

**TRƯỜNG ĐẠI HỌC THỦY LỢI  
KHOA CÔNG NGHỆ THÔNG TIN**



# **GIÁO TRÌNH**

## **THỰC HÀNH PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG**

Hà Nội, 2.2025

# MỤC LỤC

CHƯƠNG 1. Làm quen .....	3
Bài 1) Tạo ứng dụng đầu tiên .....	3
1.1) Android Studio và Hello World.....	3
1.2) Giao diện người dùng tương tác đầu tiên .....	23
1.3) Trình chỉnh sửa bố cục .....	37
1.4) Văn bản và các chế độ cuộn .....	37
1.5) Tài nguyên có sẵn .....	37
Bài 2) Activities.....	37
2.1) Activity và Intent.....	37
2.2) Vòng đời của Activity và trạng thái .....	37
2.3) Intent ngầm định .....	37
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ .....	37
3.1) Trình gỡ lỗi.....	37
3.2) Kiểm thử đơn vị .....	37
3.3) Thư viện hỗ trợ .....	37
CHƯƠNG 2. Trải nghiệm người dùng.....	38
Bài 1) Tương tác người dùng .....	38
1.1) Hình ảnh có thể chọn.....	38
1.2) Các điều khiển nhập liệu .....	38
1.3) Menu và bộ chọn .....	38
1.4) Điều hướng người dùng .....	38
1.5) RecyclerView .....	38
Bài 2) Trải nghiệm người dùng thú vị.....	38
2.1) Hình vẽ, định kiểu và chủ đề .....	38
2.2) Thẻ và màu sắc .....	38
2.3) Bố cục thích ứng.....	38
Bài 3) Kiểm thử giao diện người dùng .....	38

3.1) Espresso cho việc kiểm tra UI .....	38
CHƯƠNG 3. Làm việc trong nền .....	38
Bài 1) Các tác vụ nền.....	38
1.1) AsyncTask.....	38
1.2) AsyncTask và AsyncTaskLoader .....	38
1.3) Broadcast receivers .....	38
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền.....	38
2.1) Thông báo.....	38
2.2) Trình quản lý cảnh báo .....	38
2.3) JobScheduler .....	38
CHƯƠNG 4. Lưu dữ liệu người dùng .....	39
Bài 1) Tùy chọn và cài đặt.....	39
1.1) Shared preferences .....	39
1.2) Cài đặt ứng dụng .....	39
Bài 2) Lưu trữ dữ liệu với Room .....	39
2.1) Room, LiveData và ViewModel.....	39
2.2) Room, LiveData và ViewModel.....	39
3.1) Trình gowx lỗi .....	

## CHƯƠNG 1. LÀM QUEN

### Bài 1) Tạo ứng dụng đầu tiên

#### 1.1) Android Studio và Hello World

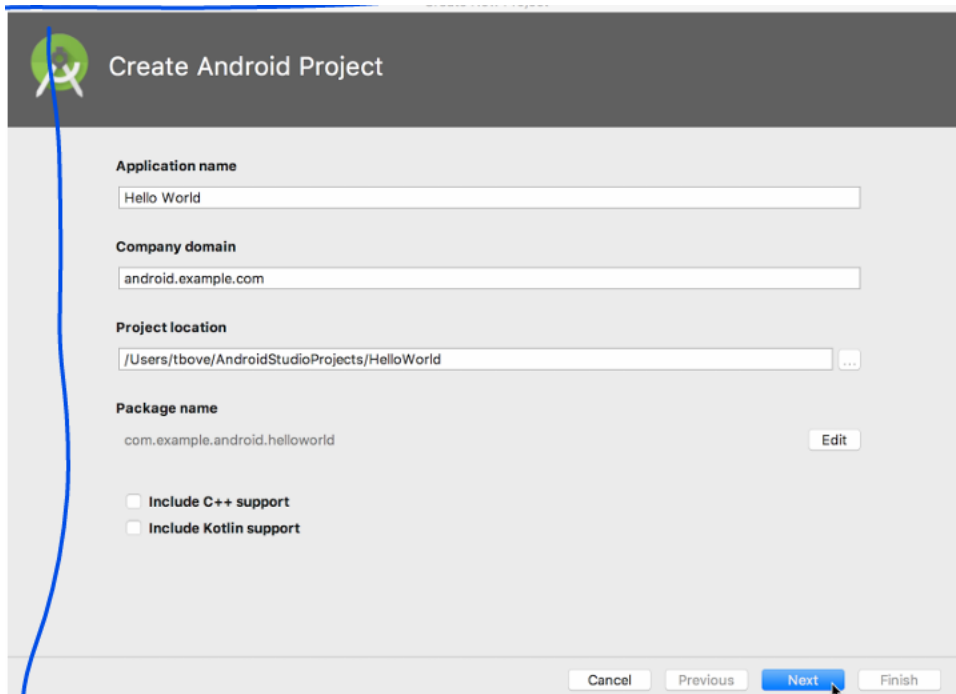
### Giới thiệu

Trong bài thực hành này, bạn sẽ tìm hiểu cách cài đặt Android Studio, môi trường phát triển Android. Bạn cũng sẽ tạo và chạy ứng dụng Android đầu tiên của mình, Hello World, trên một trình giả lập và trên một thiết bị vật lý.

### Những gì Bạn nên biết

Bạn nên có khả năng:

- Hiểu quy trình phát triển phần mềm tổng quát cho các ứng dụng lập trình hướng đối tượng sử dụng một IDE (môi trường phát triển tích hợp) như Android Studio.
- Chứng minh rằng bạn có ít nhất 1-3 năm kinh nghiệm trong lập trình hướng đối tượng, với một phần trong số đó tập trung vào ngôn ngữ lập trình Java. (Các bài thực hành này sẽ không giải thích về lập trình hướng đối tượng hoặc ngôn ngữ Java.



### Những gì Bạn sẽ cần:

- Một máy tính chạy Windows hoặc Linux, hoặc một Mac chạy macOS. Xem trang tải xuống Android Studio để biết yêu cầu hệ thống cập nhật.
- Truy cập Internet hoặc một phương pháp thay thế để tải các cài đặt mới nhất của Android Studio và Java lên máy tính của bạn.

### Những gì bạn sẽ học

- Cách cài đặt và sử dụng IDE Android Studio.
- Cách sử dụng quy trình phát triển để xây dựng ứng dụng Android.
- Cách tạo một dự án Android từ một mẫu.
- Cách thêm thông điệp ghi lại vào ứng dụng của bạn để phục vụ mục đích gỡ lỗi.

## Những gì bạn sẽ làm

- Cài đặt môi trường phát triển **Android Studio**.
- Tạo một trình giả lập (thiết bị ảo) để chạy ứng dụng của bạn trên máy tính.
- Tạo và chạy ứng dụng **Hello World** trên các thiết bị ảo và vật lý.
- Khám phá cấu trúc dự án.
- Tạo và xem các thông điệp ghi lại từ ứng dụng của bạn.
- Khám phá tệp **AndroidManifest.xml**

## Tổng quan ứng dụng

Sau khi bạn cài đặt thành công Android Studio, bạn sẽ tạo một dự án mới, từ mẫu, cho ứng dụng Hello World. Ứng dụng đơn giản này sẽ hiển thị chuỗi chữ “Hello World” trên màn hình của thiết bị Android ảo hoặc thiết bị vật lý.

Đây là giao diện của ứng dụng sau khi hoàn thành:

9:08



Hello World!



1:1



## Nhiệm vụ 1: Cài đặt Android Studio

Android Studio cung cấp một môi trường phát triển tích hợp (IDE) hoàn chỉnh, bao gồm một code editor hiện đại và một bộ mẫu ứng dụng. Ngoài ra, nó còn chứa các công cụ hỗ trợ phát triển, gỡ lỗi, kiểm thử và tối ưu hiệu suất, giúp việc phát triển ứng dụng trở nên nhanh và dễ hơn. Bạn có thể thử ứng dụng của mình trên nhiều trình giả lập được cấu hình sẵn hoặc trên thiết bị di động của mình, xây dựng ứng dụng hoàn chỉnh và đăng lên cửa hàng Google Play.

Android Studio có sẵn cho các máy tính chạy Windows, Linux và macOS. Phiên bản mới nhất của OpenJDK (Java Development Kit) được tích hợp sẵn trong Android Studio.

Để bắt đầu với Android Studio, trước tiên hãy kiểm tra các yêu cầu hệ thống để đảm bảo thiết bị của bạn đáp ứng đủ điều kiện. Quá trình cài đặt tương tự trên tất cả các nền tảng, chỉ có một số điểm khác biệt nhỏ được lưu ý dưới đây.

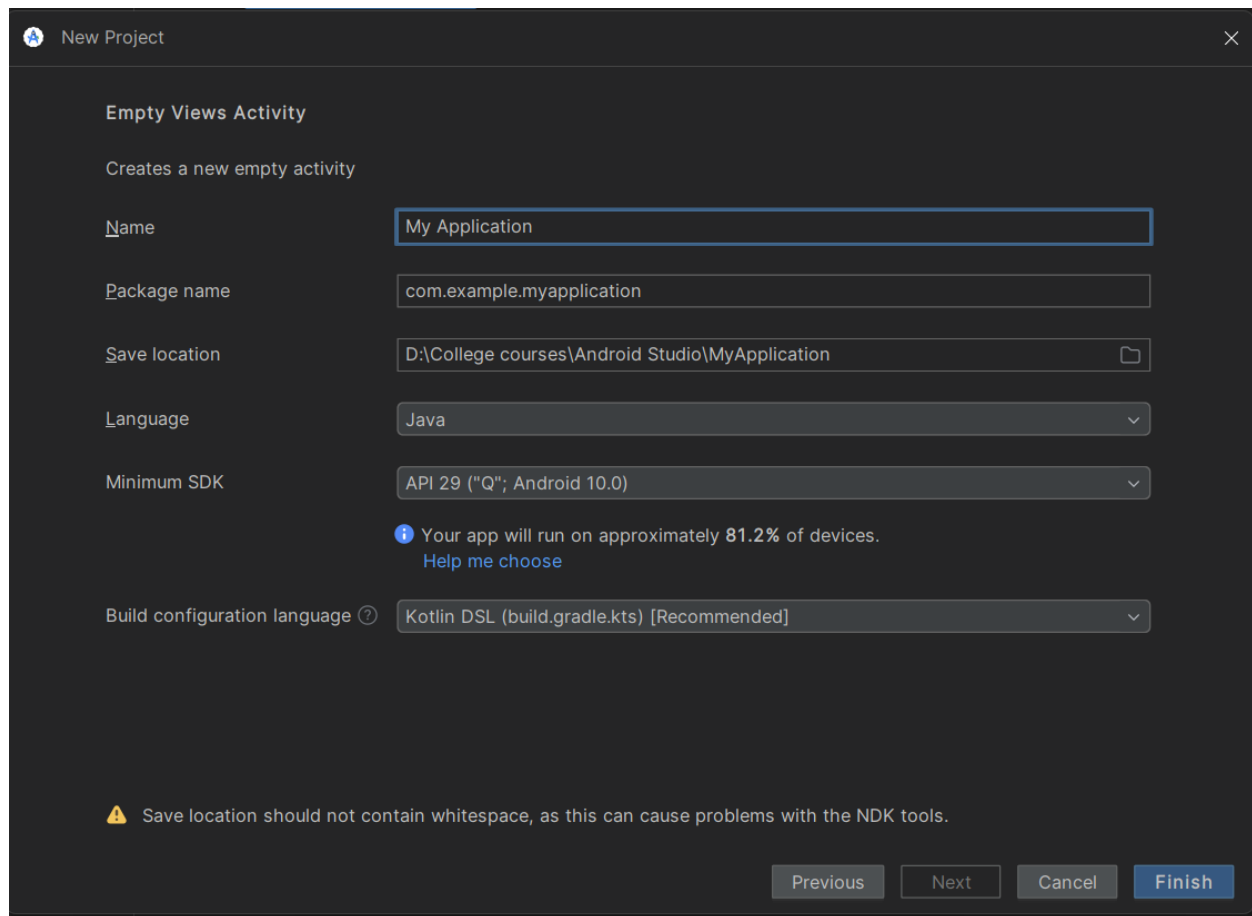
1. Truy cập trang web của Android Developers và làm theo hướng dẫn để tải xuống và cài đặt Android Studio.
2. Chấp nhận các cấu hình mặc định trong tất cả các bước và đảm bảo rằng tất cả các thành phần cần thiết đều được chọn để cài đặt.
3. Sau khi cài đặt xong, Setup Wizard sẽ tải xuống và cài đặt thêm một số thành phần khác, bao gồm Android SDK. Hãy kiên nhẫn vì quá trình này có thể mất thời gian tùy thuộc vào tốc độ Internet của bạn, và một số bước có thể lặp lại.
4. Khi quá trình tải xuống hoàn tất, Android Studio sẽ khởi động và bạn đã sẵn sàng tạo dự án đầu tiên của mình.

