**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](#_Toc191662194)

[Bài 1) Tạo ứng dụng đầu tiên 3](#_Toc191662195)

[1.1) Android Studio và Hello World 3](#_Toc191662196)

[1.2) Tổng quan về ứng dụng 4](#_Toc191662197)

[1.3) Giao diện người dùng tương tác đầu tiên 24](#_Toc191662198)

[1.4) Trình chỉnh sửa bố cục 55](#_Toc191662200)

[1.5) Văn bản và các chế độ cuộn 79](#_Toc191662201)

[1.6) Tài nguyên có sẵn 79](#_Toc191662202)

[Bài 2) Activities 79](#_Toc191662203)

[2.1) Activity và Intent 79](#_Toc191662204)

[2.2) Vòng đời của Activity và trạng thái 79](#_Toc191662205)

[2.3) Intent ngầm định 79](#_Toc191662206)

[Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ 79](#_Toc191662207)

[3.1) Trình gỡ lỗi 79](#_Toc191662208)

[3.2) Kiểm thử đơn vị 79](#_Toc191662209)

[3.3) Thư viện hỗ trợ 79](#_Toc191662210)

[CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG 80](#_Toc191662211)

[Bài 1) Tương tác người dùng 80](#_Toc191662212)

[1.1) Hình ảnh có thể chọn 80](#_Toc191662213)

[1.2) Các điều khiển nhập liệu 80](#_Toc191662214)

[1.3) Menu và bộ chọn 80](#_Toc191662215)

[1.4) Điều hướng người dùng 80](#_Toc191662216)

[1.5) RecycleView 80](#_Toc191662217)

[Bài 2) Trải nghiệm người dùng thú vị 80](#_Toc191662218)

[2.1) Hình vẽ, định kiểu và chủ đề 80](#_Toc191662219)

[2.2) Thẻ và màu sắc 80](#_Toc191662220)

[2.3) Bố cục thích ứng 80](#_Toc191662221)

[Bài 3) Kiểm thử giao diện người dùng 80](#_Toc191662222)

[3.1) Espresso cho việc kiểm tra UI 80](#_Toc191662223)

[CHƯƠNG 3. LÀM VIỆC TRONG NỀN 80](#_Toc191662224)

[Bài 1) Các tác vụ nền 80](#_Toc191662225)

[1.1) AsyncTask 80](#_Toc191662226)

[1.2) AsyncTask và AsyncTaskLoader 80](#_Toc191662227)

[1.3) Broadcast receivers 80](#_Toc191662228)

[Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền 80](#_Toc191662229)

[2.1) Thông báo 80](#_Toc191662230)

[2.2) Trình quản lý cảnh báo 80](#_Toc191662231)

[2.3) JobScheduler 80](#_Toc191662232)

[CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG 80](#_Toc191662233)

[Bài 1) Tùy chọn và cài đặt 80](#_Toc191662234)

[1.1) Shared preferences 81](#_Toc191662235)

[1.2) Cài đặt ứng dụng 81](#_Toc191662236)

[Bài 2) Lưu trữ dữ liệu với Room 81](#_Toc191662237)

[2.1) Room, LiveData và ViewModel 81](#_Toc191662238)

[2.2) Room, LiveData và ViewModel 81](#_Toc191662239)

# LÀM QUEN

## Tạo ứng dụng đầu tiên

### 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.

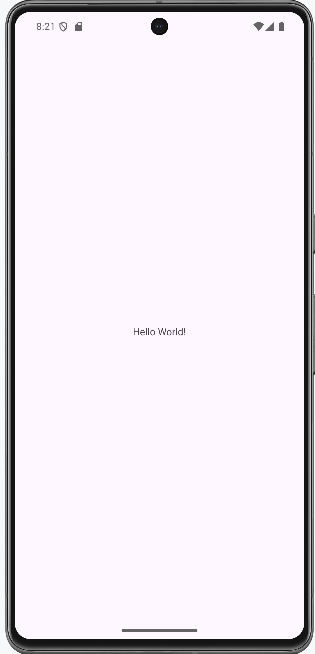
Những gì bạn sẽ làm

* 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 về ứng dụng

Sau khi thành công tải xuống **Android Studio** , từ bảng mẫu , bạn sẽ tạo dự án mới cho ứng dụng Hello World. Ứng dụng đơn giản này sẽ hiển thị dòng chữ “**Hello World**” trên màn hình của máy ảo hay thiết bị vật lý của bạn.

Ứng dụng khi hoạt động sẽ trông như thế này:



#### Nhiệm vụ 1: Cài đặt Android Studio

**Android Studio** cung cấp một môi trường phát triển được tích hợp hoàn chỉnh (IDE) .

Bao gồm trình chỉnh sửa mã cao cấp và một tập các mẫu ứng dụng. Ngoài ra , nó còn chứa các công cụ để phát triển ,gỡ lỗi , thử nghiệm và tối ưu hiệu suất , Điều này giúp việc phát triển ứng dụng trở lên nhanh chóng và dễ dàng hơn . Bạn có thể kiểm thử ứng dụng của mình bằng các trình giả lập được cấu hình sẵn hoặc trên thiết bị di động ,tạo ứng dụng chính thức và phát hành trên cửa hàng **Google Play**.

**Android Studio** có sẵn cho máy tính chạy các hệ điều hành như : **Windows , Linux hay MacOS**. Phiên bản mới nhất của **OpenJDK** cũng được tích hợp sẵn trong **Android Studio**.

Để thiết lập và chạy **Android Studio**  , trước tiên hãy kiểm tra “Những yêu cầu về thiết bị “ để đảm bảo thiết bị của bạn đáp ứng được những yêu cầu này . Việc cài đặt trên mọi nền tảng là tương tự nhau.

Khi cài đặt lưu ý một số điều dưới đây:

1. Truy cập trang web dành dành cho nhà phát triển **Android** và làm theo hướng dẫn để tải xuống và cài đặt **Android Studio**.
2. Chấp nhận cấu hình mặc định tại tất cả các bước, đảm bảo tất cả các thành phần được chọn để cài đặt.
3. Sau khi tải xuống hoàn tất ,**Setup Wizard** sẽ tải và cài đặt một số thành phần bổ sung bao gồm **Android SDK** . Điều này sẽ mất một chút thời gian , phụ thuộc vào tốc độ **Internet** của bạn .
4. Khi việc tải xuống hoàn thành , **Android Studio** sẽ khởi động , và **Enjoy your life.**

#### Nhiệm vụ 2 : Tạo ứng dụng “ Hello World”

Trong nhiệm vụ này , bạn sẽ tạo một ứng dụng hiển thị “ Hello World” để xác nhận rằng Android Studio đã được cài đặt chính xác , đồng thời tìm hiểu những kiến thức cơ bản về phát triển bằng **Android Studio.**

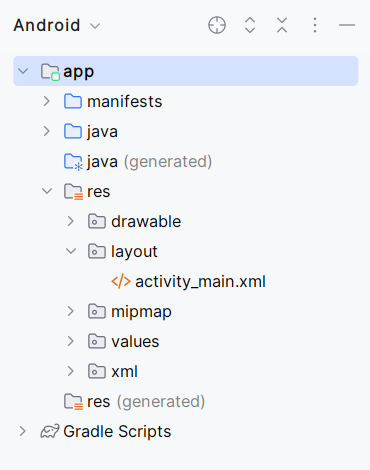
##### 2.1 Tạo dự án cho ứng dụng.

1. Mở Android Studio
2. Click **File -> New Project .**
3. **Empty Views Activity -> Next.**
4. Nhập **tên** **dự án** “Hello World” , **Tên gói (Tên miền viết ngược)**, **Địa chỉ lưu trữ , ngôn ngữ ( Java) .**
5. Nhấn **Finish.**

**Android Studio** sẽ tự động tạo thư mục cho dự án của bạn , và xây dựng dự án bằng Gradle.

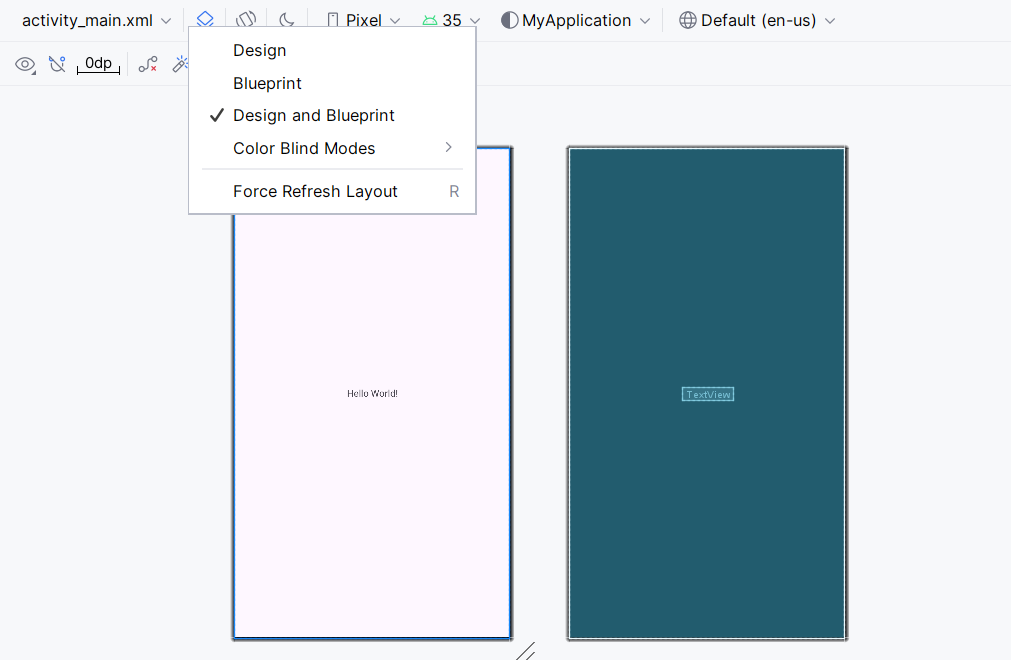
Trình chỉnh sửa của **Android Studio** sẽ xuất hiện . Làm theo các bước sau:

1. Nhấp vào tab activity\_main.xml để xem trình chỉnh sửa bố cục.





1. Nhấp vào tab Design để hiển thị biểu diễn đồ họa của bố cục như hình dưới đây:



1. Nhấn tab MainActivity.Java để xem trình viết mã :





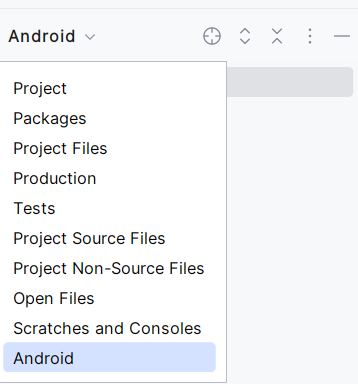
##### 2.2 Khám phá dự án -> Androi pane

Trong phần này , bạn sẽ khám phá ra cách dự án được tổ chức trong **Android Studio.**

1. Nhấn vào **Project** ở bên trái màn hình .

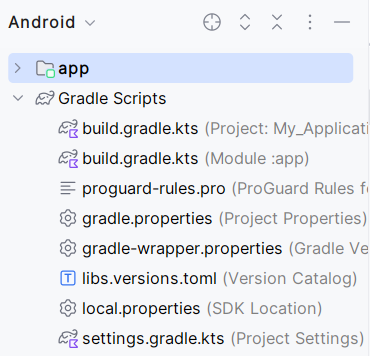


1. Nhấn **Android**



###### Thư mục Gradle Scripts

Hệ thống build Gradle trong Android Studio giúp bạn dễ dàng thêm các tệp nhị phân bên ngoài hoặc các module thư viện khác vào dự án dưới dạng dependency.



