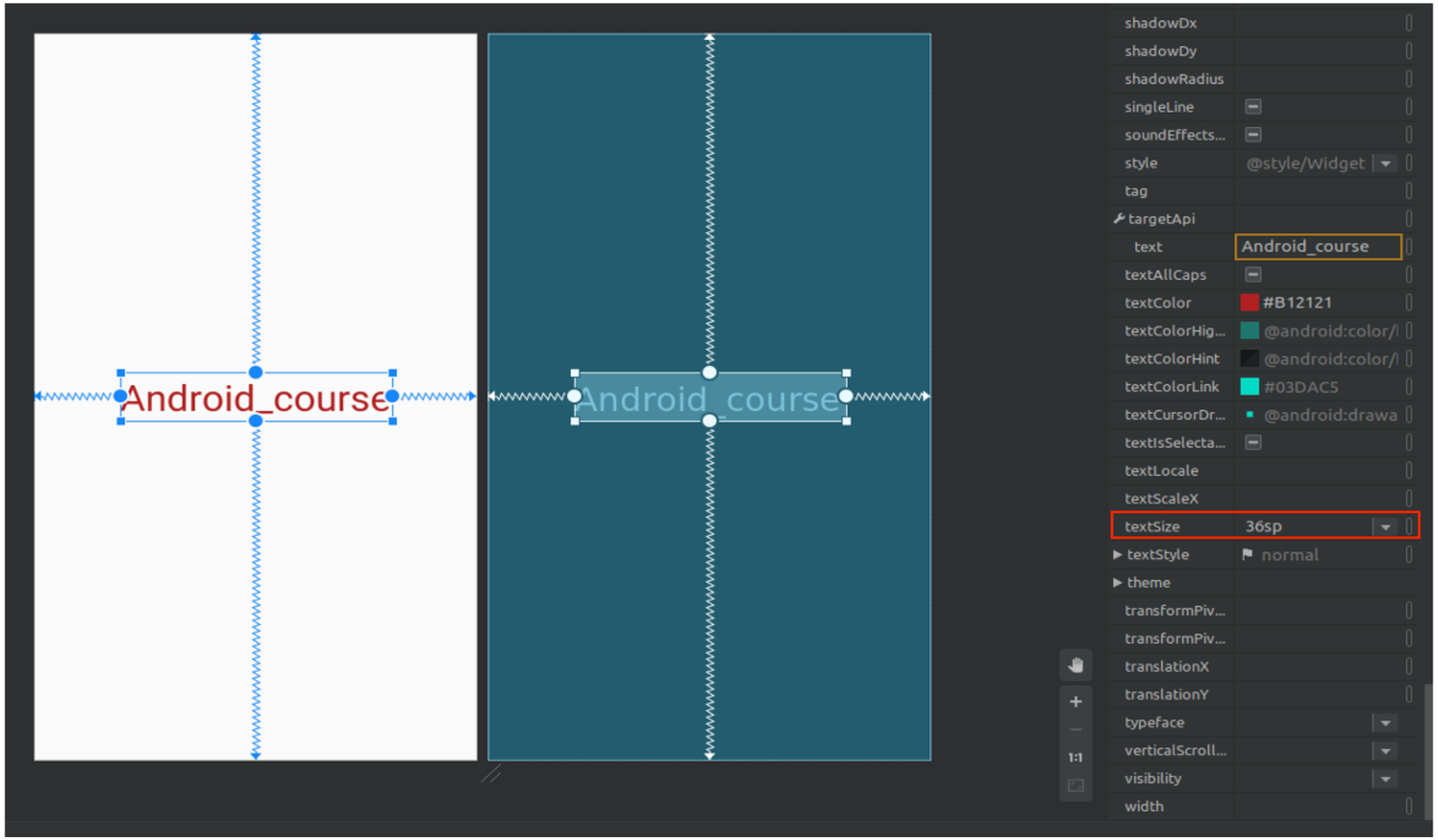
Android Studio Layout Editor（方法一）

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



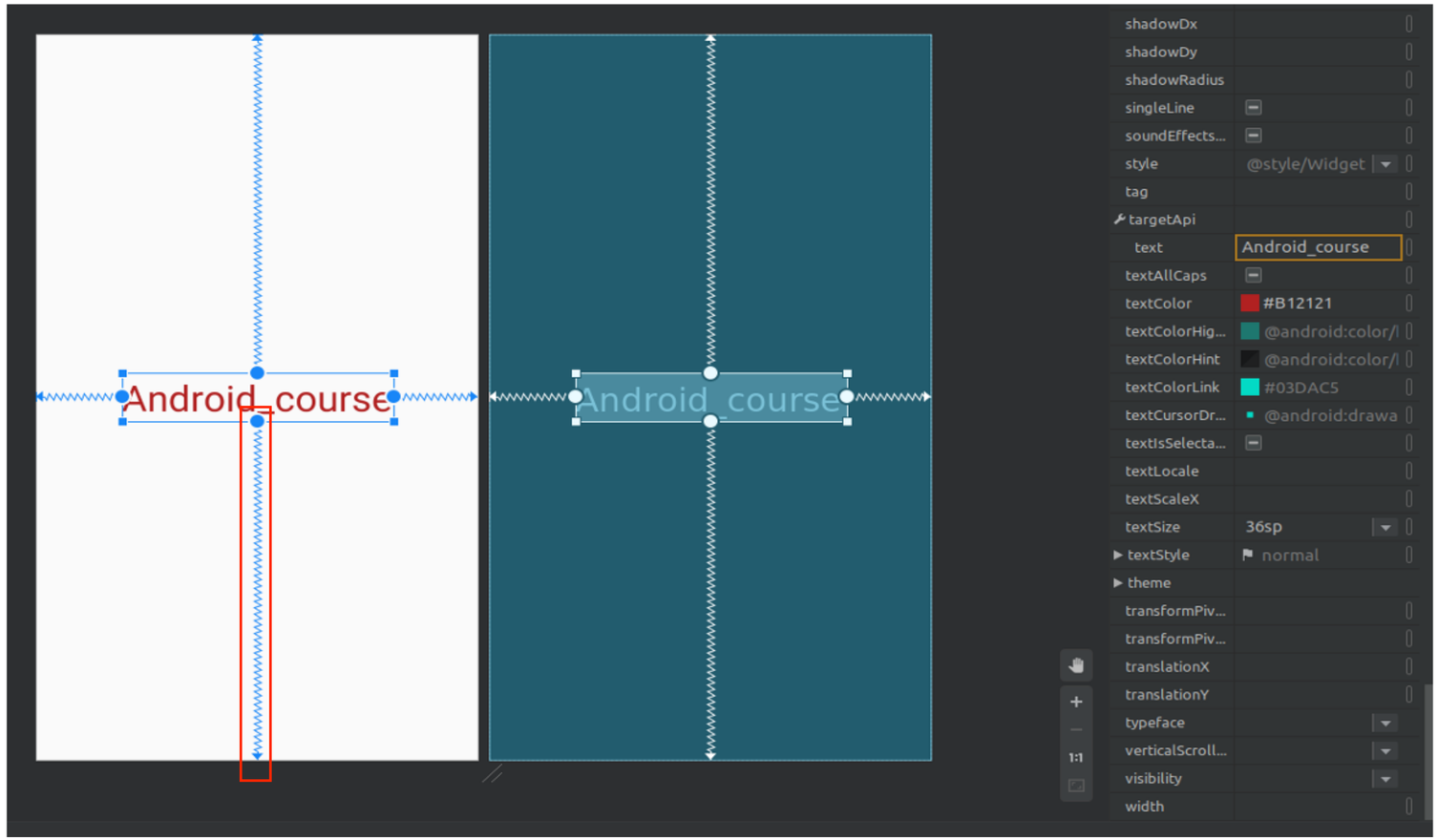