## Nhiệm vụ 2: Tạo ứng dụng Hello World

Trong nhiệm vụ này, bạn sẽ tạo một ứng dụng hiển thị dòng chữ "**Hello World**" để kiểm tra xem Android Studio đã được cài đặt đúng chưa, đồng thời làm quen với các bước cơ bản trong quá trình phát triển ứng dụng với Android Studio.

### 2.1 Tạo dự án ứng dụng

1. Mở **Android Studio** nếu chưa mở.
2. Ở cửa sổ chào mừng **Welcome to Android Studio**, nhấp vào **Start a new Android Studio project** (Bắt đầu một dự án Android Studio mới).
3. Trong cửa sổ **Create Android Project**, nhập **Hello World** vào ô **Application name** (Tên ứng dụng).



4. Xác nhận rằng thư mục mặc định trong **Project location** là nơi bạn muốn lưu trữ ứng dụng **Hello World** và các dự án khác trong Android Studio. Nếu cần, bạn có thể thay đổi sang thư mục mong muốn.

5. Chấp nhận giá trị mặc định **android.example.com** cho **Company Domain**, hoặc nhập một tên miền tùy chỉnh.

Nếu bạn không có kế hoạch xuất bản ứng dụng, bạn có thể giữ nguyên giá trị mặc định. Lưu ý: Việc thay đổi tên gói (package name) sau này sẽ tốn thêm công sức.

6. Bỏ chọn các tùy chọn **Include C++ support** và **Include Kotlin support**, sau đó nhấp **Next**.

7. Ở màn hình **Target Android Devices**, đảm bảo rằng **Phone and Tablet** đã được chọn. Kiểm tra rằng **API 29: Android 10.0 "Q"** được đặt làm **Minimum SDK**. Nếu không, hãy sử dụng menu thả xuống để chọn đúng phiên bản này.

8. Đảm bảo rằng tùy chọn **Generate Layout file** được chọn. Mặc định, tên tệp bố cục sẽ là **activity\_main**. Bạn có thể thay đổi nếu muốn, nhưng bài học này sẽ sử dụng **activity\_main**.

9. Đảm bảo rằng tùy chọn **Backwards Compatibility (App Compat)** được chọn. Tùy chọn này giúp ứng dụng của bạn tương thích với các phiên bản Android cũ hơn.

10. Nhấp **Finish**

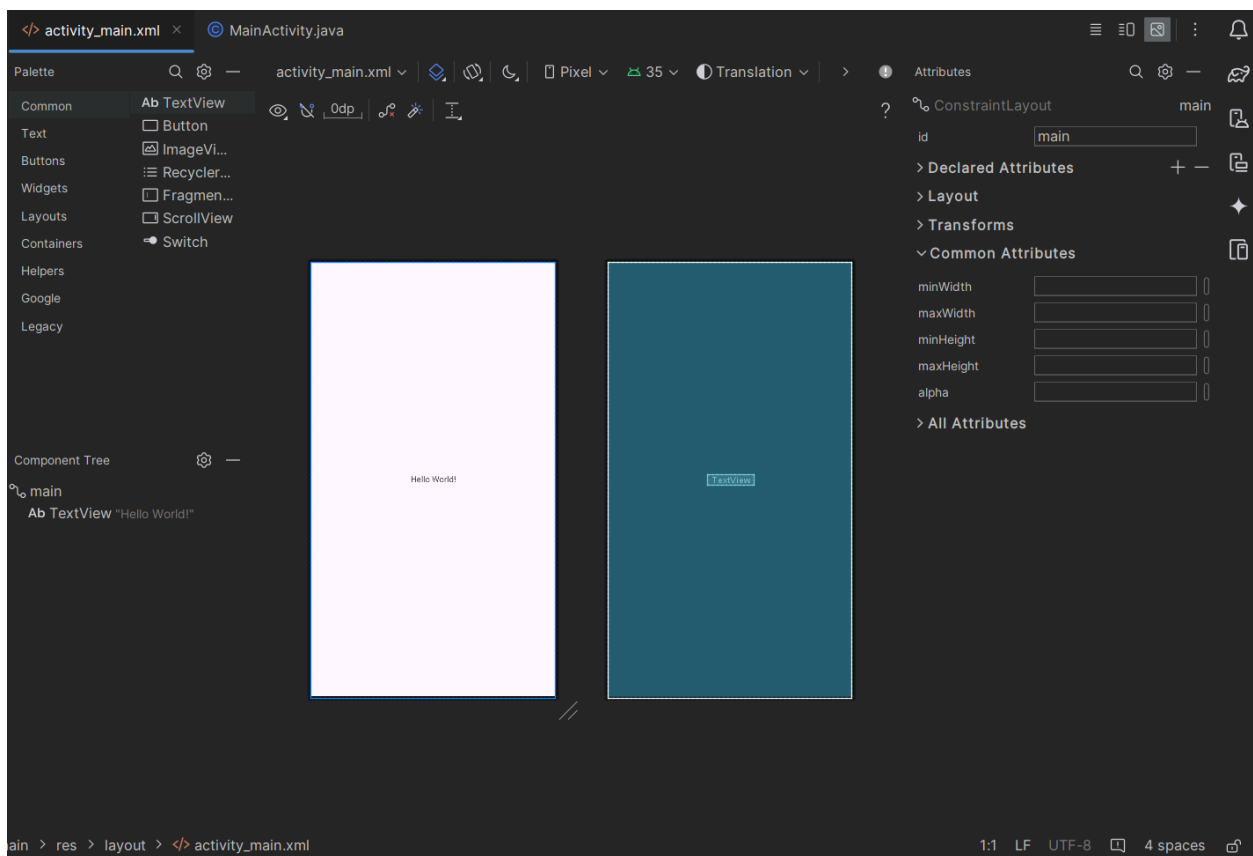


Android Studio tạo một thư mục cho các dự án của bạn và xây dựng dự án bằng Gradle (quá trình này có thể mất vài phút).

Mẹo: Xem trang tài liệu "Configure your build" dành cho nhà phát triển để biết thông tin chi tiết. Bạn cũng có thể thấy một thông báo "Tip of the day" với các phím tắt và mẹo hữu ích khác. Nhấp **Close** để đóng thông báo này.

Trình chỉnh sửa Android Studio sẽ xuất hiện. Thực hiện các bước sau:

1. Nhấp vào tab **activity\_main.xml** để mở trình chỉnh sửa bố cục.
2. Nhấp vào tab **Design** trong trình chỉnh sửa bố cục, nếu chưa được chọn, để hiển thị bản xem trước giao diện như hình minh họa dưới đây.



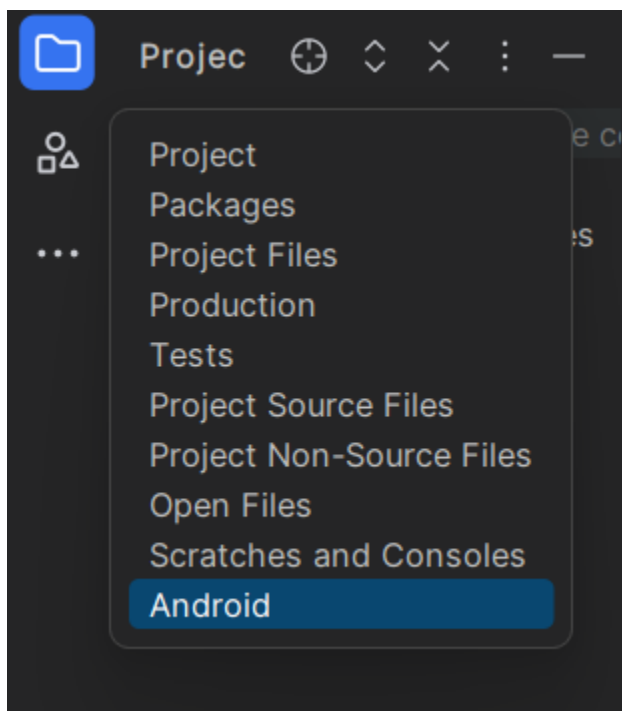
3. Nhấp vào tab **MainActivity.java** để mở trình chỉnh sửa mã như hình minh họa dưới đây.

```
activity_main.xml MainActivity.java x
1 package com.example.translation;
2
3 > import ...
10
11 <> public class MainActivity extends AppCompatActivity {
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         EdgeToEdge.enable( $this$enableEdgeToEdge: this);
17         setContentView(R.layout.activity_main);
18         ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v, insets) -> {
19             Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
20             v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
21             return insets;
22         });
23     }
24 }
```

## 2.2 Khám phá Project > Android pane

Trong phần thực hành này, bạn sẽ tìm hiểu cách tổ chức dự án trong Android Studio.