Làm theo các bước để khám phá hệ thống Gradle :

1. **build.gradle.kts (** Project: Hello\_World)

Đây là nơi bạn sẽ tìm thấy những lựa chọn cấu hình chung cho tất cả mô đun tạo lên dự án của bạn . Mọi dự án **Android Studio** đều chứa một tệp build.gradle cấp cao nhất . Hầu như ta không cần thay đổi nội dung tệp này nhưng việc hiểu nó vẫn là cần thiết.

Mặc định thì tệp build cấp cao nhất sử dụng khối buildscripts để xác định kho lưu trữ Gradle và các thành phần phụ thuộc chung cho tất cả mô đun của dự án . Khi phần phụ thuộc của bạn không phải là thư viện cục bộ hoặc cây tệp, Gradle sẽ tìm kiếm các tệp trong bất kỳ kho lưu trữ trực tuyến nào được chỉ định trong khối kho lưu trữ của tệp này. Mặc định, các dự án Android Studio mới khai báo JCenter và Google là vị trí kho lưu trữ.

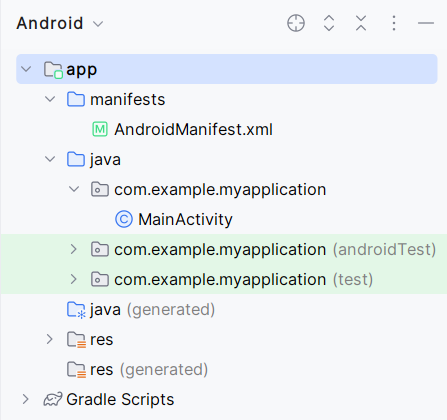
1. **build.gradle.kts (**Module: app)

Ngoài file build.gradle.kts() cấp cao nhất thì mỗi mô đun đều có một file build.gradle của riêng nó, cho phép bạn cấu hình từng mô đun cụ thể. Việc cấu hình các tùy chọn cài đặt này cho phép bạn cung cấp các gói tùy chỉnh. Bạn cũng có thể ghi đè các cài đặt trong tệp AndroidManifest.xml hoặc file build.gradle.kts cấp cao nhất.

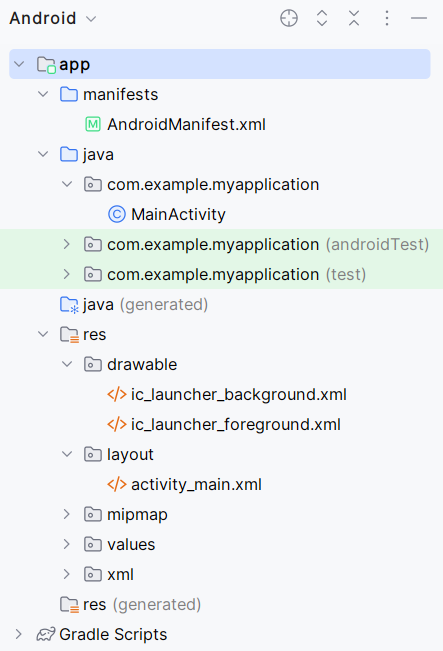
Tệp này thường là tệp cần chỉnh sửa khi thay đổi cấu hình cấp ứng dụng , chẳng hạn như khai báo các phần phụ thuộc trong phần phụ thuộc. Bạn có thể khai báo phần phụ thuộc thư viện bằng cách sử dụng một trong một số cấu hình phần phụ thuộc khác nhau . Mỗi cấu hình phần phụ thuộc cung cấp cho Gradle các hướng dẫn khác nhau về cách sử dụng thư viện.

###### Thư mục app và res.

Tất cả code và tài nguyên đều nằm trong thư mục app và res.



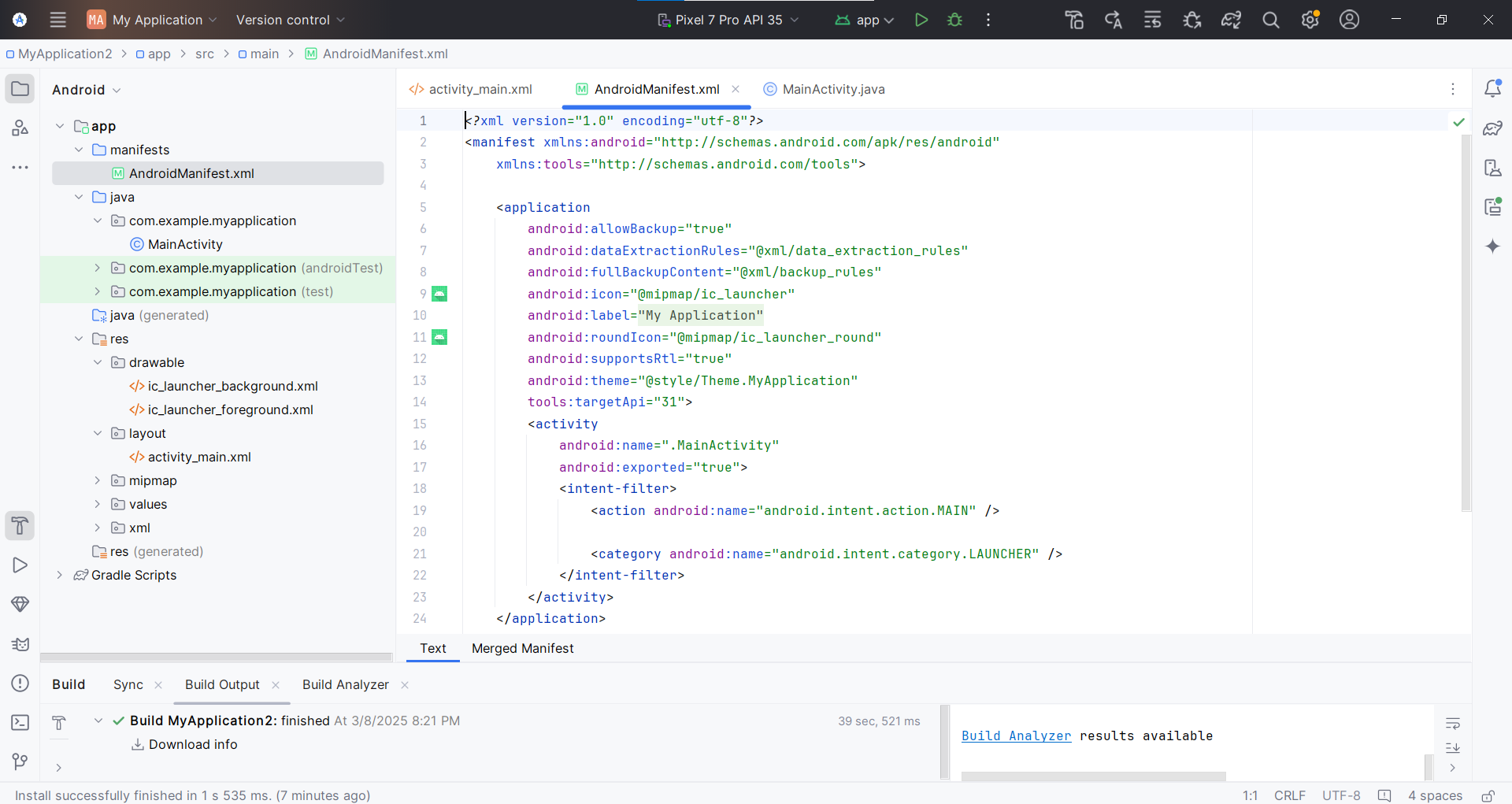
Thư mục java chứa các tệp lớp java trong 3 thư mục con .Thư mục **com.example.myapplication** chứa tất cả các file cho gói ứng dụng. 2 thư mục còn lại được dùng trong việc kiểm thử và mô tả.



Thư mục res lưu trữ tài nguyên như bố cục , hình ảnh … .1 activity thường được liên kết với 1 một layout giao diện người dùng được định nghĩa dưới dạng xml. Tên layout và tên Activity thường giống nhau.

###### Thư mục manifest

Thư mục này chứa các file cung cấp thông tin cần thiết về ứng dụng của bạn cho hệ thống Android hệ thống phải có thông tin này trước khi có thể chạy bất kỳ mã nào của ứng dụng.



Thư mục AndroidManifest.xml mô tả tất cả các thành phần của ứng dụng của bạn.

Tất cả các thành phần , ví dụ MainActivity , phải được khai báo trong tệp xml này.

#### Nhiệm vụ 3 : sử dụng máy ảo.

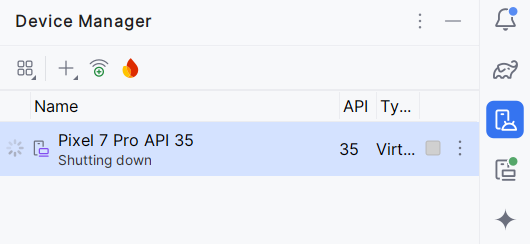
Trong nhiệm vụ này , bạn sẽ sử dụng trình quản lý máy ảo Android (AVD) để tạo máy ảo.Điều này giúp mô phỏng cấu hình cho một loại thiết bị Android cụ thể và sử dụng thiết bị ảo đó để chạy ứng dụng.

Khi sử dụng Trình quản lý AVD, bạn xác định các đặc điểm phần cứng của thiết bị, cấp độ API, bộ nhớ, giao diện và các thuộc tính khác và lưu thiết bị đó dưới dạng thiết bị ảo. Với thiết bị ảo, bạn có thể kiểm tra ứng dụng trên các cấu hình thiết bị khác nhau (chẳng hạn như máy tính bảng và điện thoại) với các cấp độ API khác nhau mà không cần phải sử dụng thiết bị vật lý

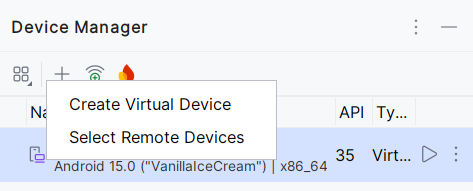
##### 3.1 Tạo máy ảo Android

Để chạy trình mô phỏng trên máy tính, bạn phải tạo một cấu hình mô tả máy ảo.