Android Studio Layout Editor（方法一）

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



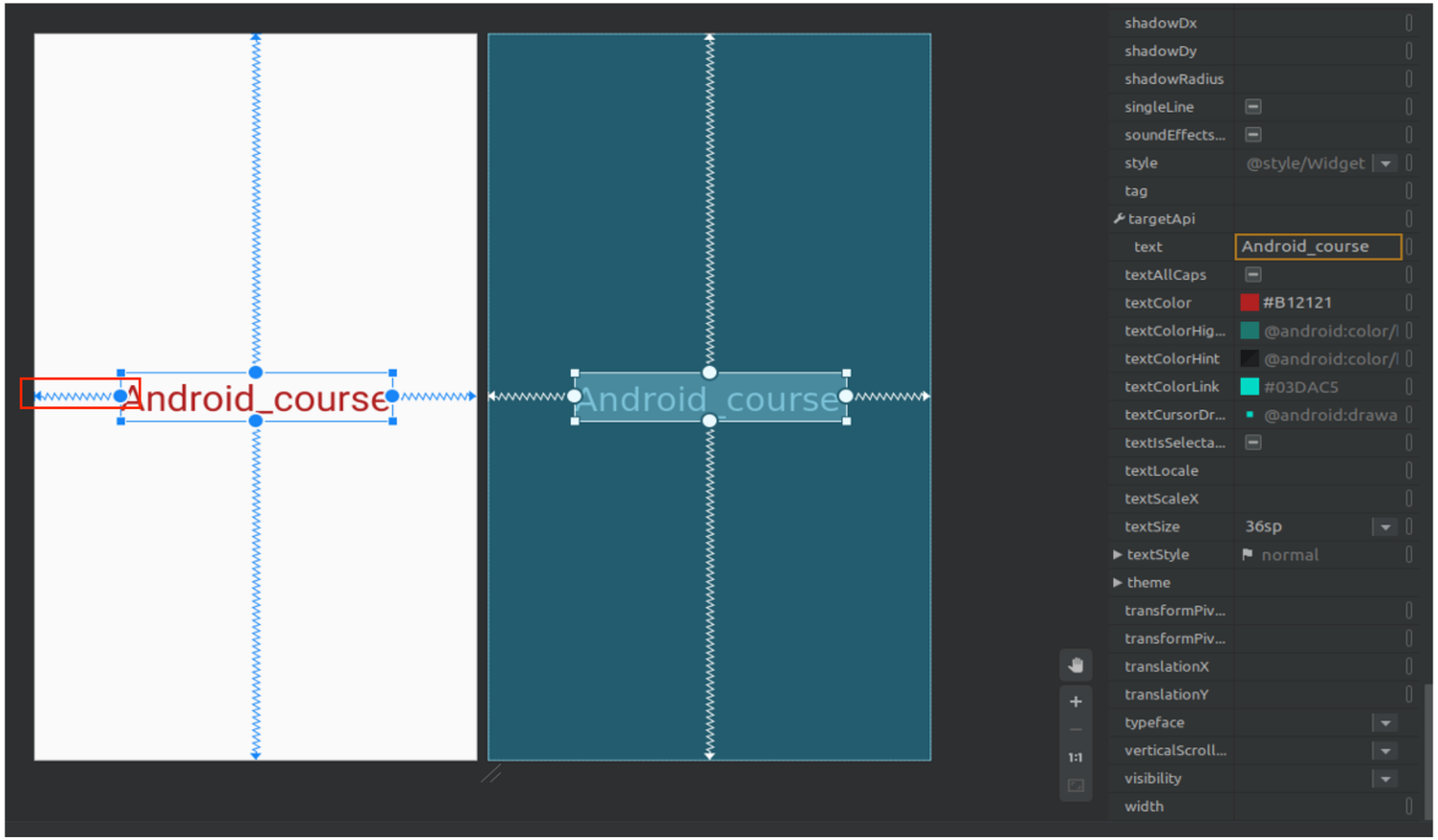
Android Studio Layout Editor（方法一）

```
<TextView