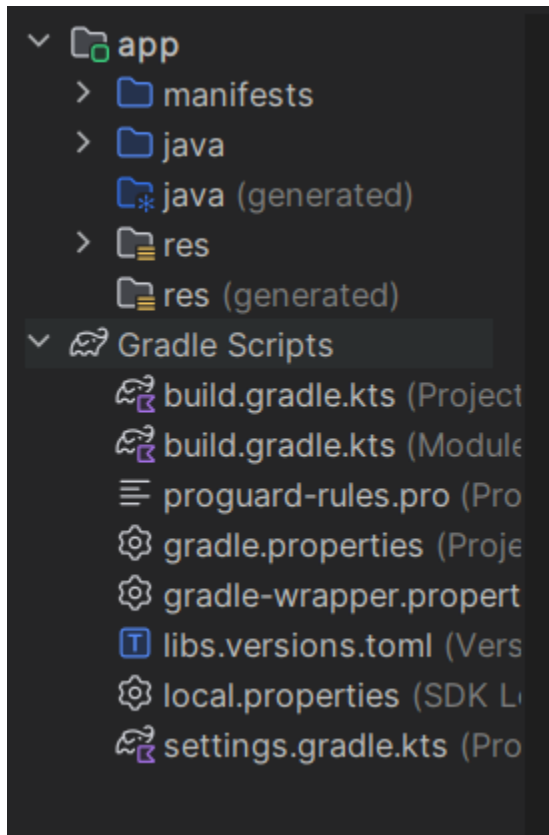
1. Nếu chưa được chọn, nhấp vào tab **Project** trong cột tab dọc bên trái của cửa sổ Android Studio. Ngăn **Project** sẽ xuất hiện.
2. Để xem dự án theo cấu trúc chuẩn của Android, chọn **Android** từ menu thả xuống ở đầu ngăn **Project**, như hình minh họa bên dưới.



## 2.3 Khám phá thư mục Gradle Scripts

Hệ thống xây dựng Gradle trong Android Studio giúp bạn dễ dàng thêm các tệp nhị phân bên ngoài hoặc các mô-đun thư viện khác vào dự án dưới dạng dependencies.

Khi bạn tạo một dự án ứng dụng mới, **Project > Android pane** sẽ hiển thị với thư mục **Gradle Scripts** được mở rộng như hình minh họa bên dưới.



Thực hiện các bước sau để tìm hiểu về hệ thống Gradle:

1. Nếu thư mục **Gradle Scripts** chưa được mở rộng, nhấp vào biểu tượng tam giác để mở nó.

Thư mục này chứa tất cả các tệp cần thiết cho hệ thống xây dựng.

2. Tìm tệp **build.gradle (Project: HelloWorld)**.

Đây là nơi chứa các tùy chọn cấu hình chung cho tất cả các mô-đun trong dự án. Mọi dự án trong Android Studio đều có một tệp Gradle build cấp cao nhất. Thông thường, bạn không cần thay đổi tệp này, nhưng hiểu nội dung của nó vẫn rất hữu ích.

Theo mặc định, tệp build cấp cao nhất sử dụng khối **buildscript** để xác định các kho lưu trữ Gradle và dependencies chung cho tất cả các mô-đun trong dự án. Khi một dependency

không phải là thư viện cục bộ hoặc tệp trong cây thư mục, Gradle sẽ tìm kiếm nó trong các kho lưu trữ trực tuyến được chỉ định trong khối **repositories** của tệp này. Mặc định, các dự án Android Studio mới sẽ khai báo **JCenter** và **Google** (bao gồm cả kho lưu trữ Google Maven) làm vị trí kho lưu trữ.

### 3. Tìm tệp **build.gradle (Module: app)**

Ngoài tệp **build.gradle** cấp dự án, mỗi mô-đun trong dự án cũng có một tệp **build.gradle** riêng. Tệp này cho phép bạn cấu hình các cài đặt xây dựng (build settings) cho từng mô-đun cụ thể (trong trường hợp này, ứng dụng **HelloWorld** chỉ có một mô-đun). Việc cấu hình các thiết lập xây dựng giúp bạn tùy chỉnh các tùy chọn đóng gói, chẳng hạn như các loại bản dựng (build types) bổ sung và các phiên bản sản phẩm (product flavors). Bạn cũng có thể ghi đè các thiết lập trong tệp **AndroidManifest.xml** hoặc tệp **build.gradle** cấp cao nhất.

Tệp này thường được chỉnh sửa nhiều nhất khi thay đổi các cấu hình ở cấp ứng dụng, chẳng hạn như khai báo dependencies trong phần **dependencies**. Bạn có thể khai báo một thư viện dependency bằng nhiều cách khác nhau. Mỗi cách cung cấp cho Gradle một hướng dẫn khác nhau về cách sử dụng thư viện đó. Ví dụ, câu lệnh: `implementation fileTree(dir: 'libs', include: ['*.jar'])` sẽ thêm tất cả các tệp **".jar"** bên trong thư mục **libs** làm dependencies.

Dưới đây là nội dung tệp **build.gradle (Module: app)** dành cho ứng dụng **HelloWorld**:

```

plugins {
    alias(libs.plugins.android.application)
}

android {
    namespace = "com.example.translation"
    compileSdk = 35

    defaultConfig {
        applicationId = "com.example.translation"
        minSdk = 29
        targetSdk = 35
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            isMinifyEnabled = false
            proguardFiles(
                getDefaultProguardFile("proguard-android-optimize.txt"),
                "proguard-rules.pro"
            )
        }
    }
}

```

4. Nhấn vào hình tam giác để tắt **Gradle Scripts**.

## 2.4 Khám phá thư mục app và res

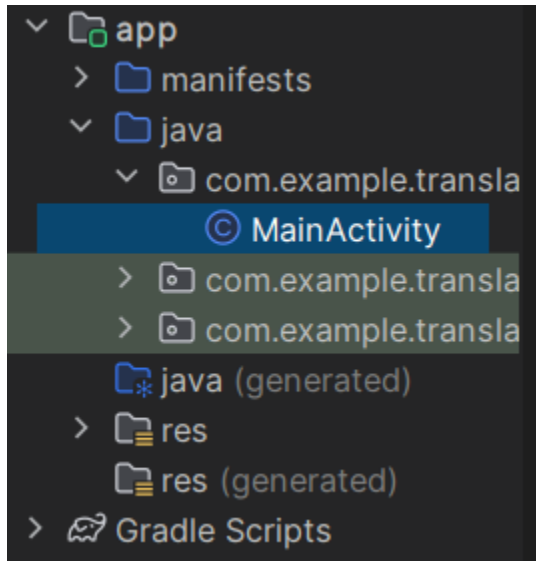
Tất cả mã nguồn và tài nguyên (resources) của ứng dụng được lưu trữ trong các thư mục **app** và **res**.

1. Mở rộng thư mục **app**, sau đó mở tiếp thư mục **java** và thư mục con **com.example.android.helloworld** để tìm tệp **MainActivity.java**. Nhấp đúp vào tệp **MainActivity.java** để mở nó trong trình chỉnh sửa mã (code editor).

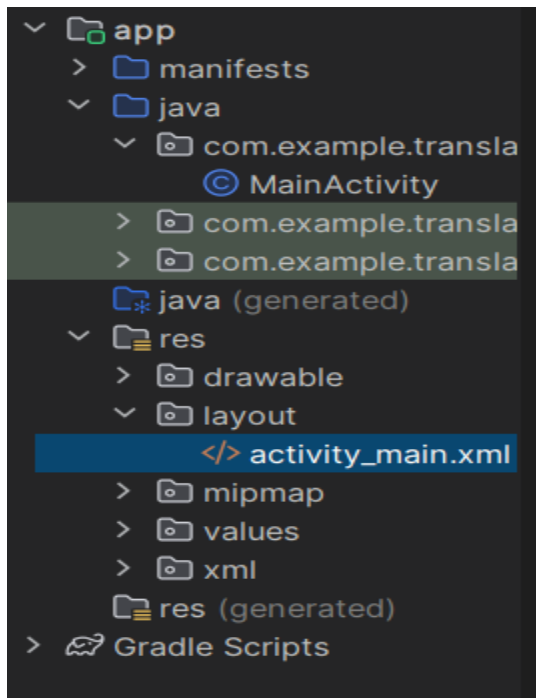
Thư mục **java** chứa các tệp lớp Java trong ba thư mục con, như minh họa trong hình trên. Thư mục **com.example.hello.helloworld** (hoặc tên miền bạn đã chỉ định) chứa tất cả các tệp cho một gói ứng dụng. Hai thư mục còn lại được sử dụng để kiểm thử (testing) và sẽ được mô tả trong một bài học khác. Đối với ứng dụng **Hello World**, chỉ có một gói duy nhất và nó

chứa tệp **MainActivity.java**. Tên của Activity đầu tiên (màn hình đầu tiên mà người dùng nhìn thấy), đồng thời khởi tạo các tài nguyên ở cấp ứng dụng, thường được đặt là **MainActivity** (phần mở rộng tệp bị ẩn trong ngăn **Project > Android**).

2. Mở rộng thư mục **res**, sau đó mở tiếp thư mục **layout**, và nhấp đúp vào tệp **activity\_main.xml** để mở nó trong trình chỉnh sửa bố cục (layout editor).



Thư mục **res** chứa các tài nguyên như bố cục (layouts), chuỗi ký tự (strings), và hình ảnh (images). Một **Activity** thường được liên kết với một bố cục giao diện người dùng (UI) được định nghĩa dưới dạng tệp XML. Thông thường, tệp này được đặt tên theo tên của **Activity** tương ứng



## 2.5 Khám phá thư mục manifests

Thư mục **manifests** chứa các tệp cung cấp thông tin quan trọng về ứng dụng của bạn cho hệ thống Android. Hệ thống cần những thông tin này trước khi có thể chạy bất kỳ mã nào của ứng dụng.

1. Mở rộng thư mục **manifests**.
2. Mở tệp **AndroidManifest.xml**.

Tệp **AndroidManifest.xml** mô tả tất cả các thành phần của ứng dụng Android. Mọi thành phần trong ứng dụng, chẳng hạn như mỗi **Activity**, đều phải được khai báo trong tệp XML này. Trong các bài học sau, bạn sẽ chỉnh sửa tệp này để thêm các tính năng và quyền truy cập (permissions). Để tìm hiểu thêm, hãy xem phần giới thiệu về **App Manifest Overview**.