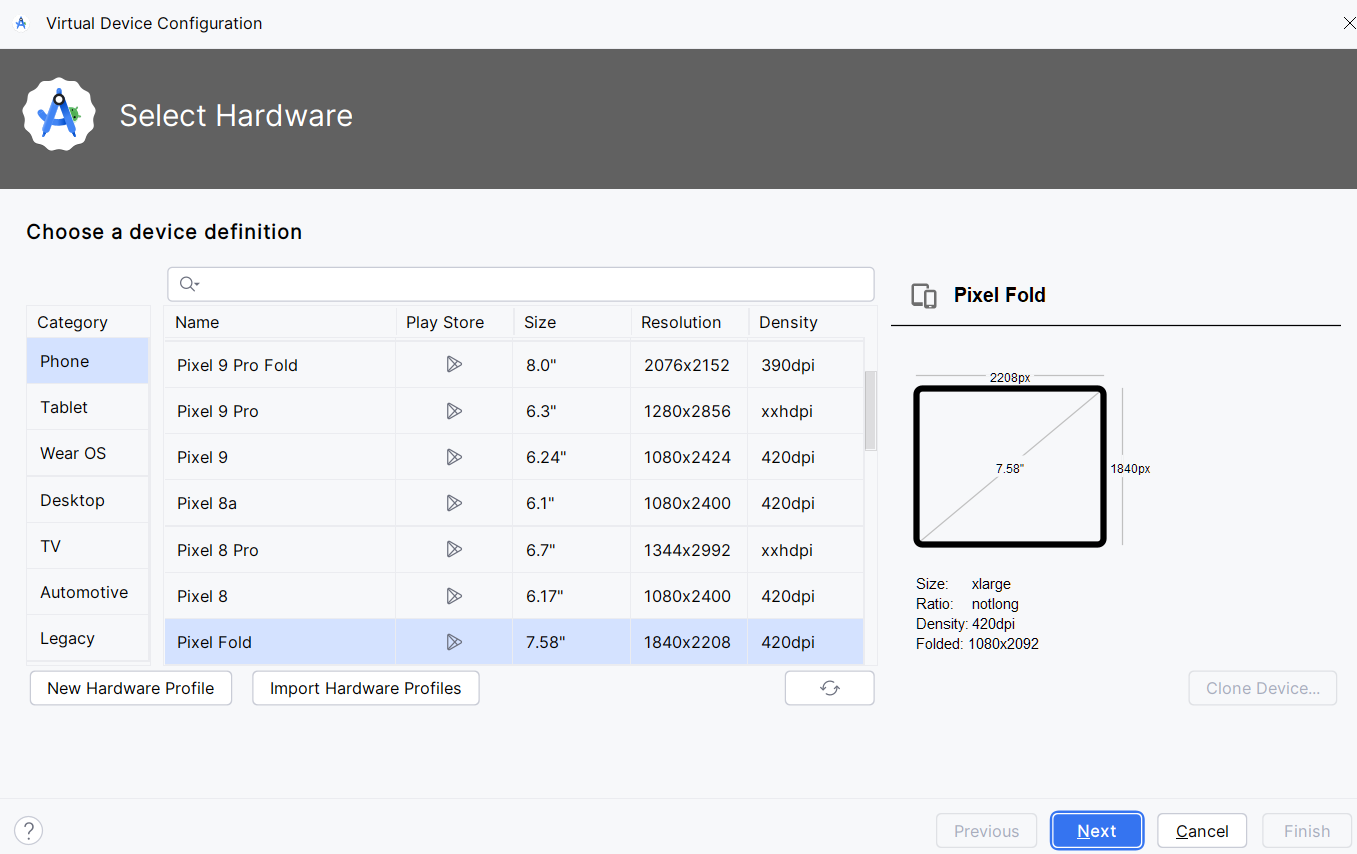
1. Mở Android Studio , chọn **tool > Android > AVD manager** . Màn hình máy ảo của bạn sẽ xuất hiện . Nếu bạn đã tạo máy ảo , màn hình sẽ hiển thị nó ,nếu không bạn sẽ thấy 1 danh sách trống.



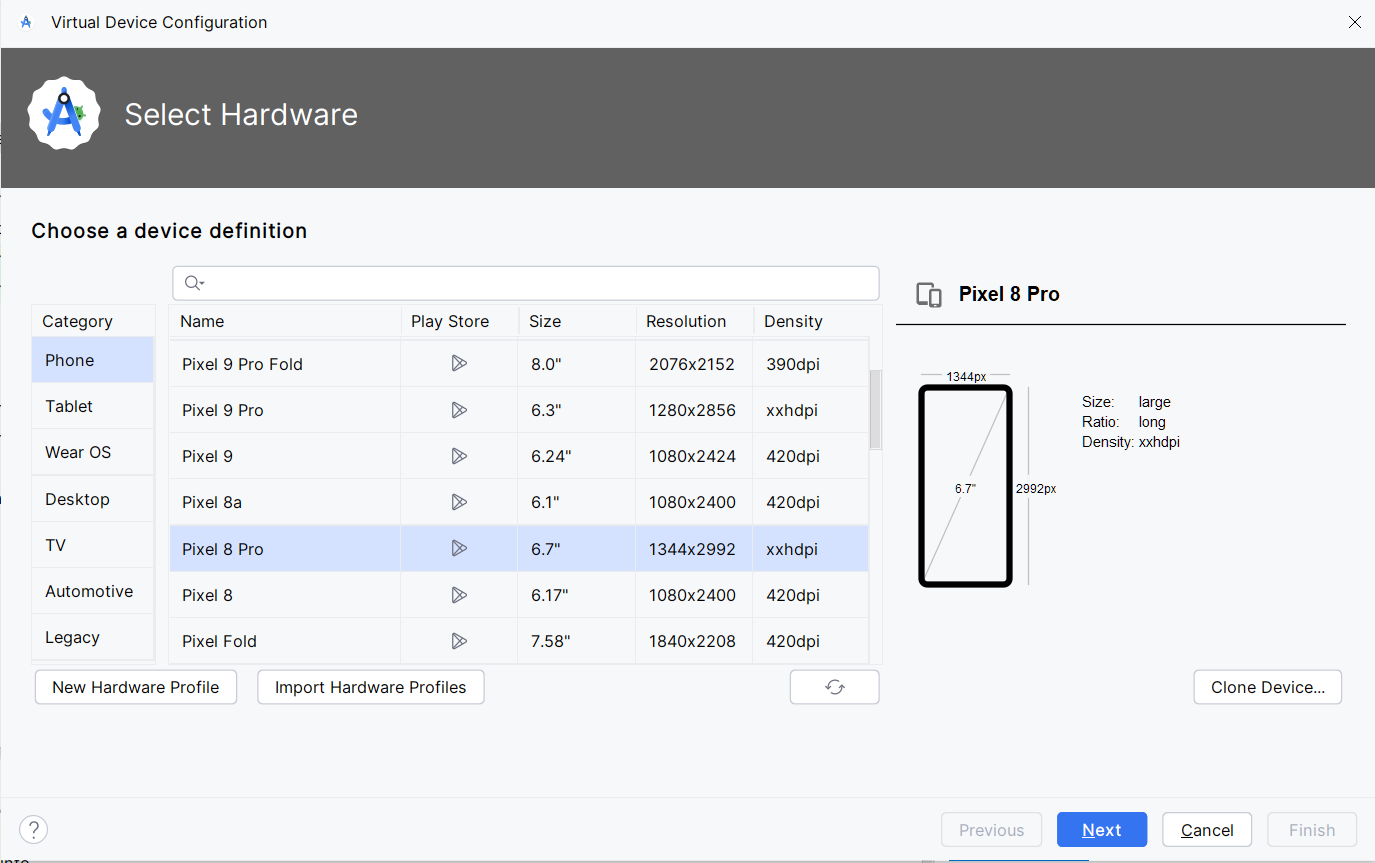
1. Nhấn **+ > create virtual Device**



1. Cửa số  **Select Hardware** sẽ hiển thị danh sách các thiết bị phần cứng đã được cấu hình sẵn . Với mỗi thiết bị , danh sách này sẽ cung cấp các cột thông tin như **kích thước , độ phân giải , mật độ điểm.**



1. Chọn thiết bị bất kì , ví dụ pixel 6 pro . Nhấn **Next.**

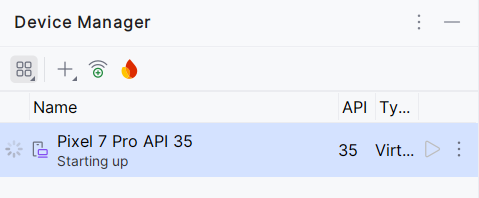


1. Chọn phiên bản và chờ tải xuống . **Finish -> Next** ;

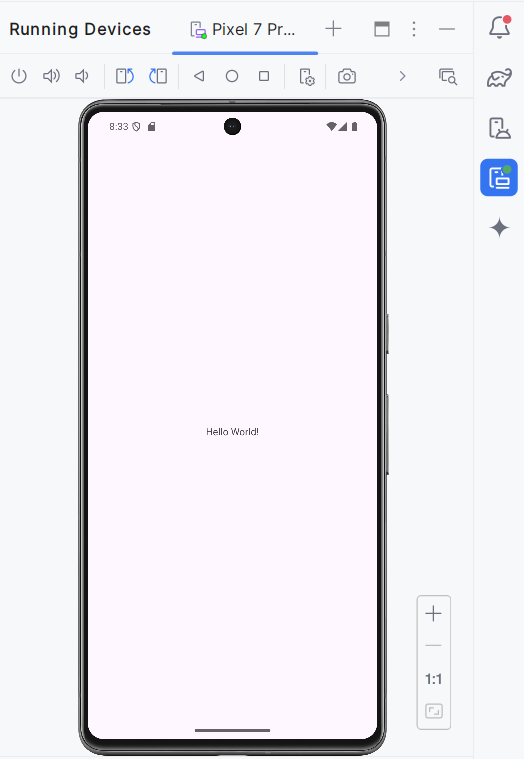
##### Chạy ứng dụng trên máy ảo.

Chạy ứng dụng Hello World thôi.

1. Nhấn icon **Run > Run app**
2. Chọn máy ảo -> nhấn **OK.**



1. Kết quả :



#### Nhiệm vụ 4 : thiết bị vật lý ( bỏ qua).

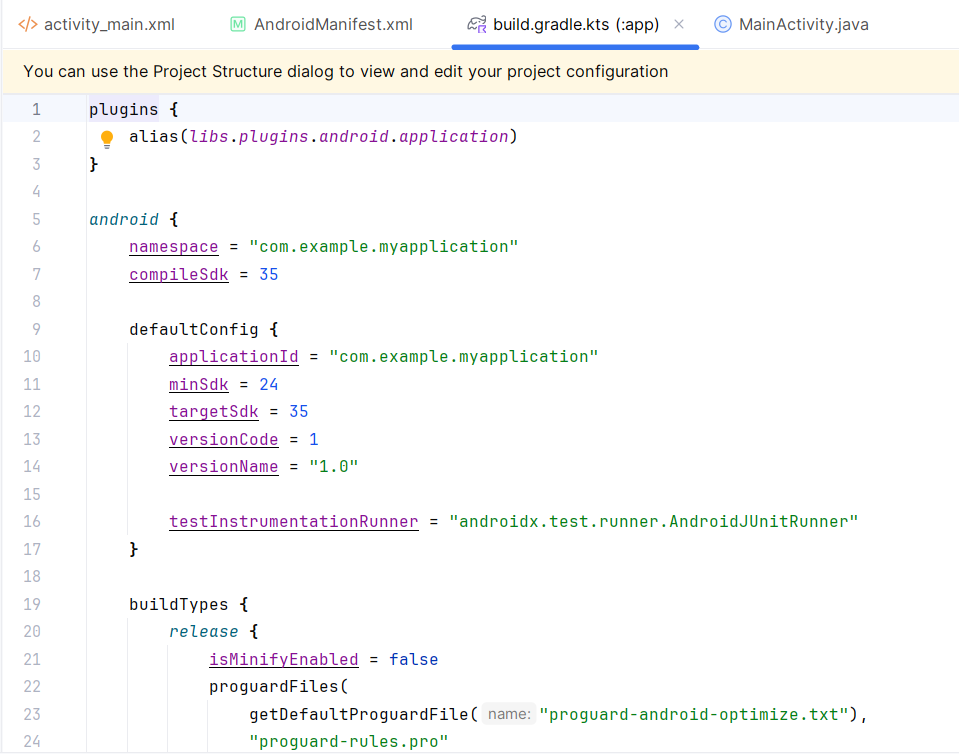
#### Nhiệm vụ 5: Cấu hình Gradle

Ở nhiệm vụ này , bạn sẽ thay đổi một vài thứ về cấu hình ứng dụng bên trong tệp build.gradle.kts(Module:app) để tìm hiểu cách thực hiện các thay đổi và đồng bộ hóa chúng với dự án Android Studio của bạn.

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

Làm theo các bước:

1. Mở tệp tin **build.gradle.kts(Module:app)** trong thư mục **Gradle scripts**



1. Thay đổi giá trị minSdk

##### 5.2 Đồng bộ hóa cấu hình Gradle mới.

#### Nhiệm vụ 6: LogCat

##### 6.1 Quan sát LogCat pane

6.2 Thêm LogCat

**Coding challenge**

### Giao diện người dùng tương tác đầu tiên

**Giới thiệu**

Giao diện người dùng hiển thị trên thiết bị Android bao gồm một hệ thống phân cấp các đối tượng gọi là **Views -**  mỗi phần tử trên màn hình được gọi là **View.** Lớp **View**  đại diện cho khối xây dựng cơ bản cho tất cả các thành phần giao diện người dùng , và lớp cơ sở cho các lớp cung cấp các thành phần giao diện người dùng tương tác như nút, hộp kiểm và trường nhập văn bản.