      ...

app:layout_constraintBottom_
toBottomOf="parent"
      ...
/>
```



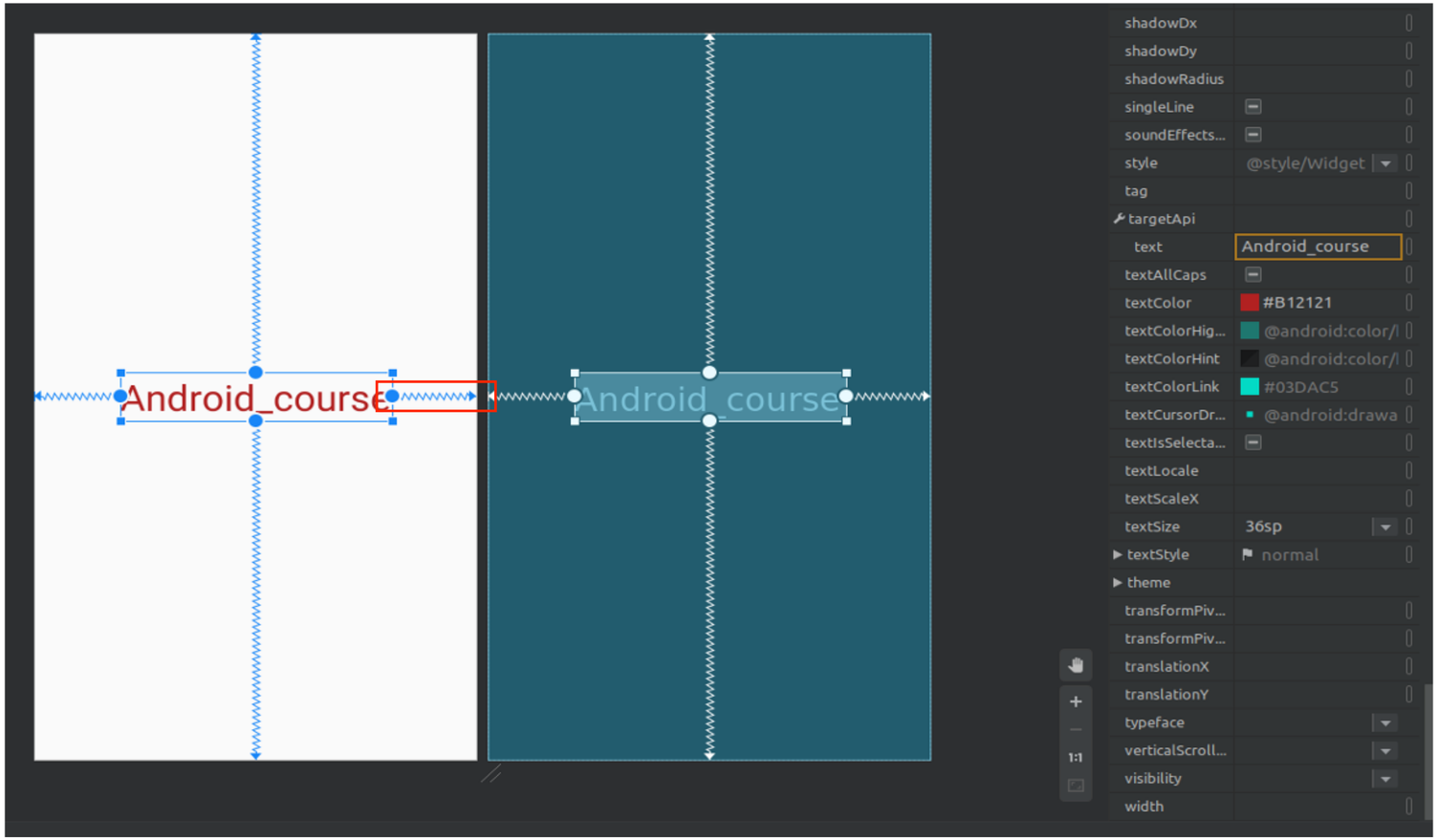
Android Studio Layout Editor（方法一）

```
<TextView
      ...