### Nhiệm vụ 3: Sử dụng thiết bị ảo (emulator)

Trong nhiệm vụ này, bạn sẽ sử dụng **Android Virtual Device (AVD) Manager** để tạo một thiết bị ảo (còn gọi là **emulator**) mô phỏng cấu hình của một thiết bị Android cụ thể, sau đó chạy ứng dụng trên thiết bị ảo đó. Lưu ý rằng **Android**

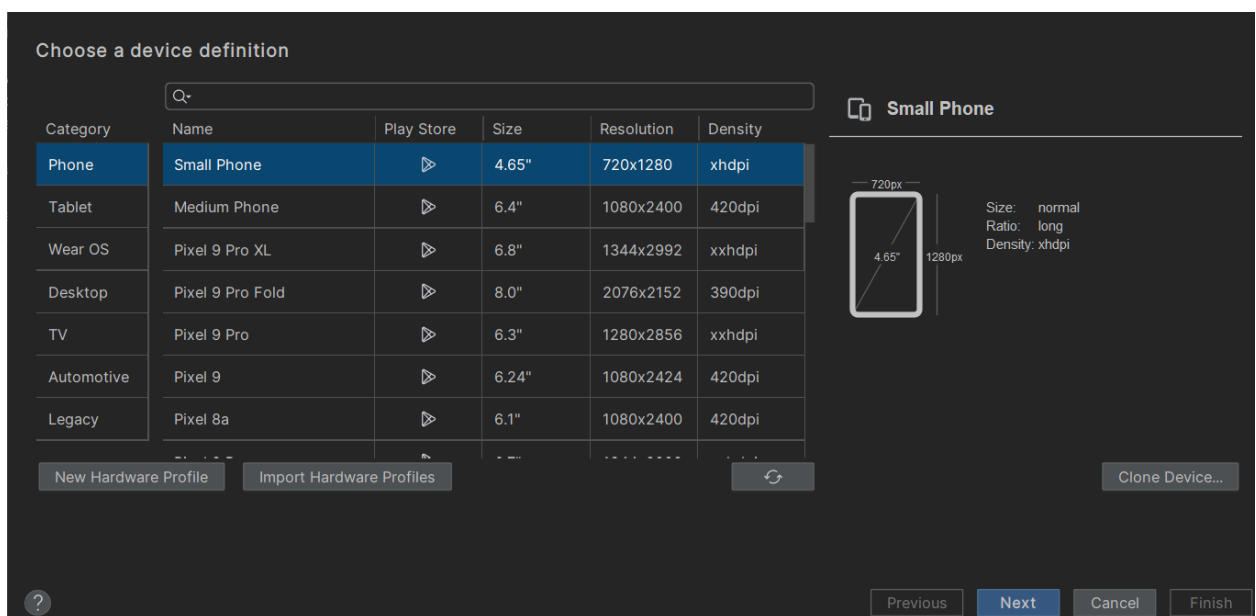
**Emulator** có các yêu cầu bổ sung so với các yêu cầu hệ thống cơ bản của **Android Studio**.

Sử dụng **AVD Manager**, bạn có thể định nghĩa các đặc điểm phần cứng của thiết bị, thiết lập mức API, dung lượng lưu trữ, giao diện (skin) và các thuộc tính khác, lưu cấu hình đó dưới dạng một thiết bị ảo. Với thiết bị ảo, bạn có thể kiểm thử ứng dụng trên nhiều cấu hình thiết bị khác nhau (ví dụ: điện thoại và máy tính bảng) với các mức API khác nhau, mà không cần sử dụng thiết bị vật lý.

### 3.1 Tạo một thiết bị ảo Android (AVD)

Để chạy trình giả lập (**emulator**) trên máy tính, bạn cần tạo một cấu hình mô tả thiết bị ảo.

1. Trong **Android Studio**, chọn **Tools > Android > AVD Manager**, hoặc nhấp vào biểu tượng **AVD Manager** trên thanh công cụ. Màn hình **Your Virtual Devices** sẽ xuất hiện. Nếu bạn đã tạo các thiết bị ảo trước đó, danh sách sẽ hiển thị chúng. Nếu chưa, danh sách sẽ trống.



2. Nhấp vào **+Create Virtual Device**. Cửa sổ **Select Hardware** sẽ xuất hiện, hiển thị danh sách các thiết bị phần cứng được cấu hình sẵn. Đối với mỗi thiết bị, bảng hiển thị các thông số sau: **Size**: Kích thước màn hình theo đường chéo (inch), **Resolution**: Độ phân giải màn hình (pixel), **Density**: Mật độ điểm ảnh (**dpi** - dots per inch).
3. Chọn một thiết bị như **Small Phone**, sau đó nhấp vào **Next**. Cửa sổ **System Image** sẽ xuất hiện.
4. Nhấp vào tab **Recommended** (nếu chưa được chọn), sau đó chọn phiên bản hệ điều hành Android mà bạn muốn chạy trên thiết bị ảo (ví dụ: **Oreo**).



Có nhiều phiên bản Android hơn những gì hiển thị trong tab **Recommended**. Bạn có thể xem thêm trong các tab **x86 Images** và **Other Images**.

Nếu có nút **Download** bên cạnh hệ điều hành bạn muốn sử dụng, điều đó có nghĩa là phiên bản đó chưa được cài đặt. Nhấp vào **Download** để bắt đầu tải xuống, sau đó nhấp vào **Finish** khi quá trình tải hoàn tất

5. Sau khi chọn phiên bản hệ điều hành, nhấp vào **Next**. Cửa sổ **Android Virtual Device (AVD)** sẽ xuất hiện. Tại đây, bạn có thể thay đổi tên của thiết bị ảo (AVD) nếu muốn. Kiểm tra lại cấu hình của bạn, sau đó nhấp vào **Finish** để hoàn tất.

### 3.2 Chạy ứng dụng trên máy ảo

Ở nhiệm vụ này, bạn sẽ chạy ứng dụng Hello World

1. Trong **Android Studio**, chọn **Run > Run app** hoặc nhấp vào biểu tượng **Run** trên thanh công cụ.

2. Trong cửa sổ **Select Deployment Target**, dưới mục **Available Virtual Devices**, chọn thiết bị ảo mà bạn vừa tạo, sau đó nhấp vào **OK**.

Trình giả lập (**emulator**) sẽ khởi động giống như một thiết bị thật. Tùy thuộc vào tốc độ của máy tính, quá trình này có thể mất một chút thời gian.

Bạn sẽ thấy ứng dụng **Hello World** hiển thị trên màn hình trình giả lập.

9:34



Hello World!



**Install successfully finished in 578 ms**

App restart successful without requiring a re-install.

## Nhiệm vụ 5: Thay đổi cấu hình Gradle của ứng dụng

Trong nhiệm vụ này, bạn sẽ thay đổi một số cấu hình trong tệp **build.gradle(Module:app)** để học cách chỉnh sửa và đồng bộ chúng với dự án **Android Studio** của mình.

### 5.1 Thay đổi phiên bản SDK tối thiểu cho ứng dụng

Thực hiện các bước sau:

1. Mở rộng thư mục **Gradle Scripts** (nếu chưa mở), sau đó nhấp đúp vào tệp **build.gradle(Module:app)**.
  - o Nội dung tệp sẽ xuất hiện trong trình chỉnh sửa mã (**code editor**).
2. Trong khối **defaultConfig**, thay đổi giá trị của **minSdkVersion** thành **17**, như ví dụ dưới đây (ban đầu được đặt là 15).

```
defaultConfig {  
    applicationId = "com.example.translation"  
    minSdk = 29  
    targetSdk = 35  
    versionCode = 1  
    versionName = "1.0"  
  
    testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"  
}  
  
buildTypes {  
    release {  
        isMinifyEnabled = false  
        proguardFiles(  
            getDefaultProguardFile("proguard-android-optimize.txt"),  
            "proguard-rules.pro"  
        )  
    }  
}
```

### 5.2 Đồng bộ cấu hình Gradle mới

Khi bạn thực hiện thay đổi trong các tệp cấu hình **build** của dự án, **Android Studio** yêu cầu bạn đồng bộ các tệp dự án để nhập các thay đổi cấu hình và kiểm tra xem chúng có gây ra lỗi biên dịch hay không.

Để đồng bộ tệp dự án, bạn có thể: Nhấp vào **Sync Now** trên thanh thông báo xuất hiện sau khi chỉnh sửa (như trong hình trước đó). Hoặc nhấp vào biểu tượng **Sync Project with Gradle Files** trên thanh công cụ.

Sau khi quá trình đồng bộ Gradle hoàn tất, bạn sẽ thấy thông báo **Gradle build finished** xuất hiện ở góc dưới bên trái của cửa sổ **Android Studio**.

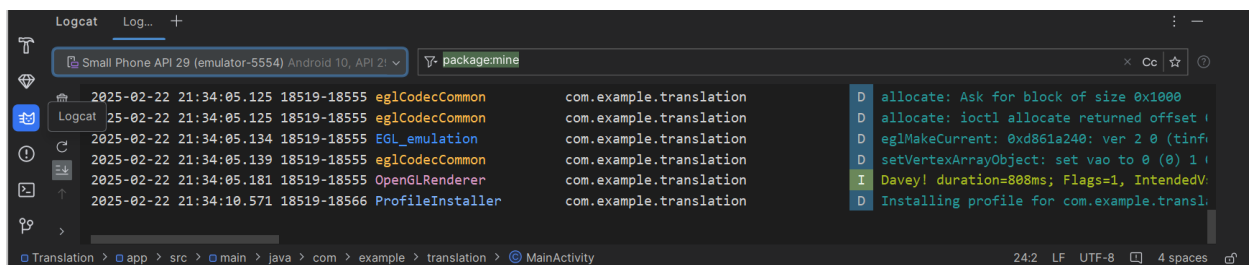
Để tìm hiểu sâu hơn về **Gradle**, hãy tham khảo tài liệu **Build System Overview** và **Configuring Gradle Builds**.

## Nhiệm vụ 6: Thêm câu lệnh log vào ứng dụng của bạn

Trong nhiệm vụ này, bạn sẽ thêm các câu lệnh Log vào ứng dụng của mình. Các câu lệnh **Log** sẽ hiển thị thông báo trong **Logcat pane**. Nhật ký (**Log messages**) là một công cụ gỡ lỗi mạnh mẽ, giúp bạn kiểm tra giá trị, theo dõi luồng thực thi và báo cáo lỗi (**exceptions**).

### 6.1 Quan sát Logcat pane

Để xem **Logcat pane**, hãy nhấp vào tab **Logcat** ở phía dưới cửa sổ **Android Studio**, như minh họa trong hình dưới đây.



Trong hình trên:

1. **Tab Logcat** để mở và đóng **Logcat pane**, hiển thị thông tin về ứng dụng của bạn khi nó đang chạy. Nếu bạn thêm các câu lệnh **Log** vào ứng dụng, các thông báo **Log** sẽ xuất hiện tại đây.
2. **Menu cấp độ Log** được đặt thành **Verbose** (mặc định), hiển thị tất cả các thông báo **Log**. Các tùy chọn khác bao gồm **Debug**, **Error**, **Info** và **Warn**.

### 6.2 Thêm câu lệnh log vào ứng dụng của bạn

Bạn có thể thêm các câu lệnh **Log** vào mã của mình để hiển thị thông tin trong Logcat. Ví dụ:

```
Log.d("MainActivity", "Hello World");
```

Các phần của thông điệp bao gồm:

- **Log**: Lớp Log dùng để gửi thông điệp log đến Logcat.
- **d**: Cấp độ log Debug để lọc và hiển thị thông điệp log trong Logcat. Các cấp độ log khác bao gồm **e** cho Error, **w** cho Warn và **i** cho Info.
- **"MainActivity"**: Đối số đầu tiên là một thẻ (tag) có thể được sử dụng để lọc thông điệp trong Logcat. Thông thường, đây là tên của Activity mà thông điệp được gửi từ đó. Tuy nhiên, bạn có thể đặt bất kỳ tên nào hữu ích cho quá trình gỡ lỗi.