* **TextView** : Hiển thị văn bản.
* **EditText :** Cho phép người dùng nhập và chỉnh sửa văn bản.
* **Button và các phần tử để nhấn khác ( RadioButton , CheckBox , Spinner)** : Cung cấp hành vi tương tác.
* **ScollView & RecyclerView** : Hiển thị danh sách cuộn.
* **ImageView :** Hiển thị hình ảnh.
* **ConstraintLayout & LinearLayout :** Để chứa các **View**  và định vị chúng.

Mã Java hiển thị và điều khiển giao diện người dùng được chứa trong lớp mở rộng của lớp **Activity** . Một **Activity**  thường được liên kết với một **Layout**  của chế độ xem giao diện định nghĩa dưới dạng tệp **xml**. Tệp **xml** thường được đặt tên theo **Activity**  của nó và định nghĩa bố cục của các phần tử trên màn hình.

Ví dụ , mã **Activity** trong ứng dụng **Hello World** hiển thị bố cục được định nghĩa trong tệp bố cục **activity\_main.xml .** Tệp này chứa **TextView** với văn bản “Hello World”.

Trong các ứng dụng phức tạp hơn , **Activity** có thể triển khai các hoạt động để phản hồi các thao tác nhấn của người dùng, vẽ nội dung đồ họa , yêu cầu dữ liệu từ cơ sở dữ liệu hoặc internet .

Trong bài học này , bạn sẽ tìm hiểu các tạo ứng dụng tương tác – ứng dụng cho phép tương tác với người dùng. Bạn tạo ứng dụng sử dụng mẫu **Empty Activity**.

Bạn cũng học cách sử dụng trình xoạn thảo bố cục để thiết kế một bố cục và cách chỉnh sửa bố cục trong **XML .**

**Những gì bạn nên biết :**

* Cài đặt và mở **Android Studio** .
* Tạo ứng dụng HelloWorld.
* Chạy ứng dụng HelloWorld.

**Những gì bạn sẽ học :**

* Tạo ứng dụng có khả năng tương tác.
* Sử dụng trình chỉnh sửa bố cục để thiết kế bố cục.
* Chỉnh sửa bố cục trong tệp XML.

**Những gì bạn sẽ làm :**

* Tạo ứng dụng và thêm hai **Button** và một **TextView** vào bố cục.
* Thao tác từng phần tử trong ConstraintLayout để hạn chế chúng ở lề và các phần tử khác.
* Thay đổi thuộc tính thành phần giao diện người dùng.
* Thay đổi bố cục của ứng dụng trong XML.
* Trích xuất các chuỗi được mã hóa cứng vào tài nguyên chuỗi.
* Triển khai các phương thức xử lý nhấp chuột để hiển thị thông báo trên màn hình khi người dùng nhấn vào từng nút

**Tổng quan**

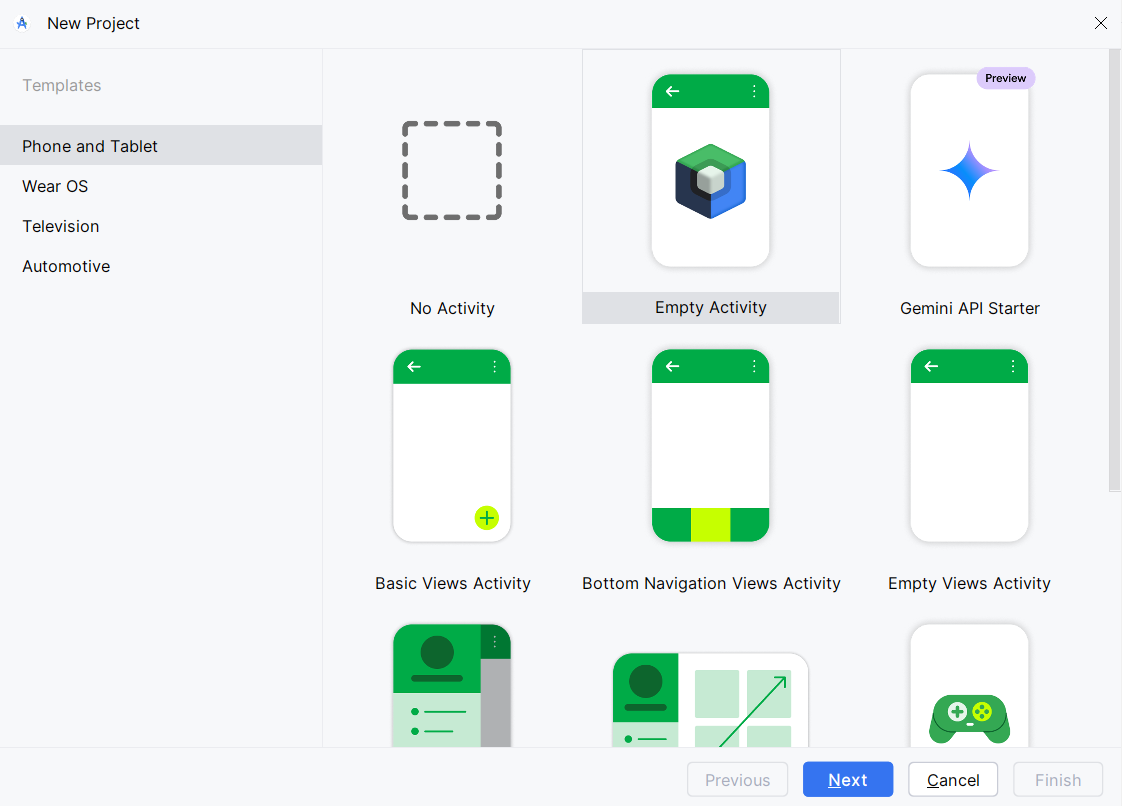
Ứng dụng HelloToast bao gồm hai **Button** và một **TextView**. Khi người dùng nhấn vào

Button thứ nhất thì màn hình sẽ hiển thị một thông báo ngắn ( gọi là Toast).Khi nhấn vào Button thứ hai thì tăng số đếm của TextView ( bắt đầu từ 0).

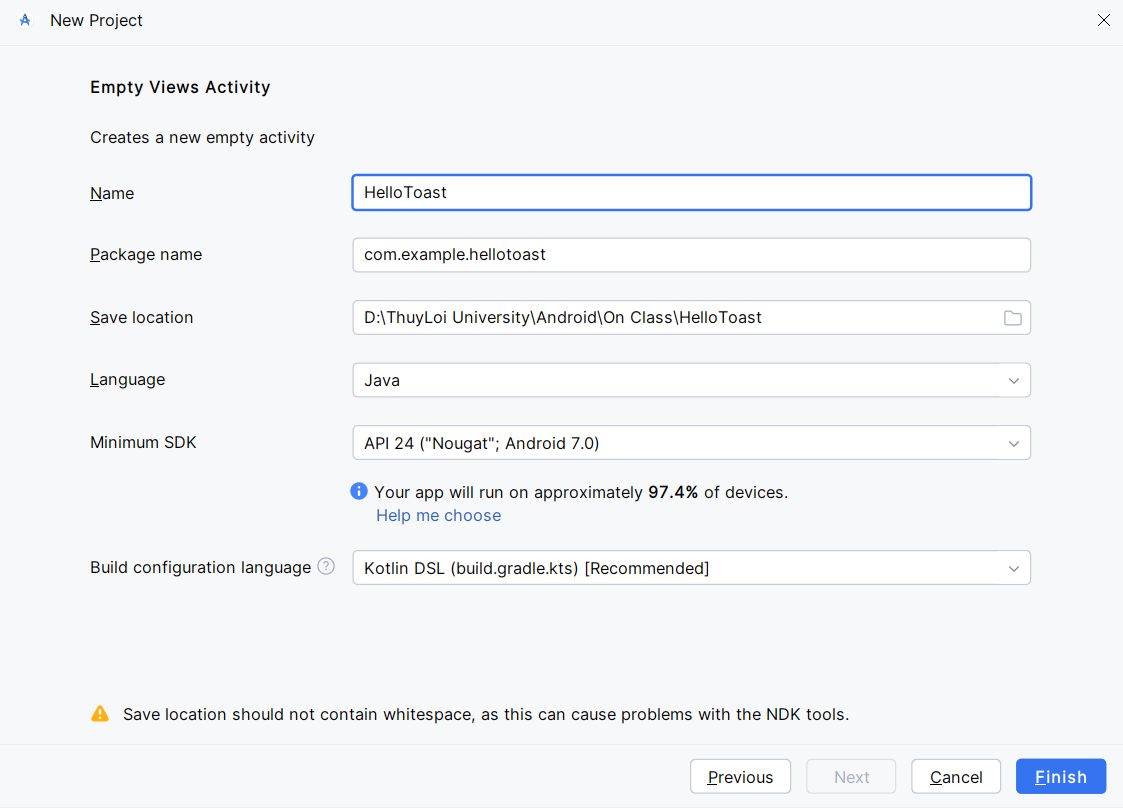
#### Nhiệm vụ 1 : Tạo và khám phá ứng dụng.

##### 1.1 Tạo ứng dụng Android.

Khởi động Android Studio và tạo một dự án mới với các thông số sau:



Chọn **Empty views Activity > Next.**

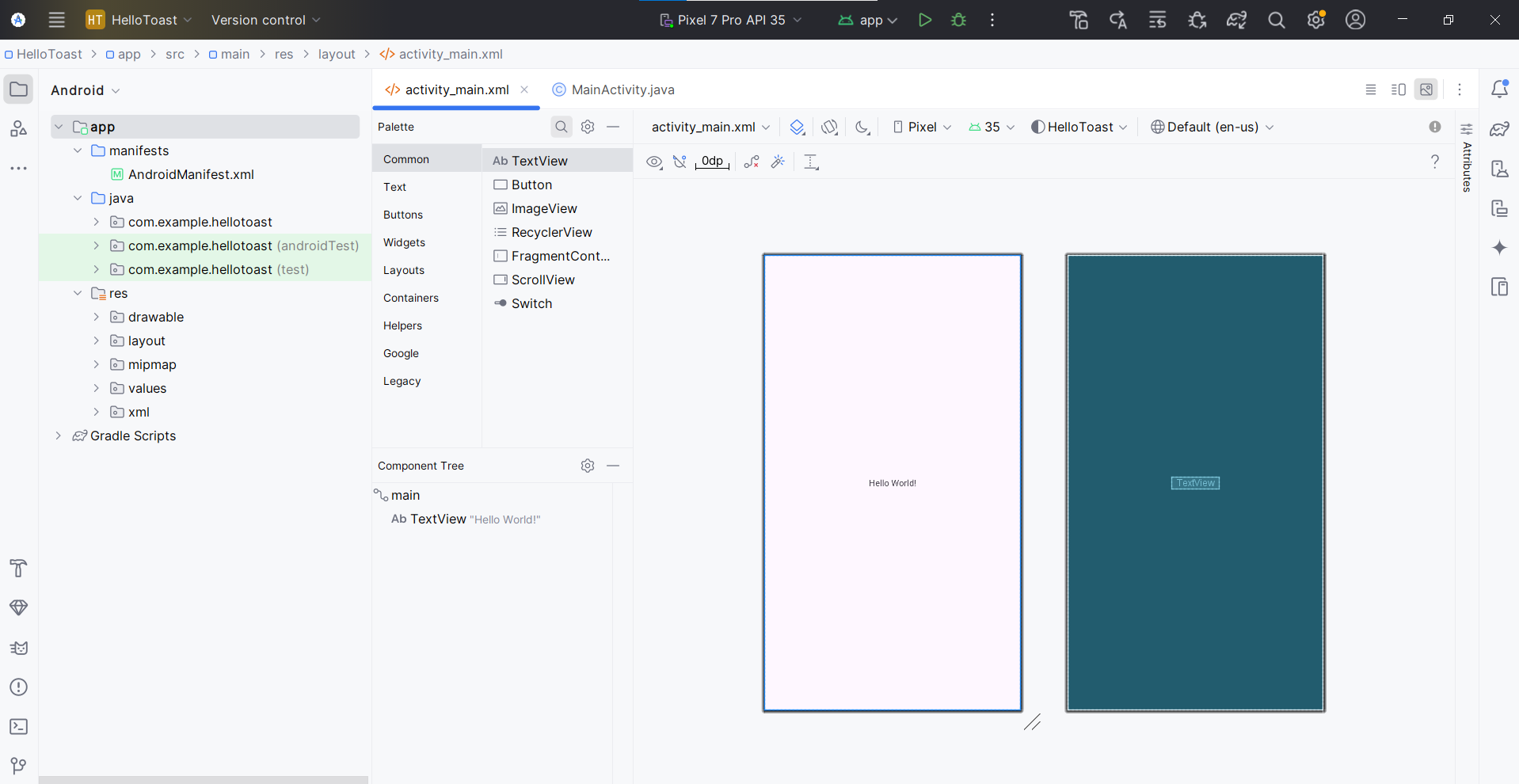


Chọn **Finsish** để biên dịch và chạy ứng dụng trên trình giả lập hoặc thiết bị của bạn.