app:layout_constraintLeft_to
LeftOf="parent"
      ...
/>
```



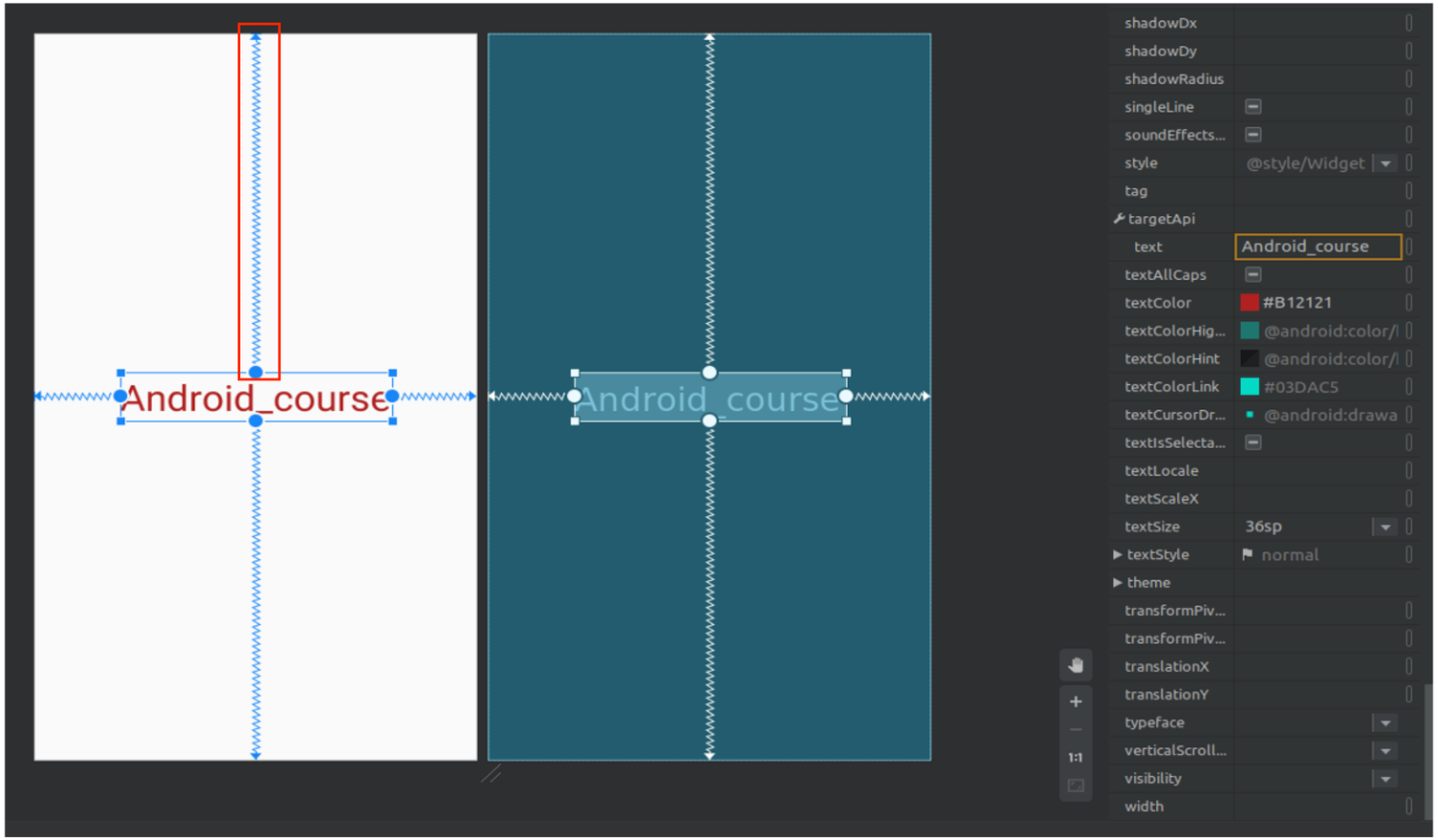
Android Studio Layout Editor（方法一）

```
<TextView
      ...

app:layout_constraintRight_t
oRightOf="parent"
      ...