Theo quy ước, các thẻ log được định nghĩa là hằng số cho Activity:

```
private static final String LOG_TAG = MainActivity.class.getSimpleName();
```

Làm theo các bước sau:

1. Mở ứng dụng **Hello World** trong Android Studio và mở tệp **MainActivity**.
2. Để tự động thêm các thư viện nhập khẩu (imports) không mơ hồ vào dự án của bạn (chẳng hạn như `android.util.Log` cần thiết để sử dụng `Log`), thực hiện:
  - Trên **Windows**: Chọn **File > Settings**.
  - Trên **macOS**: Chọn **Android Studio > Preferences**.
3. Chọn **Editor > General > Auto Import**.
  - Tích chọn tất cả các ô
  - Đặt **Insert imports on paste** thành **All**.
4. Nhấn **Apply**, sau đó nhấn **OK**.
5. Trong phương thức `onCreate()` của **MainActivity**, thêm dòng lệnh sau:

```
Log.d("MainActivity", "Hello World");
```

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Log.d( tag: "MainActivity", msg: "Hello World");  
    }  
}
```

Phương thức `onCreate()` bây giờ sẽ trông như sau

6. Nếu ngăn Logcat chưa mở, hãy nhấp vào tab **Logcat** ở cuối Android Studio để mở nó.
7. Kiểm tra xem tên của mục tiêu và tên gói của ứng dụng có đúng không.
8. Thay đổi **Log level** trong ngăn Logcat thành **Debug** (hoặc giữ nguyên **Verbose** vì có rất ít thông báo log).
9. Chạy ứng dụng của bạn.

Result