##### 1.2 Khám phá trình chỉnh sửa bố cục.

Android Studio cung cấp **Layout Editor** để nhanh chóng xây dựng giao diện người dùng (UI) cho ứng dụng. Công cụ này cho phép bạn kéo thả các thành phần vào chế độ thiết kế trực quan hoặc bản vẽ, sắp xếp chúng trong bố cục, thêm ràng buộc và thiết lập thuộc tính. **Ràng buộc (Constraints)** xác định vị trí của một phần tử UI trong bố cục. Một ràng buộc thể hiện sự liên kết hoặc kết nối với một thành phần khác, bố cục cha hoặc một đường hướng dẫn ẩn.

Khám phá **Layout Editor** và tham khảo hình dưới đây khi bạn thực hiện theo các bước được đánh số:



1.  Trong bảng điều hướng **Project > Android**, mở thư mục **app > res > layout**, sau đó nhấp đúp vào tệp **activity\_main.xml** để mở, nếu nó chưa được mở.
2.  Nhấp vào tab **Design** nếu chưa được chọn. Tab **Design** cho phép bạn chỉnh sửa trực quan các phần tử và bố cục, trong khi tab **Text** dùng để chỉnh sửa mã XML của bố cục.
3.  **Pane Palettes** hiển thị các thành phần UI mà bạn có thể sử dụng trong bố cục ứng dụng.
4.  **Pane Component Tree** hiển thị cấu trúc phân cấp của các phần tử UI. Các phần tử **View** được tổ chức theo dạng cây, trong đó một phần tử con kế thừa thuộc tính từ phần tử cha. Trong hình trên, **TextView** là phần tử con của **ConstraintLayout**. Bạn sẽ tìm hiểu thêm về các thành phần này trong bài học.
5.  **Pane thiết kế và bản vẽ kỹ thuật (blueprint)** của **Layout Editor** hiển thị các phần tử UI trong bố cục. Trong hình minh họa, bố cục chỉ có một phần tử duy nhất: **TextView** hiển thị dòng chữ "Hello World".
6.  **Tab Attributes** hiển thị **Pane Attributes**, nơi bạn có thể thiết lập các thuộc tính cho phần tử UI.

#### Nhiệm vụ 2 : Thêm các phần tử vào bố cục

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

##### Kiểm tra ràng buộc của phần tử

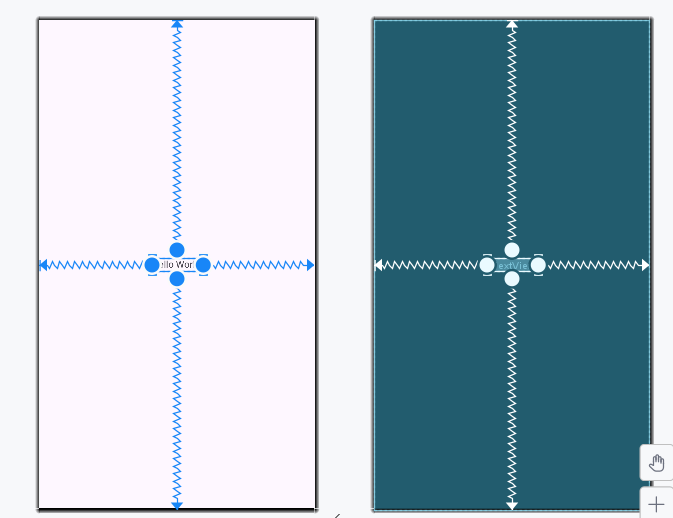
Thực hiện các bước sau:

1. Mở **activity\_main.xml** từ bảng **Project > Android** nếu nó chưa được mở. Nếu tab **Design** chưa được chọn, hãy nhấp vào nó.

* Nếu không thấy chế độ **Blueprint**, nhấn vào nút **Select Design Surface** trên thanh công cụ và chọn **Design + Blueprint**.

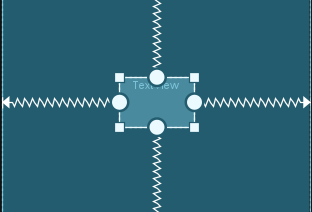
1. **Công cụ Autoconnect** nằm trên thanh công cụ và được bật theo mặc định. Hãy đảm bảo công cụ này không bị tắt.
2. Nhấp vào nút **Zoom in** để phóng to bảng thiết kế và blueprint để có cái nhìn chi tiết hơn.
3. Trong bảng **Component Tree**, chọn **TextView**. Phần tử **"Hello World" TextView** sẽ được tô sáng trong bảng thiết kế và blueprint, đồng thời các ràng buộc của nó sẽ hiển thị.
4. Làm theo hướng dẫn trong hình minh họa:

* Nhấp vào chấm tròn bên phải của **TextView** để xóa ràng buộc ngang (constraint) giữa nó và cạnh phải của bố cục. Khi đó, **TextView** sẽ nhảy sang bên trái vì không còn bị ràng buộc với cạnh phải.
* Để thêm lại ràng buộc ngang, nhấp vào cùng chấm tròn đó và kéo một đường thẳng đến cạnh phải của bố cục.



Trong bảng **Blueprint** hoặc **Design**, phần tử **TextView** sẽ hiển thị các tay cầm (**handles**) sau:

* **Constraint handle**: Dùng để tạo ràng buộc (**constraint**). Nhấp vào tay cầm ràng buộc (hình tròn ở cạnh phần tử), sau đó kéo đến một tay cầm khác hoặc đến mép của phần tử cha. Ràng buộc sẽ được hiển thị bằng một đường zigzag.
* **Resizing handle**: Dùng để thay đổi kích thước phần tử. Kéo các tay cầm vuông ở góc để thay đổi kích thước. Khi kéo, tay cầm sẽ chuyển thành góc xiên.

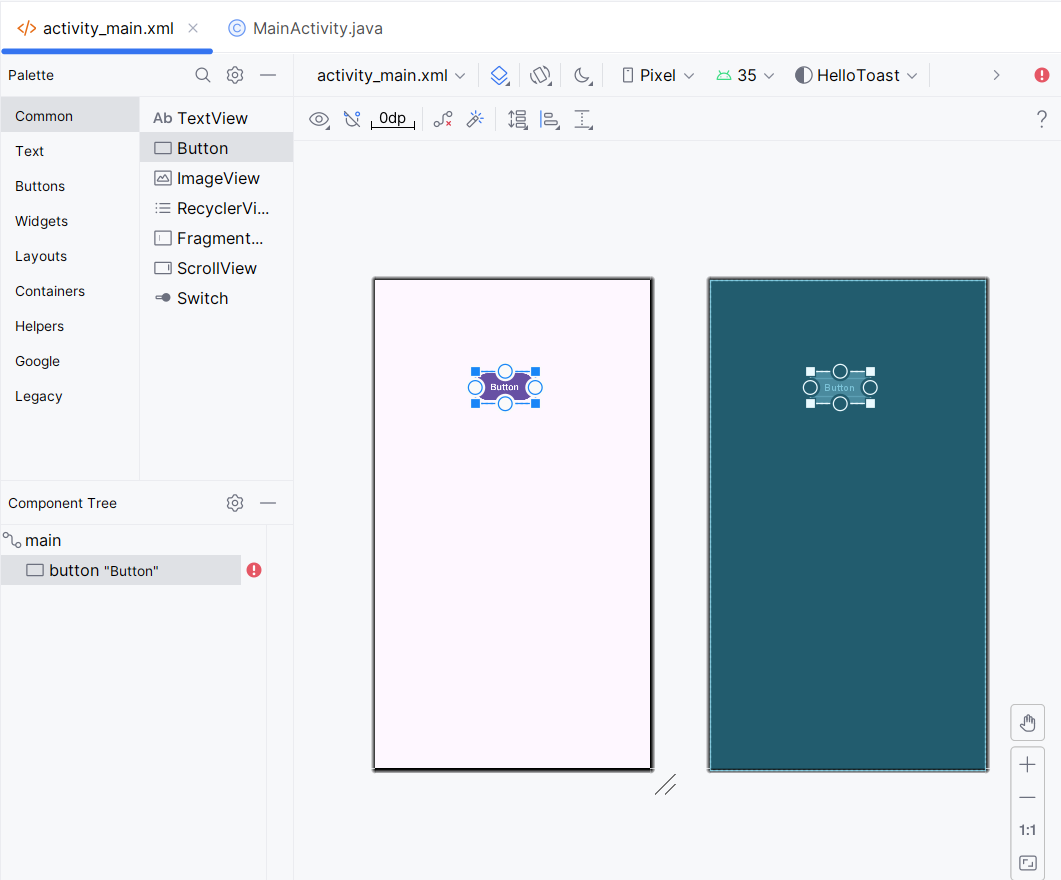


##### Thêm một Button vào bố cục

Khi được bật, công cụ **Autoconnect** tự động tạo hai hoặc nhiều ràng buộc cho một phần tử UI với bố cục cha. Sau khi kéo phần tử vào bố cục, nó sẽ tạo ràng buộc dựa trên vị trí của phần tử.