/>
```



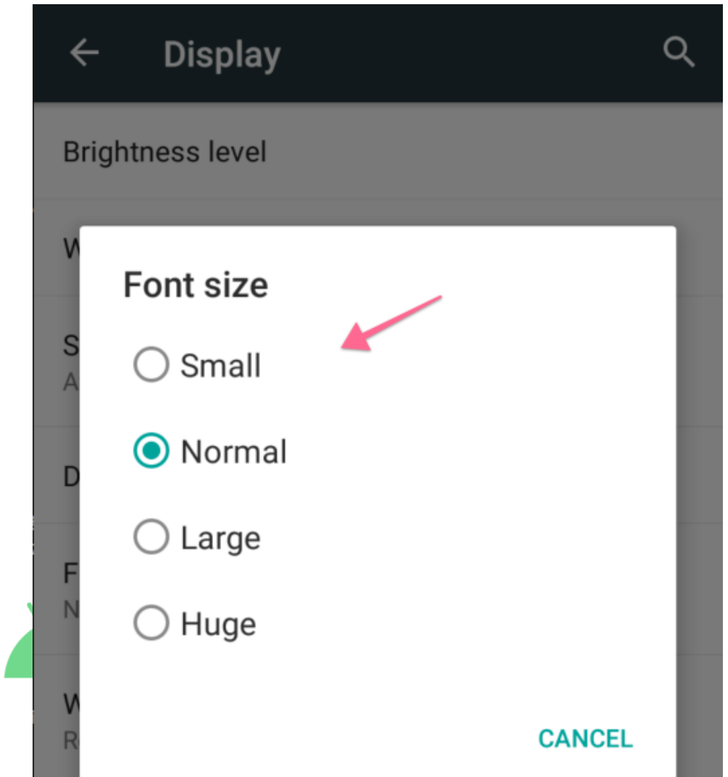
Android Studio Layout Editor（方法一）

```
<TextView
      ...

app:layout_constraintTop_toT
opOf="parent"
      ...
/>
```



通过XML定义视图（方法二）

- 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（方法三）

- 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!");
```

Android Studio Layout Editor
→ XML文件
→ Activity子类OnCreate时读取XML文件，创建视图（关联布局与功能）
setContentView(R.layout.activity_main);

视图组和视图的层次性

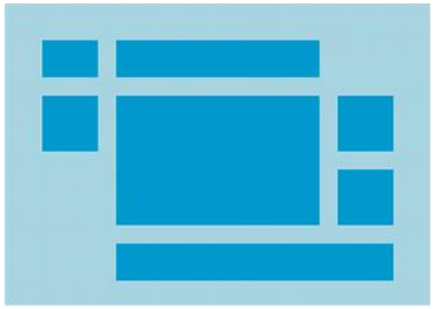
- 视图组包含“子”视图，是视图的层次化表达，常见的视图组包括：
  - LinearLayout：水平或竖直的线性布局
  - ConstraintLayout：使用约束来限制UI元素的位置。约束包括与其他元素之间或是与布局边缘之间的约束
  - FrameLayout：堆叠布局，类似画布
  - ScrollView：包含一个元素并能滚动
  - RecyclerView：包含多个元素，并通过动态添加和删除元素来滚动

布局与视图组

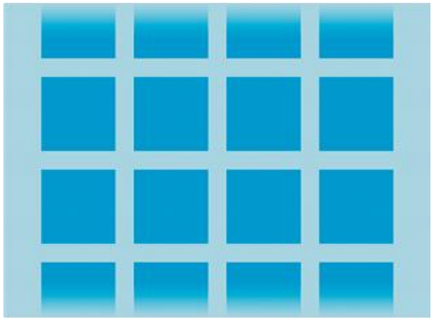
- 布局是视图组的一种特定类型 (布局类是视图组的子类)，因此也包含子视图。常见的布局包括：LinearLayout, ConstraintLayout, FrameLayout, TableLayout, GridLayout, ….