```
2025-02-25 08:57:00.262 24639-24639 MainActivity pid-24639 D Hello World
```

## Tóm tắt

- Để cài đặt **Android Studio**, hãy truy cập trang Android Studio và làm theo hướng dẫn để tải xuống và cài đặt.
- Khi tạo một ứng dụng mới, hãy đảm bảo rằng **API 15: Android 4.0.3 IceCreamSandwich** được đặt làm **Minimum SDK**.
- Để xem cấu trúc Android của ứng dụng trong **Project pane**, nhấp vào tab **Project** trong cột tab dọc, sau đó chọn **Android** trong menu thả xuống ở trên cùng.
- Chỉnh sửa tệp **build.gradle(Module:app)** khi cần thêm thư viện mới vào dự án hoặc thay đổi phiên bản thư viện.
- Mã nguồn và tài nguyên của ứng dụng được đặt trong thư mục **app** và **res**. Thư mục **java** chứa các **Activity**, các bài kiểm tra và các thành phần khác được viết bằng mã nguồn Java. Thư mục **res** chứa tài nguyên như **layout**, **chuỗi văn bản**, **hình ảnh**.
- Chỉnh sửa tệp **AndroidManifest.xml** để thêm tính năng, thành phần và quyền truy cập vào ứng dụng Android. Mọi thành phần trong ứng dụng, chẳng hạn như nhiều **Activity**, đều phải được khai báo trong tệp XML này.
- Sử dụng **Android Virtual Device (AVD) Manager** để tạo **thiết bị ảo (emulator)** nhằm chạy ứng dụng của bạn.
- Thêm các câu lệnh **Log** vào ứng dụng để hiển thị thông báo trong **Logcat pane**, đây là một công cụ cơ bản để **debug**.
- Để chạy ứng dụng trên thiết bị Android thực bằng **Android Studio**, hãy bật **USB Debugging** trên thiết bị: Mở **Settings > About phone** và nhấn **Build number** bảy lần. Quay lại màn hình trước (**Settings**) và nhấn vào **Developer options**. và chọn **USB Debugging**.

## 1.2) Giao diện người dùng tương tác đầu tiên

### a) Phần A: Giao diện người dùng tương tác đầu tiên

#### Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình của một thiết bị Android bao gồm một **hệ thống phân cấp** các đối tượng được gọi là **View** — mỗi phần tử trên màn hình đều là một **View**. Lớp **View** đại diện cho **khối xây dựng cơ bản** của tất cả các thành phần UI và là **lớp cơ sở** cho các lớp cung cấp thành phần UI tương tác như **nút bấm (Button)**, **hộp kiểm (Checkbox)**, **ô nhập văn bản (Text Entry Field)**,... Một số **lớp con phổ biến** của **View** được mô tả trong nhiều bài học bao gồm:

- **TextView**: Dùng để hiển thị văn bản.
- **EditText**: Cho phép người dùng nhập và chỉnh sửa văn bản.
- **Button** và các phần tử có thể nhấp khác (**RadioButton**, **CheckBox**, **Spinner**): Cung cấp hành vi tương tác.
- **ScrollView** và **RecyclerView**: Dùng để hiển thị nội dung có thể cuộn.
- **ImageView**: Dùng để hiển thị hình ảnh.
- **ConstraintLayout** và **LinearLayout**: Dùng để chứa các phần tử **View** khác và sắp xếp chúng trên giao diện.

Mã Java điều khiển và hiển thị giao diện người dùng (UI) nằm trong một lớp mở rộng từ **Activity**. Một **Activity** thường được liên kết với một bố cục UI được định nghĩa trong một tệp XML (**eXtended Markup Language**). Tệp XML này thường được đặt tên theo **Activity** của nó và xác định cách sắp xếp các phần tử **View** trên màn hình.

Ví dụ, mã **MainActivity** trong ứng dụng **Hello World** hiển thị bố cục được định nghĩa trong tệp **activity\_main.xml**, trong đó có một **TextView** chứa văn bản "Hello World".

Trong các ứng dụng phức tạp hơn, một **Activity** có thể. Thực hiện các hành động để phản hồi thao tác chạm của người dùng. Vẽ nội dung đồ họa. Yêu cầu dữ liệu từ cơ sở dữ liệu hoặc từ internet. Bạn sẽ tìm hiểu thêm về lớp **Activity** trong bài học tiếp theo.

Trong bài thực hành này, bạn sẽ học cách tạo ứng dụng tương tác đầu tiên của mình— một ứng dụng cho phép người dùng tương tác. Tạo một ứng dụng bằng mẫu **Empty Activity**. Sử dụng trình chỉnh sửa bố cục (**Layout Editor**) để thiết kế giao diện. Chỉnh sửa bố cục trong XML. Những kỹ năng này sẽ cần thiết để bạn hoàn thành các bài thực hành khác trong khóa học này.

#### Những thứ bạn đã biết

Bạn nên nắm vững các kiến thức sau:

- Cách cài đặt và mở **Android Studio**.
- Cách tạo ứng dụng **HelloWorld**.
- Cách chạy ứng dụng **HelloWorld**.

Những thứ bạn sẽ học

- Cách tạo một ứng dụng có **tương tác với người dùng**.
- Cách sử dụng **layout editor** để thiết kế giao diện.
- Cách chỉnh sửa **layout bằng XML**.
- Rất nhiều thuật ngữ mới. Hãy xem **glossary (bảng thuật ngữ và khái niệm)** để có định nghĩa dễ hiểu hơn.

Những thứ bạn sẽ làm

- Tạo một ứng dụng và thêm **hai nút Button** cùng một **TextView** vào layout.
- Điều chỉnh từng phần tử trong **ConstraintLayout**, ràng buộc chúng với lề và các phần tử khác.
- Thay đổi **thuộc tính của các phần tử giao diện**.
- Chỉnh sửa **layout trong XML**.
- Trích xuất **chuỗi văn bản cứng (hardcoded strings)** vào **tài nguyên chuỗi (string resources)**.
- Triển khai **phương thức xử lý sự kiện khi nhấn nút (click-handler methods)** để hiển thị thông báo trên màn hình khi người dùng nhấn vào từng nút.

Tổng quan ứng dụng

The HelloToast app consists of two Button elements and one TextView. When the user taps the first Button, it displays a short message (a **Toast**) on the screen. Tapping the second Button increases a "click" counter displayed in the TextView, which starts at zero. Here's what the finished app looks like:





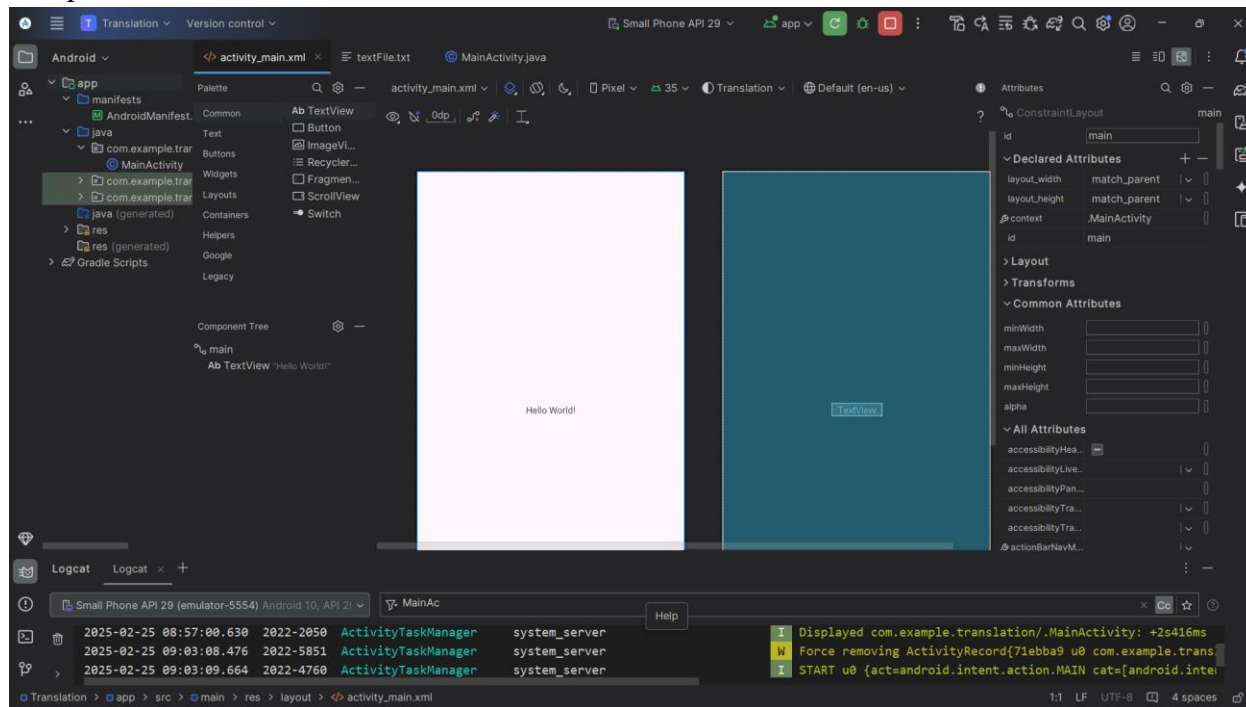
# Nhiệm vụ 1: Tập và khám phá dự án mới

Trong bài thực hành này, bạn sẽ thiết kế và triển khai một dự án cho ứng dụng **HelloToast**. Một liên kết đến mã nguồn giải pháp sẽ được cung cấp ở cuối bài.

## 1.1 Tạo 1 dự án Android Studio

## 1.2 Khám phá trình chỉnh sửa Layout

- 1 Trong **Project** > **Android** pane, điều hướng đến thư mục **app** > **res** > **layout** và nhấp đúp vào tệp **activity\_main.xml** để mở nó (nếu chưa mở).
- 2 Nhấp vào tab **Design** nếu nó chưa được chọn. Bạn sử dụng tab này để thao tác các phần tử và bố cục, còn tab **Text** để chỉnh sửa mã XML của bố cục.
- 3 **Palettes** pane hiển thị các phần tử giao diện người dùng (UI elements) mà bạn có thể sử dụng trong bố cục của ứng dụng.
- 4 **Component tree** pane hiển thị cây phân cấp của các phần tử UI. Các phần tử này được tổ chức theo dạng cây với quan hệ cha - con, trong đó một phần tử con kế thừa thuộc tính của phần tử cha. Trong hình minh họa, **TextView** là phần tử con của **ConstraintLayout**. Bạn sẽ tìm hiểu về các phần tử này trong bài học sau.
- 5 **Design** và **Blueprint** panes của trình chỉnh sửa bố cục (layout editor) hiển thị các phần tử giao diện. Trong hình minh họa, bố cục chỉ có một phần tử: **TextView** với nội dung "Hello World".
- 6 **Attributes** tab hiển thị bảng thuộc tính (**Attributes** pane) để thiết lập các thuộc tính cho một phần tử UI.

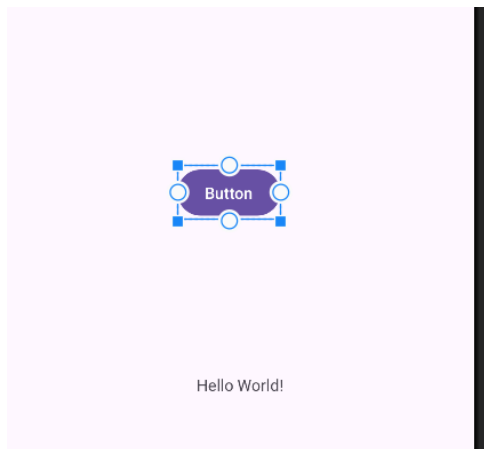


## Task 2: Thêm các phần tử View trong trình chỉnh sửa bố cục (layout editor)

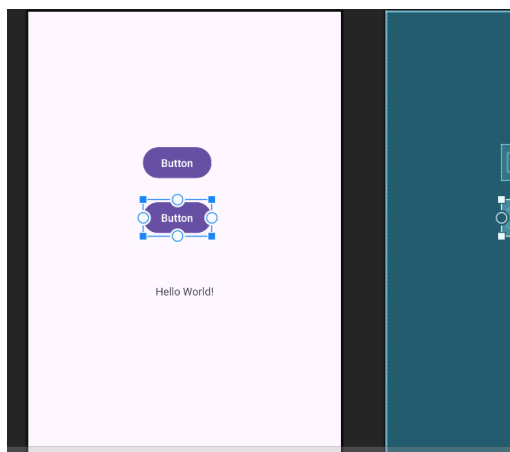
Trong nhiệm vụ này, bạn sẽ tạo bố cục giao diện người dùng (UI) cho ứng dụng HelloToast trong trình chỉnh sửa bố cục bằng cách sử dụng các tính năng của **ConstraintLayout**. Bạn có thể tạo các ràng buộc (constraints) một cách thủ công, như sẽ được hướng dẫn sau, hoặc tự động bằng công cụ **Autoconnect**.

### 2.1 Kiểm tra ràng buộc của phần tử

### 2.2 Thêm các nút bấm



### 2.3 Thêm nút thứ vào layout



## Nhiệm vụ 3: Thay đổi thuộc tính UI của phần tử

Bảng **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện người dùng (UI). Các thuộc tính này thường được gọi là **properties (thuộc tính)**, và bạn có thể tìm thấy danh sách đầy đủ trong tài liệu của lớp **View**.

Trong nhiệm vụ này, bạn sẽ nhập các giá trị mới và thay đổi các thuộc tính quan trọng của **Button**, có thể áp dụng cho hầu hết các phần tử **View**.

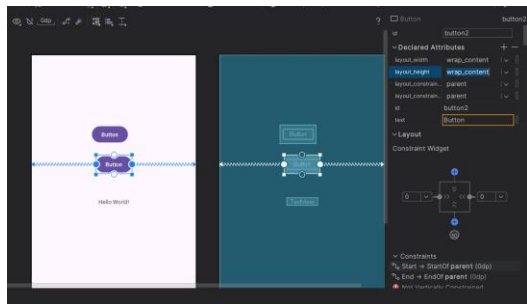
### 3.1 Thay đổi kích thước Button

Trình chỉnh sửa bố cục cung cấp các tay cầm thay đổi kích thước ở bốn góc của một View để bạn có thể thay đổi kích thước nhanh chóng. Bạn có thể kéo các tay cầm ở mỗi góc của View để thay đổi kích thước của nó, nhưng làm như vậy sẽ cố định các kích thước chiều rộng và chiều cao. Tránh cố định kích thước cho hầu hết các phần tử View, vì các kích thước cố định không thể thích ứng với nội dung và kích thước màn hình khác nhau.

Thay vào đó, hãy sử dụng bảng **Attributes** ở bên phải trình chỉnh sửa bố cục để chọn một chế độ kích thước không sử dụng kích thước cố định. Bảng **Attributes** bao gồm một bảng định cỡ hình vuông ở trên cùng, được gọi là **view inspector**.

Thuộc tính `layout_width` và `layout_height` trong bảng Thuộc tính thay đổi khi bạn điều chỉnh các điều khiển chiều cao và chiều rộng trong trình kiểm tra. Những thuộc tính này có thể nhận một trong ba giá trị đối với bố cục, là `ConstraintLayout`:

- Cài đặt `match_constraint` mở rộng phần tử View để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao—tới một lề nếu có đặt lề. Trong trường hợp này, phần tử cha là `ConstraintLayout`. Bạn sẽ tìm hiểu thêm về `ConstraintLayout` trong nhiệm vụ tiếp theo.
- Cài đặt `wrap_content` thu nhỏ kích thước của phần tử View sao cho vừa đủ để chứa nội dung của nó. Nếu không có nội dung, phần tử View sẽ trở nên vô hình.
- Để chỉ định một kích thước cố định có thể điều chỉnh theo kích thước màn hình của thiết bị, hãy sử dụng một số cố định của pixel độc lập với mật độ (dp). Ví dụ, `16dp` có nghĩa là 16 pixel độc lập với mật độ.



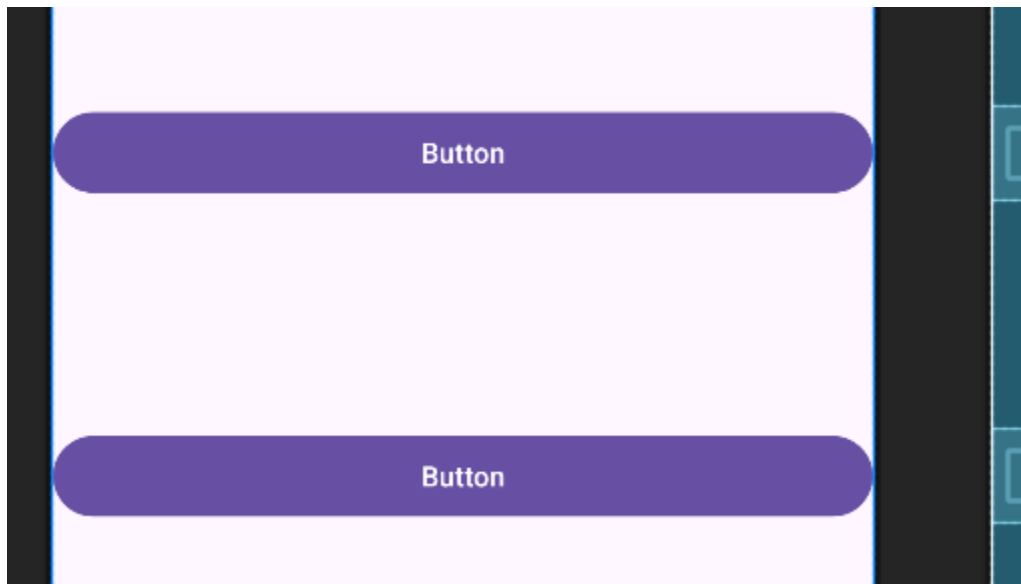
## 3.2 Thay đổi thuộc tính Button

Để xác định từng View một cách duy nhất trong bố cục của một Activity, mỗi View hoặc lớp con của View (chẳng hạn như Button) cần một ID duy nhất. Và để có thể sử dụng, các phần tử Button cần có văn bản. Các phần tử View cũng có thể có nền là màu hoặc hình ảnh.

Bảng Thuộc tính cung cấp quyền truy cập vào tất cả các thuộc tính mà bạn có thể gán cho một phần tử View. Bạn có thể nhập giá trị cho từng thuộc tính, chẳng hạn như `android:id`, `background`, `textColor`, và `text`.

Hình động sau đây minh họa cách thực hiện các bước sau:

1. Sau khi chọn Button đầu tiên, chỉnh sửa trường ID ở đầu bảng Thuộc tính thành `button_toast` cho thuộc tính `android:id`, được sử dụng để xác định phần tử trong bố cục.
2. Đặt thuộc tính `background` thành `@color/colorPrimary`. (Khi bạn nhập `@c`, các lựa chọn sẽ xuất hiện để dễ dàng chọn.)
3. Đặt thuộc tính `textColor` thành `@android:color/white`.
4. Chỉnh sửa thuộc tính `text` thành `Toast`.
5. Thực hiện các thay đổi thuộc tính tương tự cho Button thứ hai, sử dụng `button_count` làm ID, `Count` cho thuộc tính `text`, và cùng màu nền và màu chữ như các bước trước.



## Nhiệm vụ 4: Thêm TextEdit và sửa thuộc tính

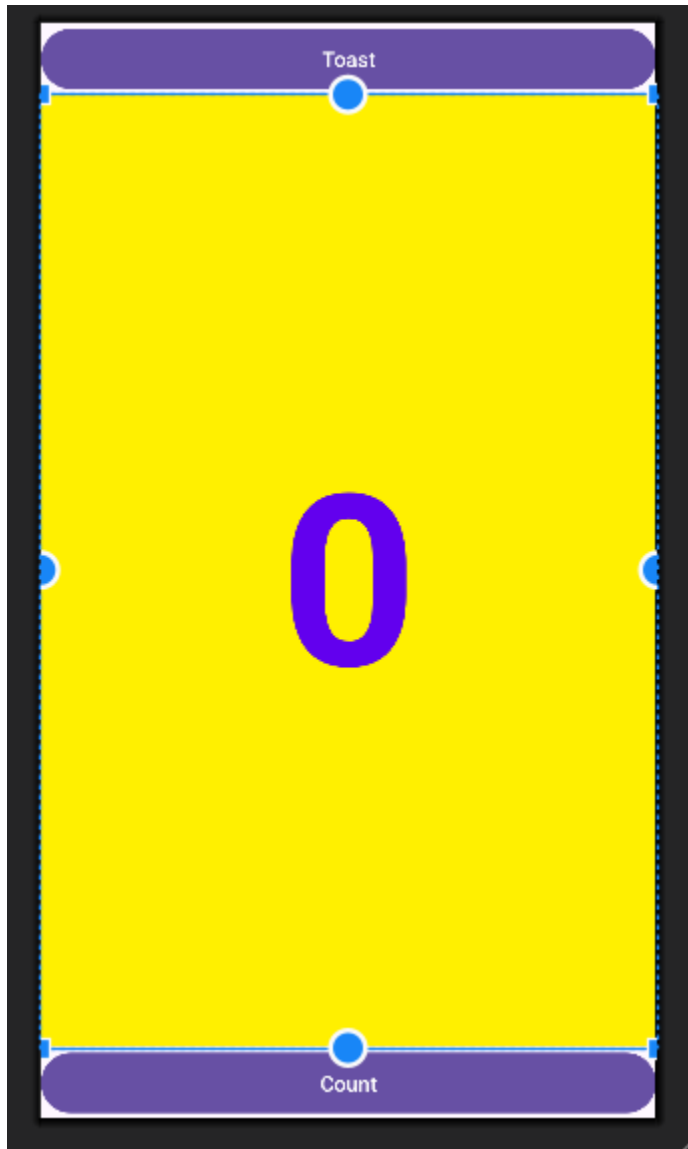
Một trong những lợi ích của `ConstraintLayout` là khả năng căn chỉnh hoặc ràng buộc các phần tử so với các phần tử khác. Trong nhiệm vụ này, bạn sẽ thêm một `TextView` vào giữa bố cục và ràng buộc nó theo chiều ngang với lề và theo chiều

dọc với hai phần tử `Button`. Sau đó, bạn sẽ thay đổi các thuộc tính cho `TextView` trong bảng `Attributes`.

#### 4.1 Thêm `TextView` và ràng buộc

#### 4.2 Chỉnh thuộc tính `TextView`

- Đặt ID thành `show_count`.
- Đặt văn bản thành `0`.
- Đặt `textSize` thành `160sp`.
- Đặt `textStyle` thành `B` (in đậm) và `textAlignment` thành `ALIGNCENTER` (căn giữa đoạn văn).
- Thay đổi điều khiển kích thước chiều ngang và chiều dọc (`layout_width` và `layout_height`) thành `match_constraint`.
- Đặt `textColor` thành `@color/colorPrimary`.



## Nhiệm vụ 5: Chỉnh sửa giao diện trong XML

Bố cục của ứng dụng Hello Toast gần như đã hoàn thành! Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi phần tử UI trong Cây Thành Phần (Component Tree). Di chuột qua các dấu chấm than này để xem thông báo cảnh báo, như hình dưới đây. Cùng một cảnh báo xuất hiện cho cả ba phần tử: các chuỗi được mã hóa cứng (hardcoded strings) nên sử dụng tài nguyên (resources).

Cách dễ nhất để khắc phục các vấn đề về bố cục là chỉnh sửa bố cục trong XML. Mặc dù trình chỉnh sửa bố cục là một công cụ mạnh mẽ, nhưng một số thay đổi sẽ dễ thực hiện hơn trực tiếp trong mã nguồn XML.

### 5.1 Mở XML code cho giao diện

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/button"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Toast"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        tools:layout_editor_absoluteY="40dp" />


    <Button
        android:id="@+id/button2"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Count"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        tools:layout_editor_absoluteY="100dp" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

### 5.2 Trích xuất chuỗi

Thay vì mã hóa cứng chuỗi, một phương pháp tốt nhất là sử dụng tài nguyên chuỗi, đại diện cho các chuỗi. Việc đặt chuỗi trong một tệp riêng biệt giúp dễ dàng

quản lý hơn, đặc biệt nếu bạn sử dụng các chuỗi này nhiều lần. Ngoài ra, tài nguyên chuỗi là bắt buộc để dịch và bản địa hóa ứng dụng của bạn, vì bạn cần tạo một tệp tài nguyên chuỗi cho từng ngôn ngữ.