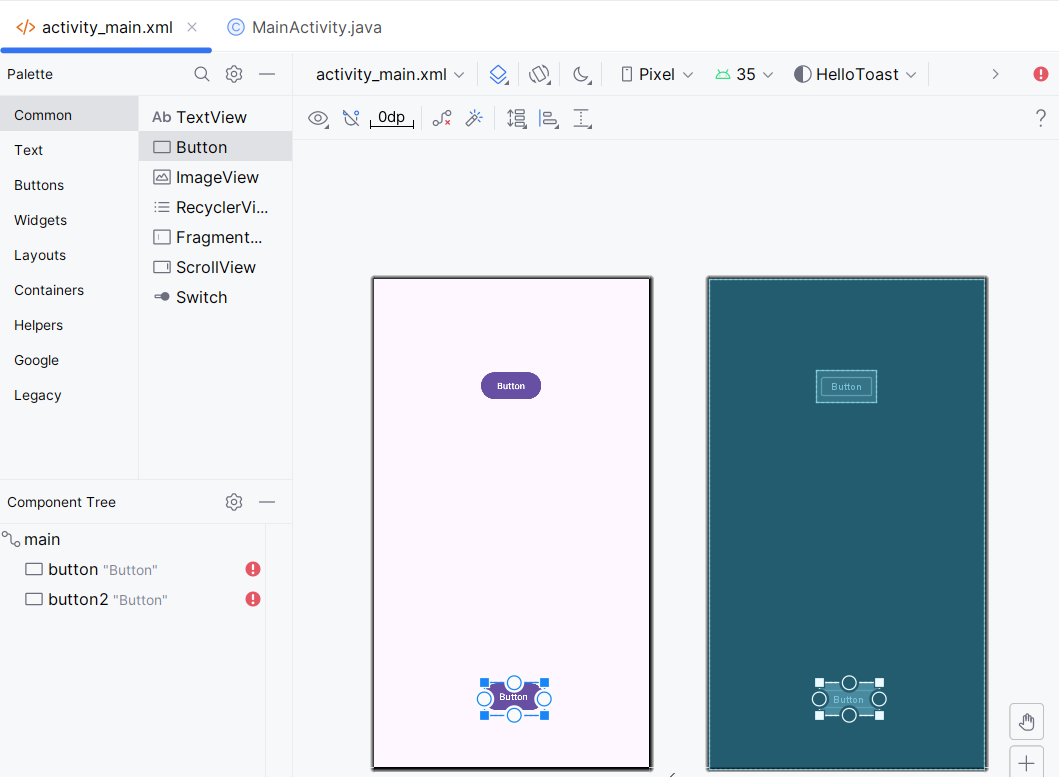
Thực hiện các bước sau để thêm một **Button**:

1. **Xóa TextView** để có bố cục trống: Khi **TextView** vẫn được chọn, nhấn phím **Delete** hoặc chọn **Edit > Delete**.
2. **Thêm Button**:
   * Kéo một **Button** từ bảng **Palette** vào bất kỳ vị trí nào trong bố cục.
   * Nếu thả **Button** vào khu vực giữa phía trên của bố cục, các ràng buộc có thể tự động xuất hiện.
   * Nếu không, kéo các ràng buộc đến cạnh trên, trái và phải của bố cục như minh họa trong hình động dưới đây.



##### Thêm Button thứ hai vào bố cục.

1.  Kéo một **Button** khác từ bảng **Palette** vào giữa bố cục như trong hình minh họa bên dưới. **Autoconnect** có thể tự động tạo ràng buộc ngang (nếu không, bạn có thể kéo ràng buộc thủ công).
2.  Kéo một ràng buộc dọc xuống cạnh dưới của bố cục (tham khảo hình minh họa bên dưới).



Bạn có thể xóa ràng buộc của một phần tử bằng cách chọn phần tử đó và di chuột qua để hiển thị nút **Clear Constraints**. Nhấp vào nút này để xóa **tất cả** ràng buộc của phần tử đã chọn.

Để xóa một ràng buộc cụ thể, nhấp vào tay cầm thiết lập ràng buộc đó.

Để xóa tất cả ràng buộc trong bố cục, nhấp vào công cụ **Clear All Constraints** trên thanh công cụ. Công cụ này hữu ích khi bạn muốn thiết lập lại toàn bộ ràng buộc trong bố cục.

#### Nhiệm vụ 3 :Thay đổi thuộc tính phần tử giao diện.

**Pane Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI.

Bạn có thể tìm thấy các thuộc tính (được gọi là **properties**) chung cho tất cả các **View** trong tài liệu của lớp **View**.

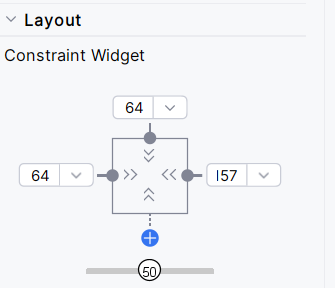
Trong nhiệm vụ này, bạn sẽ nhập giá trị mới và thay đổi các giá trị cho các thuộc tính quan trọng của **Button**, những thuộc tính này cũng áp dụng cho hầu hết các loại **View**.

##### 3.1 Thay đổi kích thước Button

Trình chỉnh sửa bố cục cung cấp các tay cầm thay đổi kích thước ở cả bốn góc của một **View**, giúp bạn có thể điều chỉnh kích thước nhanh chóng.

Bạn có thể kéo các tay cầm ở mỗi góc để thay đổi kích thước **View**, nhưng làm như vậy sẽ đặt cố định (**hardcode**) chiều rộng và chiều cao. Tránh đặt kích thước cố định cho hầu hết các phần tử **View**, vì điều này khiến chúng không thể thích ứng với nội dung và các kích thước màn hình khác nhau.

Thay vào đó, hãy sử dụng **Attributes pane** ở bên phải trình chỉnh sửa bố cục để chọn chế độ kích thước không sử dụng giá trị cố định. **Attributes pane** bao gồm một bảng điều chỉnh kích thước hình vuông (**view inspector**) ở phía trên. Các biểu tượng bên trong hình vuông đại diện cho cài đặt chiều cao và chiều rộng như sau:



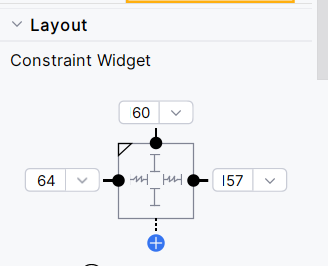
Trong hình minh họa trên:

1. **Điều khiển chiều cao**: Xác định thuộc tính layout\_height và xuất hiện dưới dạng hai đoạn ở phía trên và dưới của hình vuông. Các góc nghiêng cho thấy chế độ được đặt là wrap\_content, nghĩa là **View** sẽ mở rộng theo chiều dọc để phù hợp với nội dung. Số "64" biểu thị khoảng lề tiêu chuẩn là **64dp**.
2. **Điều khiển chiều rộng**: Xác định thuộc tính layout\_width và xuất hiện dưới dạng hai đoạn ở bên trái và bên phải của hình vuông. Các góc nghiêng cho thấy chế độ được đặt là wrap\_content, nghĩa là **View** sẽ mở rộng theo chiều ngang để phù hợp với nội dung, với lề tối đa **64dp**.
3. **Nút đóng bảng Attributes**: Nhấp vào để đóng bảng thuộc tính.

Thực hiện các bước sau:

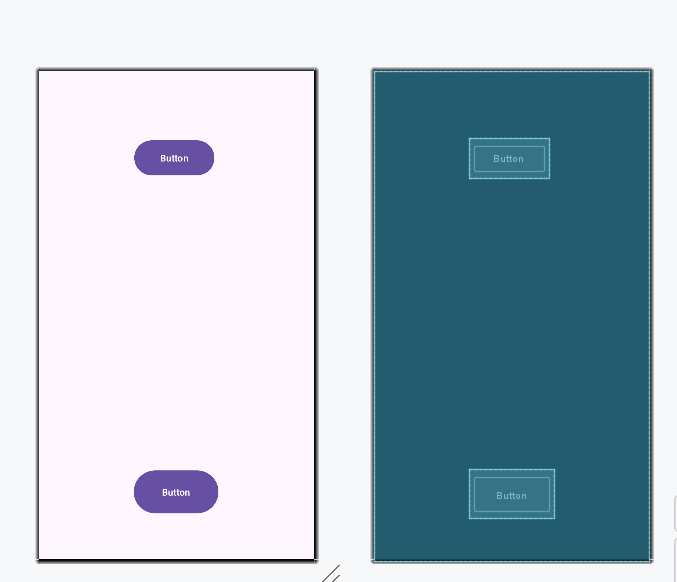
1. Chọn **Button** phía trên trong bảng **Component Tree**.
2. Nhấp vào tab **Attributes** ở bên phải cửa sổ trình chỉnh sửa bố cục.
3. Nhấp vào **điều khiển chiều rộng** hai lần:

* Lần nhấp đầu tiên thay đổi thành **Fixed** với các đường thẳng.
* Lần nhấp thứ hai thay đổi thành **Match Constraints** với biểu tượng lò xo, như minh họa trong hình động bên dưới.



Khi thay đổi **điều khiển chiều rộng**, thuộc tính layout\_width trong bảng **Attributes** sẽ hiển thị giá trị **match\_constraint**, và phần tử **Button** sẽ kéo dãn theo chiều ngang để lấp đầy khoảng trống giữa cạnh trái và cạnh phải của bố cục.

1. Chọn **Button** thứ hai và thực hiện các thay đổi tương tự cho **layout\_width** như ở bước trước, như minh họa trong hình bên dưới.



Như đã thấy ở các bước trước, các thuộc tính **layout\_width** và **layout\_height** trong bảng **Attributes** sẽ thay đổi khi bạn điều chỉnh các điều khiển chiều cao và chiều rộng trong **inspector**. Các thuộc tính này có thể nhận một trong ba giá trị trong **ConstraintLayout**:

* **match\_constraint**: Mở rộng phần tử **View** để lấp đầy không gian của phần tử cha theo chiều rộng hoặc chiều cao—tính đến lề nếu có. Phần tử cha trong trường hợp này là **ConstraintLayout**. Bạn sẽ tìm hiểu thêm về **ConstraintLayout** trong nhiệm vụ tiếp theo.
* **wrap\_content**: Thu nhỏ phần tử **View** sao cho vừa đủ để chứa nội dung bên trong. Nếu không có nội dung, **View** sẽ trở nên vô hình.
* **Fixed size (kích thước cố định)**: Để đặt kích thước cố định nhưng vẫn thích ứng với màn hình thiết bị, sử dụng đơn vị **dp** (density-independent pixels). Ví dụ, **16dp** có nghĩa là 16 pixel độc lập với mật độ màn hình.

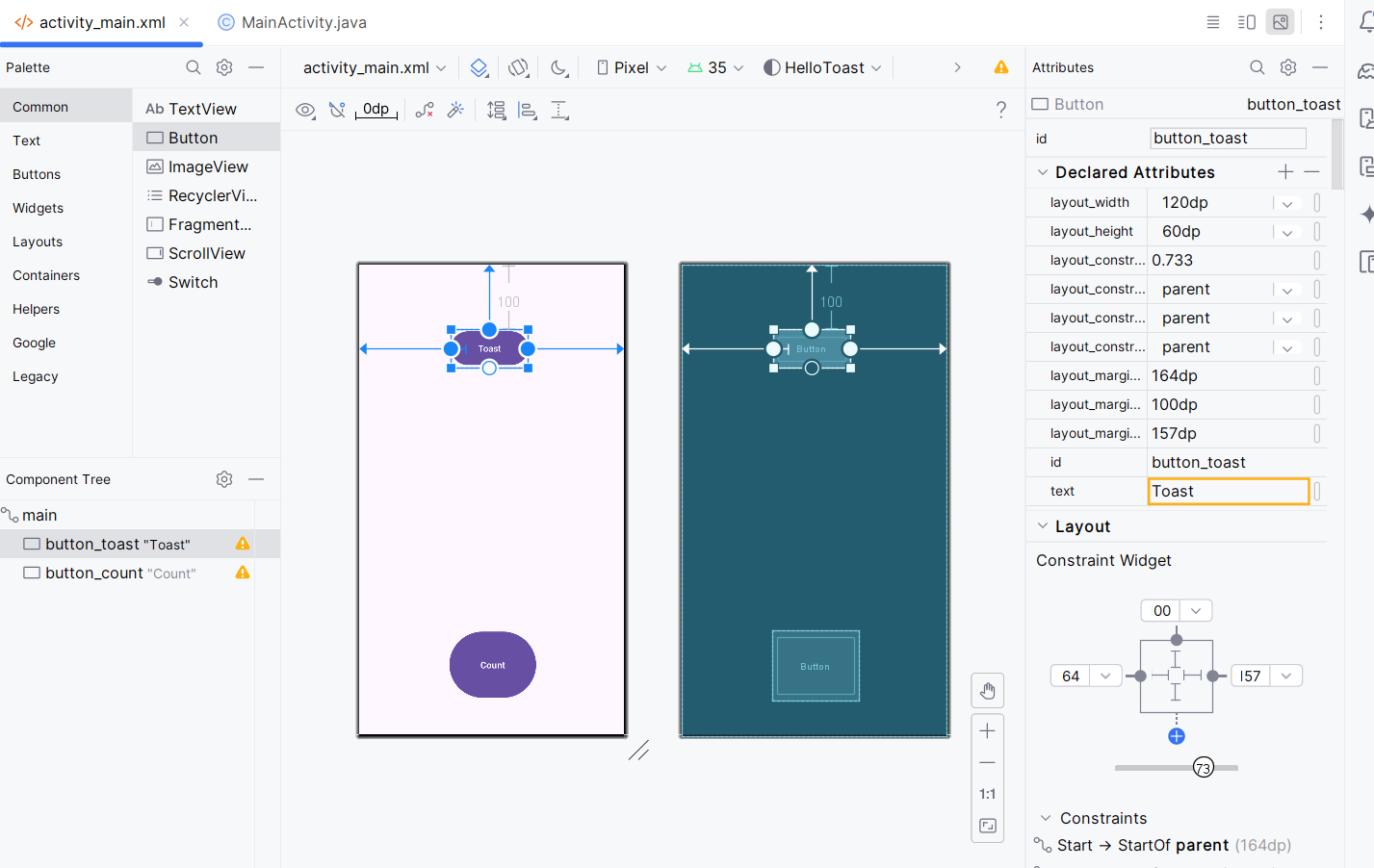
##### Thay đổi thuộc tính Button.