LinearLayout

ConstraintLayout



TableLayout

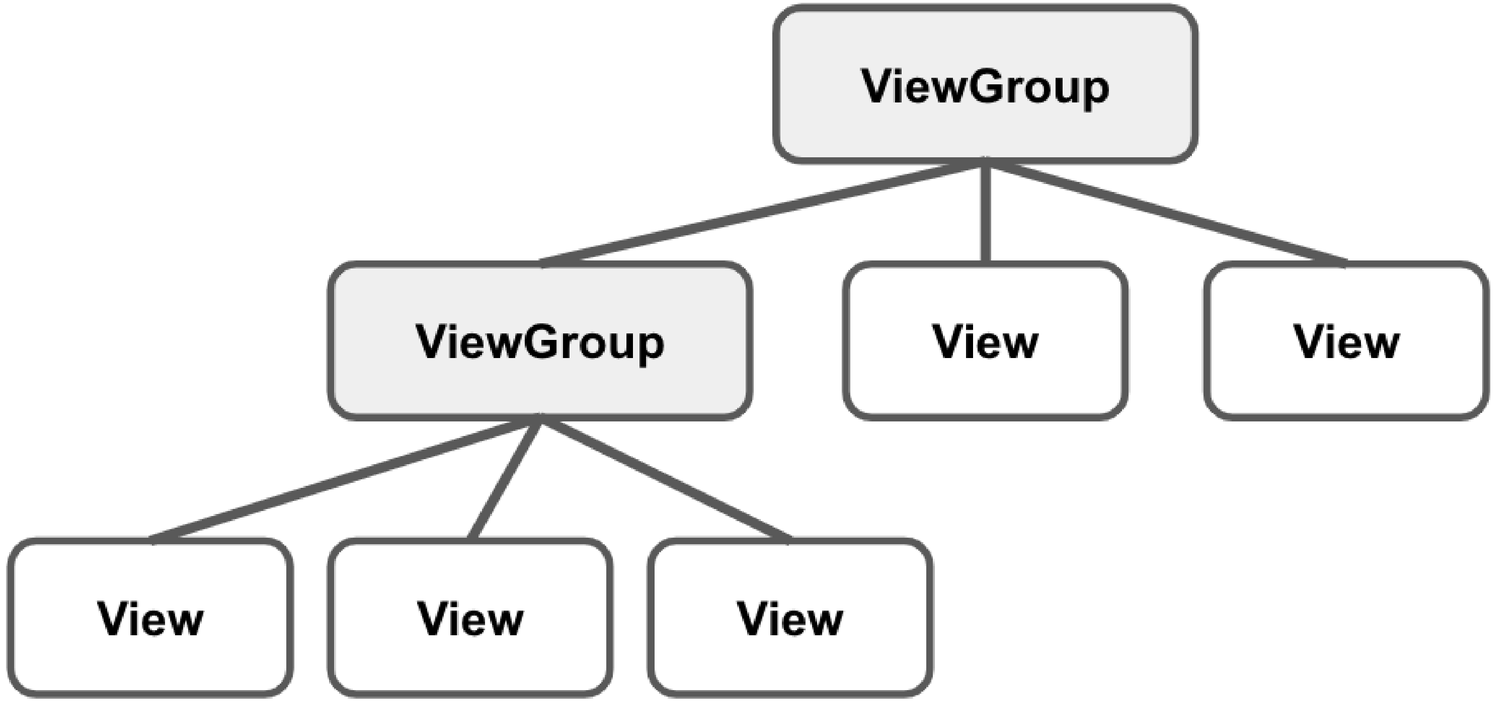


GridLayout

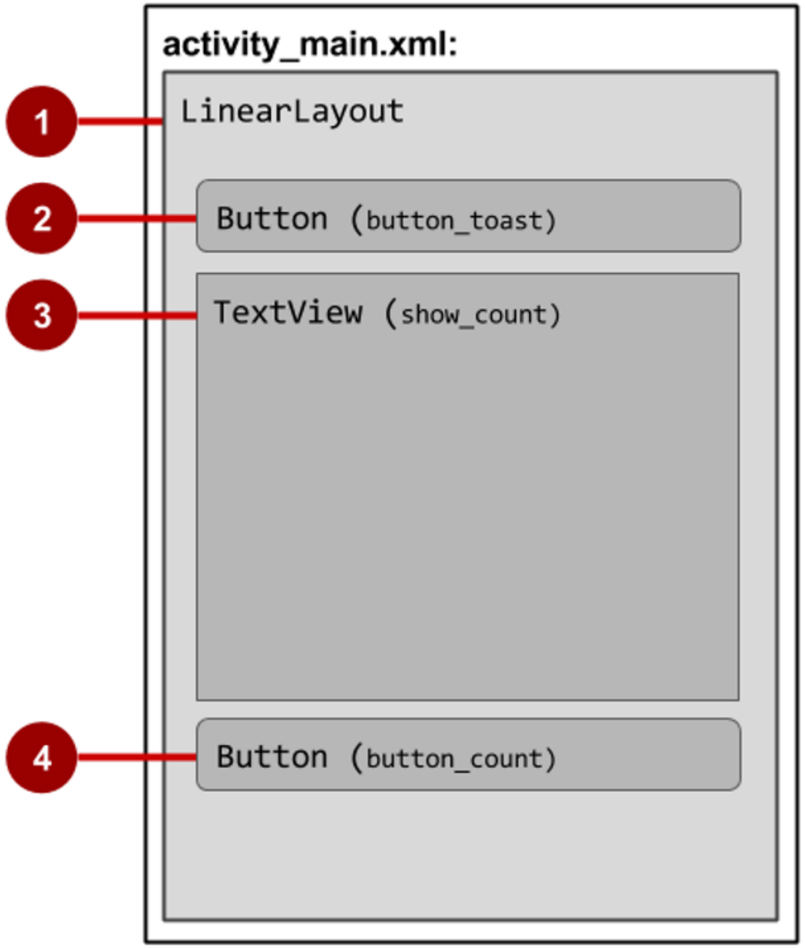
视图的层次性 vs 布局的层次性

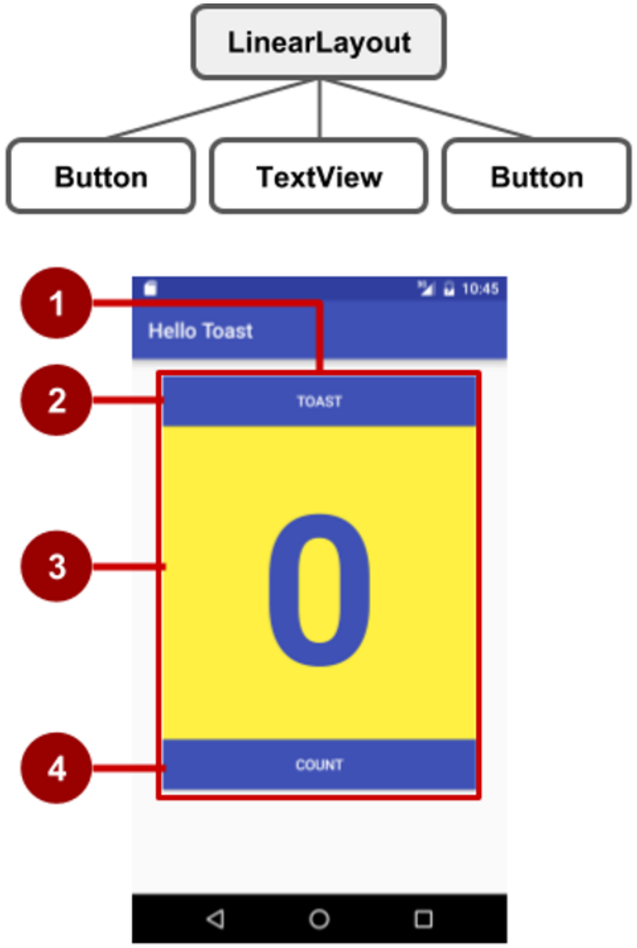
- 视图类的层次结构是标准的面向对象中的类继承
  - 例如，Button is-a TextView is-a View is-an Object
  - 上述是is-a的关系
- 布局的层次结构则是描述视图如何排列
  - 例如，LinearLayout可以包含排列在一行的多个Button
  - 上述是has-a的关系

视图组和视图的层次关系

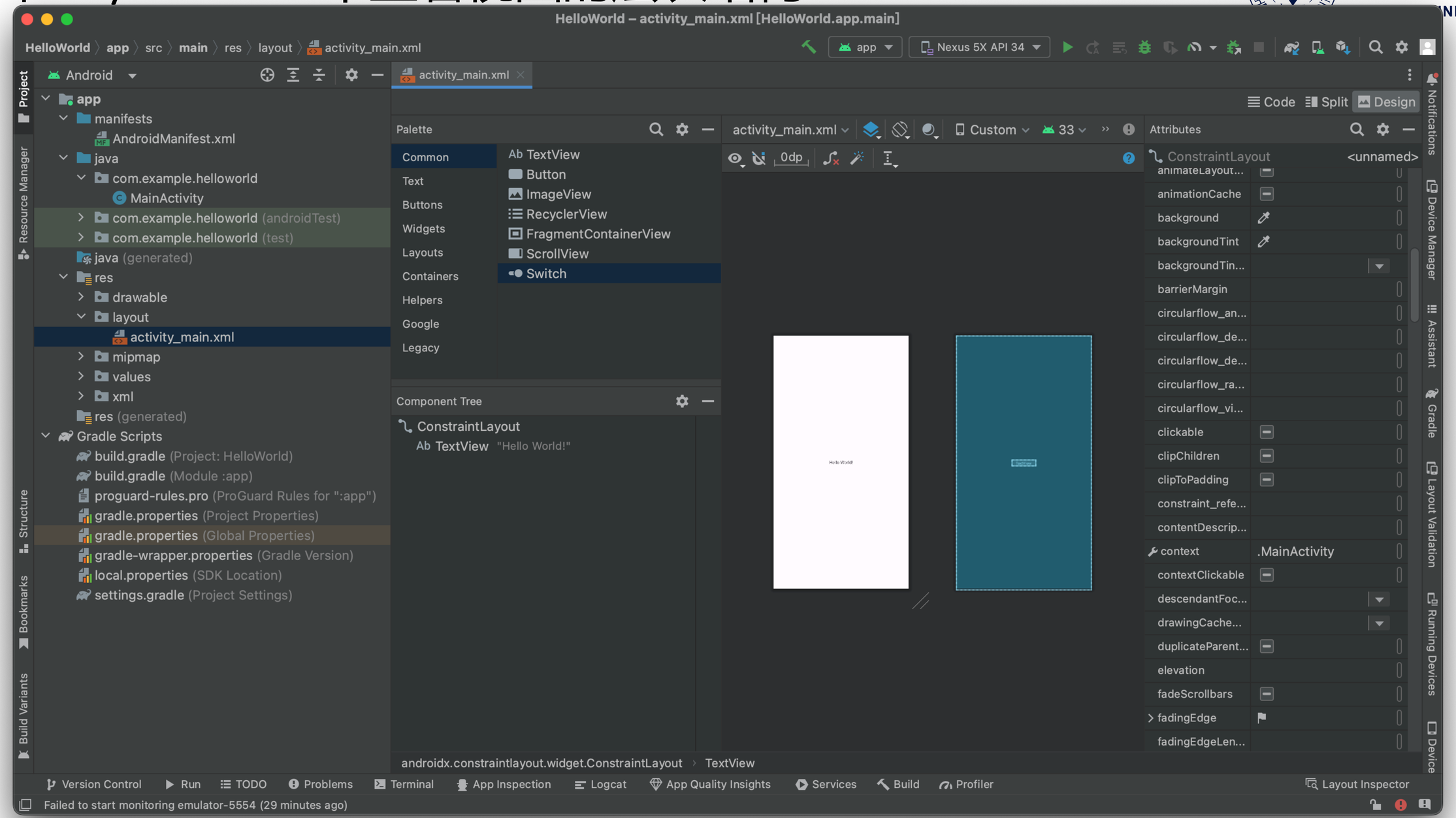


视图的层次性和屏幕布局





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

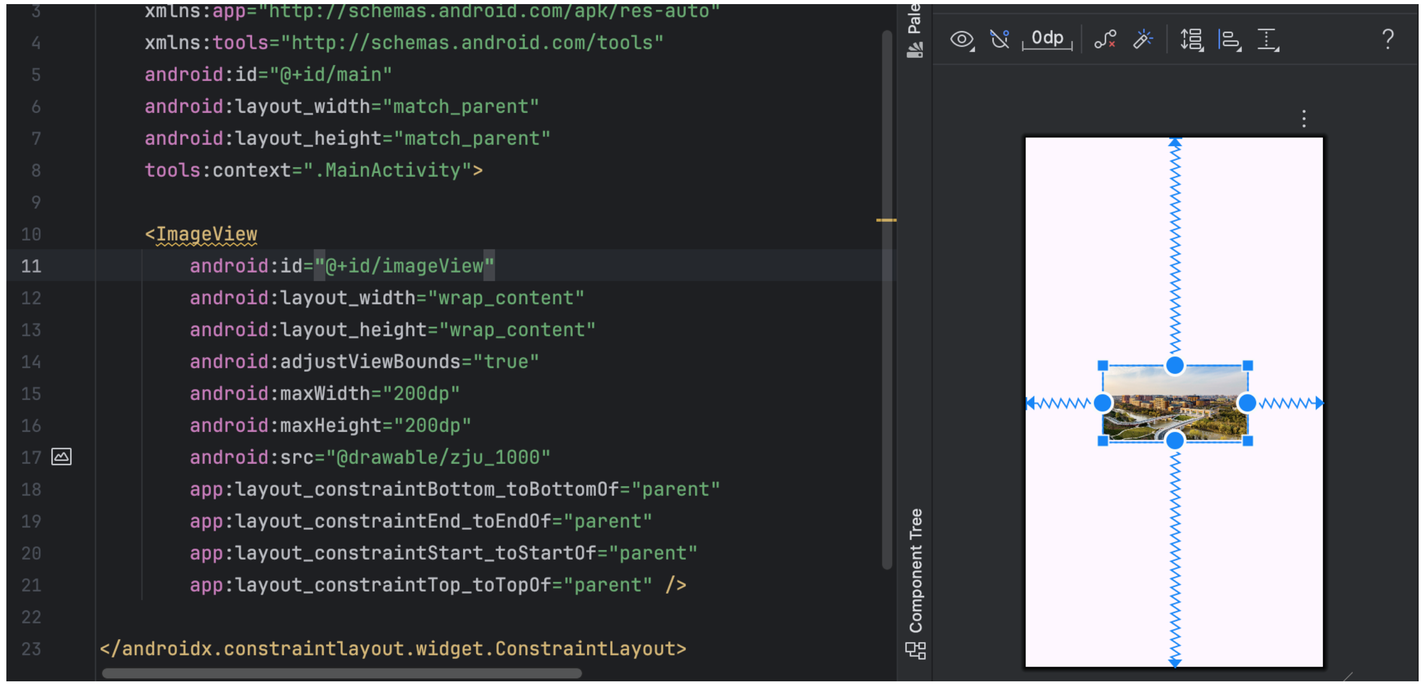
```
LinearLayout linearL = new LinearLayout(this);
linearL.setOrientation(LinearLayout.VERTICAL);
TextView myText = new TextView(this);
myText.setText("Display this text!");
linearL.addView(myText);
setContentView(linearL);
...
//Set the width and height of a view:
LinearLayout.LayoutParams layoutParams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

资源

- 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

- 资源在Java/Kotlin文件是通过R.资源类型.资源名称进行访问
- setContentView(R.layout.activity_main)关联布局activity_main.xml和功能MainActivity类，读取xml布局文件，创建所有视图实例
- findViewById(R.id.imageView)通过id查找对应的图像视图实例对象
- setImageResource(R.drawable.zju_1000)运行时动态修改图片内容

```
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

视图、布局和资源小结

- 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

同步更新

运行时Activity创建时

- 视图组与视图的层次性
  - LinearLayout、ConstraintLayout、FrameLayout
  - ScrollView、RecyclerView
  - Intent：各类视图的常用属性：textSize、padding、margin、gravity、layout_gravity等

- 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

- 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

浙江大学
ZHEJIANG UNIVERSITY

Thanks!