```
<resources>
    <string name="app_name">Hello Toast</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    <string name="count_initial_value">0</string>
     <string name="toast_message">Hello Toast!</string>
</resources>
```

## Task 6: Thêm xử lý onClick cho Button

### 6.1 Thêm thuộc tính onClick và xử lý vào mỗi Button

```
<Button
    android:id="@+id/button_toast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:shadowColor="@color/black"
    android:text="@string/button_label_toast"
    android:textColor="#FFFFFF"
    android:textColorHighlight="@color/white"
    android:textColorHint="@color/white"
    android:textColorLink="#B52525"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    app:strokeColor="#FAF9F9"
    android:onClick="showToast"
/>
```

### 6.2 Chỉnh sửa xử lý Toast Button

Bây giờ, bạn sẽ chỉnh sửa phương thức `showToast()` — trình xử lý nhấp của Button Toast trong `MainActivity` — sao cho nó hiển thị một thông báo. Một Toast cung cấp cách hiển thị một thông báo đơn giản trong một cửa sổ popup nhỏ. Nó chỉ chiếm không gian cần thiết cho thông báo. Activity hiện tại vẫn hiển thị và tương tác được. Một Toast có thể hữu ích để kiểm tra tính tương tác trong ứng dụng của bạn — hãy thêm một thông báo Toast để hiển thị kết quả khi người dùng nhấn vào Button hoặc thực hiện một hành động.

```
> public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d( tag: "MainActivity", msg: "Hello World");
    }

    1 usage
    public void showToast(View view) {
        Toast toast = Toast.makeText( context: this, R.string.toast_message, Toast.LENGTH_SHORT);
        toast.show();
    }

    1 usage
    public void countUp(View view) {
    }
}
```

### 6.3 Chỉnh sửa xử lý nút Count

Bây giờ, bạn sẽ chỉnh sửa phương thức `countUp()` — trình xử lý nhấp của Button Count trong `MainActivity` — để nó hiển thị số đếm hiện tại sau khi nhấn vào Count. Mỗi lần nhấn sẽ tăng số đếm lên một đơn vị.

Mã cho trình xử lý phải:

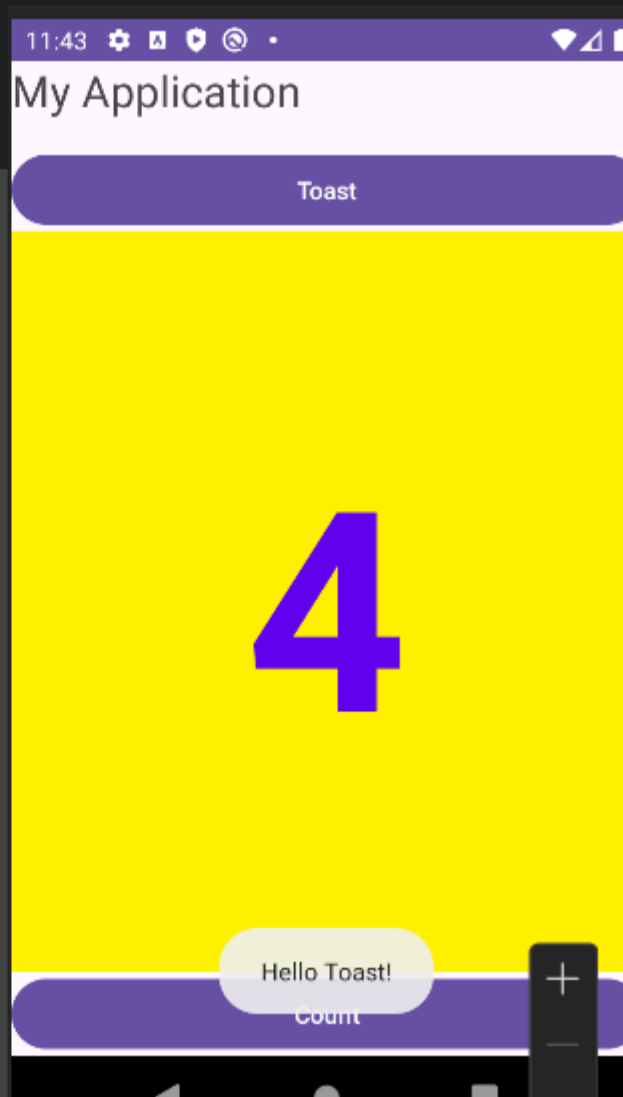
- Theo dõi số đếm khi nó thay đổi.
- Gửi số đếm đã cập nhật đến `TextView` để hiển thị.



```
1 usage
public void showToast(View view) {
    Toast toast = Toast.makeText(context: this, R.string.toast_message, Toast.LENGTH_SHORT);
    toast.show();
}