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

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

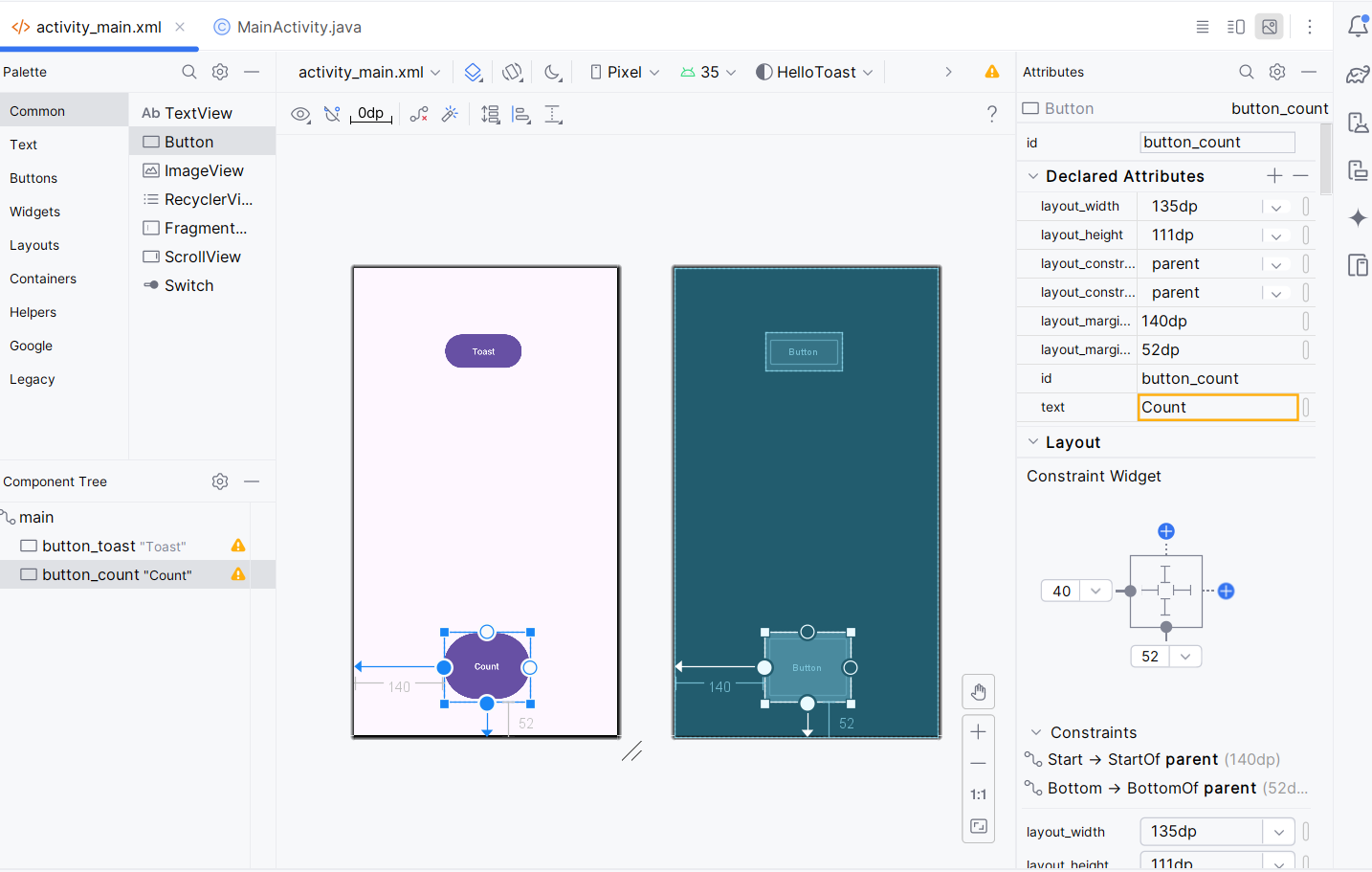
Thực hiện các bước sau:

1. Sau khi chọn **Button** đầu tiên, chỉnh sửa trường **ID** ở đầu bảng **Attributes** thành **button\_toast** cho thuộc tính **android:id**, dùng để xác định phần tử trong bố cục.
2. Đặt thuộc tính **background** thành **@color/colorPrimary**. (Khi nhập **@c**, các tùy chọn sẽ xuất hiện để dễ chọn.)
3. Đặt thuộc tính **textColor** thành **@android:color/white**.
4. Chỉnh sửa thuộc tính **text** thành **Toast**



1. Thực hiện các thay đổi thuộc tính tương tự cho **Button** thứ hai:

* Đặt **ID** thành **button\_count** cho thuộc tính **android:id**.
* Đặt thuộc tính **text** thành **Count**.
* Đặt thuộc tính **background** thành **@color/colorPrimary**.
* Đặt thuộc tính **textColor** thành **@android:color/white**.



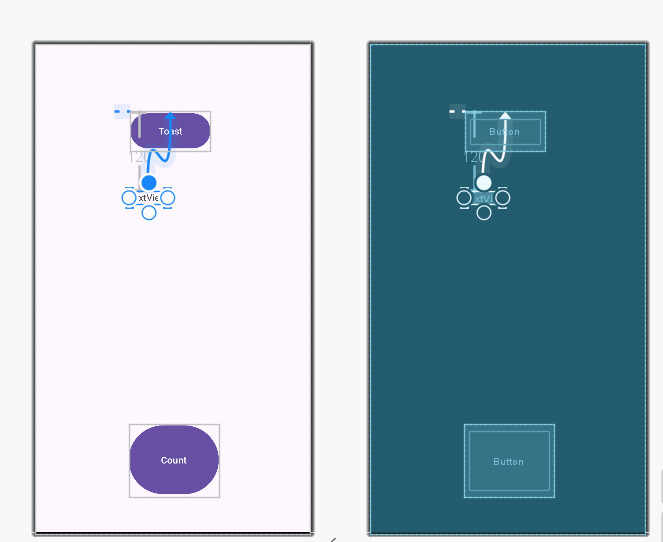
**colorPrimary** là màu chủ đạo của giao diện, một trong những màu cơ bản được định nghĩa sẵn trong tệp **colors.xml**. Màu này được sử dụng cho thanh ứng dụng (**app bar**). Việc sử dụng các màu cơ bản cho các phần tử UI khác giúp tạo ra một giao diện đồng bộ. Bạn sẽ tìm hiểu thêm về chủ đề giao diện (**app themes**) và **Material Design** trong bài học khác.

#### Nhiệm vụ 4 : Thêm EditText và thay đổi thuộc tính.

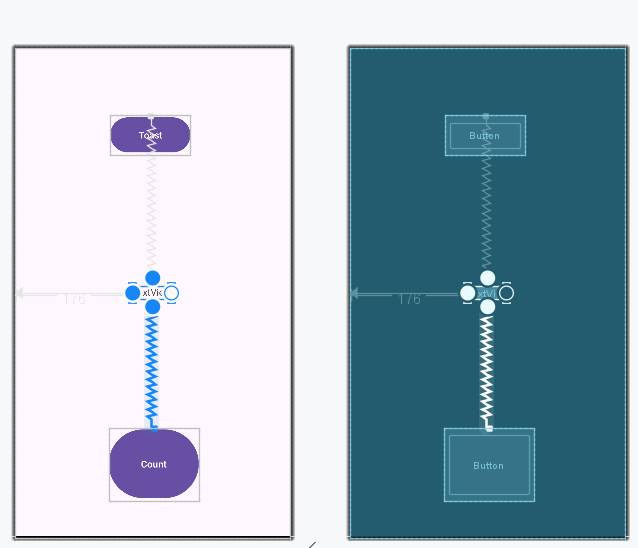
Một trong những lợi ích của **ConstraintLayout** là khả năng căn chỉnh hoặc ràng buộc các phần tử so với các phần tử khác. Trong nhiệm vụ này, bạn sẽ thêm một **TextView** vào giữa bố cục, ràng buộc nó theo chiều ngang với lề và theo chiều dọc với hai **Button**. Sau đó, bạn sẽ thay đổi các thuộc tính của **TextView** trong bảng **Attributes**.

##### Thêm EditText và các ràng buộc.

1. Kéo một **TextView** từ bảng **Palette** vào phần trên của bố cục, sau đó kéo một ràng buộc từ cạnh trên của **TextView** đến chốt (**handle**) ở cạnh dưới của nút **Toast**. Điều này sẽ ràng buộc **TextView** nằm bên dưới nút **Toast**.