1 usage
public void countUp(View view) {
    ++mCount;
    if (mShowCount != null)
        mShowCount.setText(Integer.toString(mCount));
}
}
```

.LENGTH\_SHORT);



## b) Trình chỉnh sửa bố cục

Như bạn đã học trong Bài 1.2 Phần A: Giao diện người dùng tương tác đầu tiên của bạn, bạn có thể xây dựng giao diện người dùng (UI) bằng cách sử dụng **ConstraintLayout** trong trình chỉnh sửa bố cục. **ConstraintLayout** sắp xếp các phần tử giao diện người dùng trong một bố cục bằng cách sử dụng các ràng buộc với các phần tử khác và với các cạnh của bố cục. **ConstraintLayout** được thiết kế để giúp bạn dễ dàng kéo các phần tử giao diện người dùng vào trình chỉnh sửa bố cục.

**ConstraintLayout** là một **ViewGroup**, một loại **View** đặc biệt có thể chứa các **View** khác (gọi là **children** hoặc **child views**). Trong phần thực hành này, bạn sẽ tìm hiểu thêm về các tính năng của **ConstraintLayout** và trình chỉnh sửa bố cục.

Phần thực hành này cũng giới thiệu hai lớp con khác của **ViewGroup**:

- **LinearLayout**: Một nhóm căn chỉnh các phần tử **View** con bên trong theo chiều ngang hoặc chiều dọc.
- **RelativeLayout**: Một nhóm các phần tử **View** con, trong đó mỗi phần tử **View** được định vị và căn chỉnh dựa trên các phần tử **View** khác trong **ViewGroup**. Vị trí của các phần tử **View** con được mô tả theo mối quan hệ với nhau hoặc với **ViewGroup** cha.

Những gì bạn nên biết trước  
Bạn nên có khả năng:

- Tạo ứng dụng **Hello World** bằng **Android Studio**.
- Chạy ứng dụng trên **trình giả lập** hoặc **thiết bị thực**.
- Tạo bố cục đơn giản cho ứng dụng bằng **ConstraintLayout**.
- Trích xuất và sử dụng **tài nguyên chuỗi (string resources)**.

Những gì bạn sẽ học

- Cách tạo **biến thể bố cục** cho **chế độ ngang (landscape)**.
- Cách tạo **biến thể bố cục** cho **máy tính bảng và màn hình lớn hơn**.
- Cách sử dụng **ràng buộc đường cơ sở (baseline constraint)** để căn chỉnh các **phần tử UI** với văn bản.
- Cách sử dụng **các nút pack và align** để căn chỉnh các phần tử trong bố cục.
- Cách định vị các **View** trong **LinearLayout**.
- Cách định vị các **View** trong **RelativeLayout**.

Những gì bạn sẽ làm

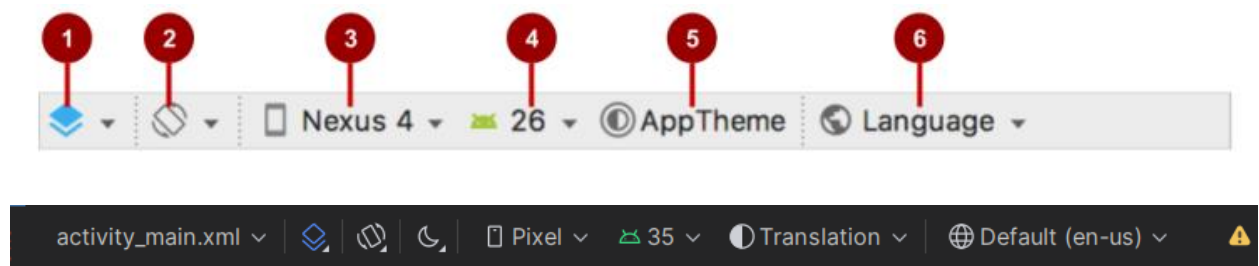
- Tạo **biến thể bố cục** cho **màn hình ngang**.
- Tạo **biến thể bố cục** cho **máy tính bảng và màn hình lớn**.
- **Chỉnh sửa bố cục** để thêm các **ràng buộc (constraints)** cho các phần tử UI.

- Sử dụng ràng buộc đường cơ sở (**baseline constraint**) để căn chỉnh phần tử với văn bản.
- Sử dụng các nút **pack** và **align** để căn chỉnh phần tử.
- Chuyển đổi bố cục sang **LinearLayout**.
- Định vị các phần tử trong **LinearLayout**.
- Chuyển đổi bố cục sang **RelativeLayout**.
- Sắp xếp lại các **View** trong bố cục chính sao cho liên kết với nhau.

## Nhiệm vụ 1: Tạo các biến thể bố cục

Trong bài học trước, thử thách lập trình yêu cầu thay đổi bố cục của ứng dụng Hello Toast để phù hợp với chế độ hiển thị ngang hoặc dọc. Trong nhiệm vụ này, bạn sẽ học một cách dễ dàng hơn để tạo các biến thể của bố cục cho chế độ ngang (còn gọi là **landscape**) và chế độ dọc (còn gọi là **portrait**) trên điện thoại, cũng như cho các màn hình lớn hơn như máy tính bảng.

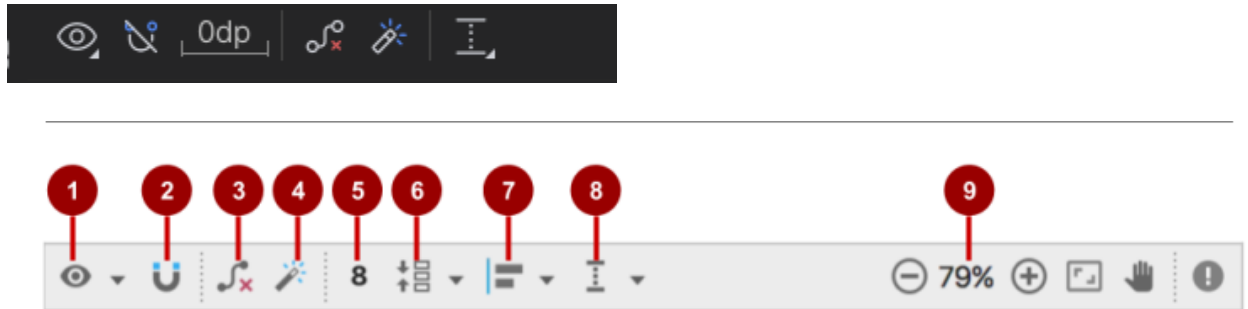
Trong nhiệm vụ này, bạn sẽ sử dụng một số nút trong hai thanh công cụ trên cùng của trình chỉnh sửa bố cục. Thanh công cụ trên cùng cho phép bạn cấu hình cách hiển thị bản xem trước bố cục trong trình chỉnh sửa bố cục.



Trong hình trên:

1. **Chọn Bề mặt Thiết kế:** Chọn **Design** để hiển thị bản xem trước có màu của bố cục hoặc **Blueprint** để chỉ hiển thị đường viền của từng phần tử UI. Để xem cả hai chế độ cạnh nhau, chọn **Design + Blueprint**.
2. **Chế độ hiển thị trong Trình chỉnh sửa:** Chọn **Portrait** hoặc **Landscape** để xem trước bố cục theo chiều dọc hoặc ngang. Điều này hữu ích để xem trước bố cục mà không cần chạy ứng dụng trên trình giả lập hoặc thiết bị. Để tạo bố cục thay thế, chọn **Create Landscape Variation** hoặc các biến thể khác.
3. **Thiết bị trong Trình chỉnh sửa:** Chọn loại thiết bị (**điện thoại/máy tính bảng, Android TV hoặc Android Wear**).
4. **Phiên bản API trong Trình chỉnh sửa:** Chọn phiên bản Android để hiển thị bản xem trước.
5. **Chủ đề trong Trình chỉnh sửa:** Chọn một chủ đề (ví dụ: **AppTheme**) để áp dụng cho bản xem trước.
6. **Ngôn ngữ trong Trình chỉnh sửa:** Chọn ngôn ngữ và khu vực để xem trước. Danh sách này chỉ hiển thị các ngôn ngữ có trong tài nguyên chuỗi (**string resources**). Bạn cũng có

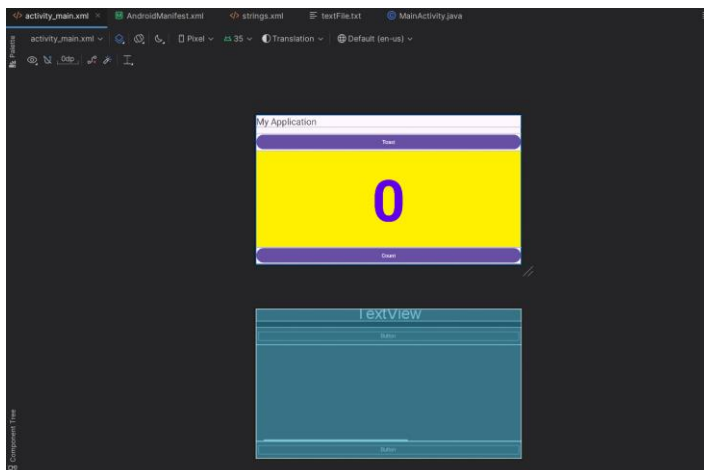
thể chọn **Preview as Right To Left** để xem bố cục dưới dạng ngôn ngữ viết từ phải sang trái (RTL).



Trong hình trên:

1. **Hiển thị:** Chọn **Show Constraints** và **Show Margins** để hiển thị hoặc ẩn chúng trong bản xem trước.
2. **Tự động kết nối (Autoconnect):** Bật hoặc tắt **Autoconnect**. Khi bật, bạn có thể kéo bất kỳ phần tử nào (ví dụ: **Button**) vào bố cục để tự động tạo ràng buộc với bố cục cha.
3. **Xóa tất cả ràng buộc:** Xóa toàn bộ ràng buộc trong bố cục.
4. **Suy luận ràng buộc (Infer Constraints):** Tạo ràng buộc bằng cách suy luận.
5. **Lề mặc định (Default Margins):** Đặt lề mặc định.
6. **Gói (Pack):** Gom nhóm hoặc mở rộng các phần tử đã chọn.
7. **Căn chỉnh (Align):** Căn chỉnh các phần tử đã chọn.
8. **Đường hướng dẫn (Guidelines):** Thêm đường hướng dẫn dọc hoặc ngang.
9. **Điều khiển thu phóng/di chuyển:** Phóng to hoặc thu nhỏ

### 1.1 Xem trước bố cục trong chế độ ngang (horizontal orientation).



1.2 Tạo một biến thể bố cục cho chế độ ngang (horizontal orientation).

Sự khác biệt trực quan giữa chế độ dọc và chế độ ngang đối với bố cục này là chữ số (0) trong phần tử **show\_count** (TextView) nằm quá thấp trong chế độ ngang—quá gần với nút **Count**. Tùy thuộc vào thiết bị hoặc trình giả lập bạn sử dụng, phần tử **TextView** có thể trông quá lớn hoặc không căn giữa do kích thước văn bản được cố định ở **160sp**.

1.3) Trình chỉnh sửa bố cục

1.4) Văn bản và các chế độ cuộn

1.5) Tài nguyên có sẵn

## Bài 2) Activities

2.1) Activity và Intent

2.2) Vòng đời của Activity và trạng thái

2.3) Intent ngầm định

## Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ

3.1) Trình gỡ lỗi

3.2) Kiểm thử đơn vị

3.3) Thư viện hỗ trợ

## **CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG**

### **Bài 1) Tương tác người dùng**

- 1.1) Hình ảnh có thể chọn**
- 1.2) Các điều khiển nhập liệu**
- 1.3) Menu và bộ chọn**
- 1.4) Điều hướng người dùng**
- 1.5) RecyclerView**

### **Bài 2) Trải nghiệm người dùng thú vị**

- 2.1) Hình vẽ, định kiểu và chủ đề**
- 2.2) Thẻ và màu sắc**
- 2.3) Bố cục thích ứng**

### **Bài 3) Kiểm thử giao diện người dùng**

- 3.1) Espresso cho việc kiểm tra UI**

## **CHƯƠNG 3. LÀM VIỆC TRONG NỀN**

### **Bài 1) Các tác vụ nền**

- 1.1) AsyncTask**
- 1.2) AsyncTask và AsyncTaskLoader**
- 1.3) Broadcast receivers**

### **Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền**

- 2.1) Thông báo**
- 2.2) Trình quản lý cảnh báo**
- 2.3) JobScheduler**

## **CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG**

### **Bài 1) Tùy chọn và cài đặt**

**1.1) Shared preferences**

**1.2) Cài đặt ứng dụng**

### **Bài 2) Lưu trữ dữ liệu với Room**

**2.1) Room, LiveData và ViewModel**

**2.2) Room, LiveData và ViewModel**