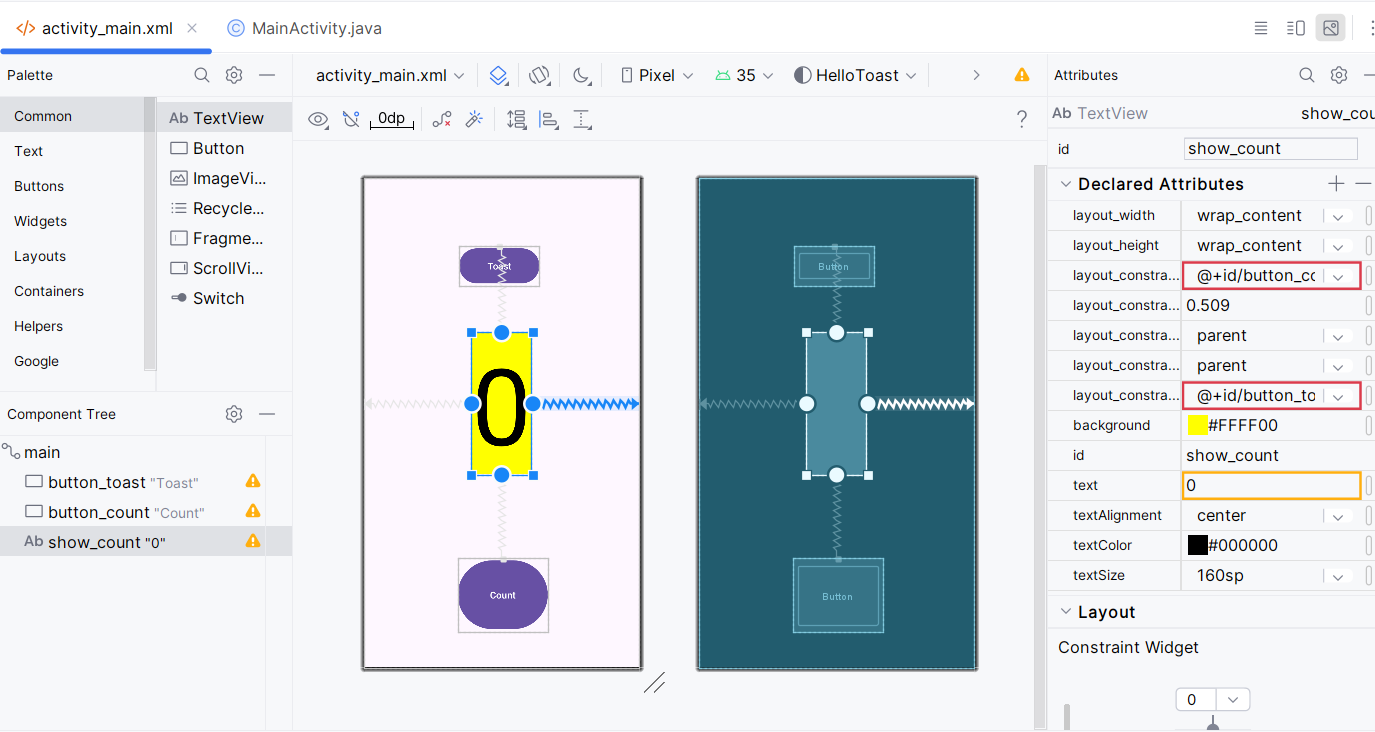
1. Kéo một ràng buộc từ cạnh dưới của **TextView** đến chốt ở cạnh trên của nút **Count**, và kéo các ràng buộc từ hai bên của **TextView** đến hai bên của bố cục. Điều này giúp **TextView** nằm ở giữa bố cục, giữa hai nút **Button**.



##### Thiết lập thuộc tính cho TextView

Với TextView được chọn, hãy mở bảng thuộc tính (Attributes) nếu chưa mở. Thiết lập các thuộc tính cho TextView như trong hình động bên dưới. Các thuộc tính chưa gặp trước đó sẽ được giải thích sau hình:

1. Đặt **ID** thành **show\_count**.
2. Đặt **text** thành **0**.
3. Đặt **textSize** thành **160dp**.
4. Đặt **textStyle** thành **B** (đậm) và **textAlignment** thành **Center** (căn giữa đoạn văn).
5. Thay đổi kích thước ngang và dọc của chế độ xem (**layout\_width** và **layout\_height**) thành **match\_constraint**.
6. Đặt **textColor** thành **@color/colorPrimary**.
7. Cuộn xuống bảng thuộc tính, nhấn **View all attributes**, tiếp tục cuộn xuống trang thứ hai để tìm thuộc tính **background**, sau đó nhập **#FFF00** để đặt màu vàng.
8. Cuộn xuống thuộc tính **gravity**, mở rộng **gravity**, và chọn **center\_ver** (căn giữa theo chiều dọc).



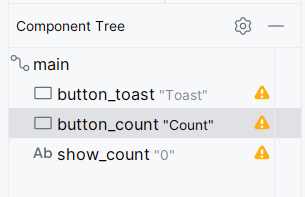
* **textSize**: Kích thước chữ của **TextView**. Trong bài học này, kích thước được đặt là **160sp**. **sp** (scale-independent pixel) là đơn vị tỷ lệ tương thích với mật độ màn hình và cài đặt kích thước phông chữ của người dùng. Khi chỉ định kích thước phông chữ, nên sử dụng đơn vị **dp** để đảm bảo kích thước phù hợp với cả mật độ màn hình và sở thích người dùng.
* **textStyle** và **textAlignment**: Kiểu chữ được đặt thành **B** (đậm) và căn chỉnh văn bản được đặt thành **center** (căn giữa đoạn văn).
* **gravity**: Thuộc tính **gravity** xác định cách một **View** được căn chỉnh trong **View** hoặc **ViewGroup** cha. Trong bước này, **TextView** được căn giữa theo chiều dọc bên trong **ConstraintLayout** cha.

Lưu ý rằng thuộc tính **background** xuất hiện ở trang đầu tiên của bảng thuộc tính đối với **Button**, nhưng nằm ở trang thứ hai đối với **TextView**. Bảng thuộc tính thay đổi tùy theo loại **View**: Các thuộc tính phổ biến nhất xuất hiện ở trang đầu tiên, phần còn lại nằm ở trang thứ hai. Để quay lại trang đầu tiên của bảng thuộc tính, nhấn vào biểu tượng trên thanh công cụ ở đầu bảng.

#### Nhiệm vụ 5 : Chỉnh sửa layout trong XML.

##### 5.1 Mở tệp bố cục XML.

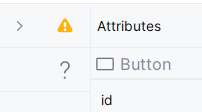
**Giao diện của ứng dụng Hello Toast gần như đã hoàn thành!** Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi phần tử UI trong **Component Tree**. Di chuột qua các dấu chấm than này để xem thông báo cảnh báo, như minh họa bên dưới. Tất cả ba phần tử đều có chung một cảnh báo: **Các chuỗi được mã hóa cứng (hardcoded) nên sử dụng tài nguyên.**



Cách dễ nhất để khắc phục sự cố bố cục là chỉnh sửa trực tiếp trong XML. Mặc dù **Layout Editor** là một công cụ mạnh mẽ, nhưng một số thay đổi sẽ dễ thực hiện hơn trong mã XML.

**Thực hiện các bước sau:**

1. Mở tệp **activity\_main.xml** (nếu chưa mở).
2. Nhấp vào tab **Text** ở cuối **Layout Editor**.



**Trình chỉnh sửa XML sẽ xuất hiện**, thay thế các bảng thiết kế và bản vẽ bố cục (**Design** và **Blueprint**). Như trong hình bên dưới, các cảnh báo được đánh dấu—bao gồm các chuỗi được mã hóa cứng **"Toast"** và **"Count"**. (Chuỗi **"0"** cũng bị đánh dấu nhưng không hiển thị trong hình minh họa.)

Di chuột qua chuỗi **"Toast"** để xem thông báo cảnh báo.



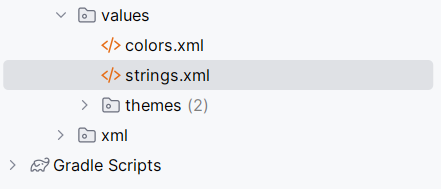
##### Trích xuất tài nguyên chuỗi.

Thay vì mã hóa cứng (hard-code) chuỗi văn bản, **thực hành tốt nhất** là sử dụng tài nguyên chuỗi (**string resources**) để đại diện cho các chuỗi. Việc lưu trữ chuỗi trong một tệp riêng giúp dễ dàng quản lý, đặc biệt nếu bạn sử dụng chúng nhiều lần. Ngoài ra, tài nguyên chuỗi là **bắt buộc** để dịch và bản địa hóa ứng dụng, vì bạn cần tạo một tệp tài nguyên chuỗi riêng cho từng ngôn ngữ.

**Thực hiện các bước sau:**

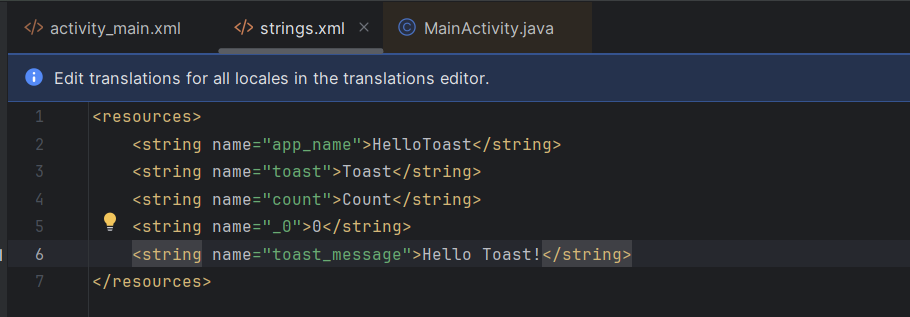
1. Nhấp vào từ **"Toast"** (cảnh báo đầu tiên được đánh dấu).
2. Nhấn **Alt + Enter** trên Windows hoặc **Option + Enter** trên macOS, sau đó chọn **Extract string resource** từ menu bật lên.
3. Nhập **button\_label\_toast** cho trường **Resource name**.
4. Nhấn **OK**. Một tài nguyên chuỗi sẽ được tạo trong tệp **values/res/strings.xml**, và chuỗi trong mã sẽ được thay thế bằng tham chiếu đến tài nguyên:
5. Trích xuất các chuỗi còn lại:

* **button\_label\_count** cho **"Count"**
* **count\_initial\_value** cho **"0"**

1. Trong **Project > Android**, mở rộng thư mục **values** trong **res**, sau đó nhấp đúp vào **strings.xml** để xem các tài nguyên chuỗi trong tệp này.
2. 



1. Bạn cần thêm một chuỗi mới để sử dụng trong bước tiếp theo. **Thêm vào tệp strings.xml một tài nguyên chuỗi mới** với:
   * **Tên:** toast\_message
   * **Giá trị:** "Hello Toast!"



### ****Lưu ý:****

Tài nguyên chuỗi bao gồm **tên ứng dụng**, xuất hiện trên thanh ứng dụng (App Bar) nếu bạn tạo dự án với mẫu **Empty Template**. Bạn có thể thay đổi tên ứng dụng bằng cách chỉnh sửa giá trị **app\_name** trong **strings.xml**.

#### Nhiệm vụ 6 : Tạo sự kiện onClick cho các Button

##### Thêm thuộc tính onClick và trình xử lý cho từng Button.

**Trình xử lý sự kiện nhấp chuột (click handler)** là phương thức được gọi khi người dùng nhấp hoặc chạm vào một phần tử giao diện có thể nhấp. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường **onClick** ở bảng **Attributes** của tab **Design**. Bạn cũng có thể chỉ định tên của phương thức xử lý trong trình chỉnh sửa XML bằng cách thêm thuộc tính **android:onClick** vào **Button**. Chúng ta sẽ sử dụng phương pháp thứ hai vì hiện tại các phương thức xử lý chưa được tạo, và trình chỉnh sửa XML cung cấp cách tự động tạo các phương thức này.

1. Khi trình chỉnh sửa XML đang mở (tab **Text**), tìm **Button** có **android:id** được đặt thành **button\_toast**.
2. Thêm thuộc tính **android:onClick** vào cuối phần tử **button\_toast**, sau thuộc tính cuối cùng và trước ký hiệu **/>**.
3. Nhấp vào biểu tượng **bóng đèn đỏ** xuất hiện bên cạnh thuộc tính. Chọn **Create click handler**, chọn **MainActivity**, rồi nhấp **OK**.

* Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy nhấp vào tên phương thức (**"showToast"**), nhấn **Alt + Enter** (hoặc **Option + Enter** trên Mac), chọn **Create 'showToast(view)' in MainActivity**, rồi nhấn **OK**.
  + Hành động này sẽ tạo một phương thức **showToast()** trong **MainActivity**.



1. Lặp lại các bước trên với **button\_count**: Thêm thuộc tính **android:onClick** và tạo trình xử lý sự kiện.



1. Nếu MainActivity.java chưa được mở, hãy mở rộng thư mục java trong chế độ xem Project > Android,

mở rộng com.example.hellotoast, sau đó nhấp đúp vào MainActivity.

Trình chỉnh sửa mã sẽ hiển thị nội dung của MainActivity.



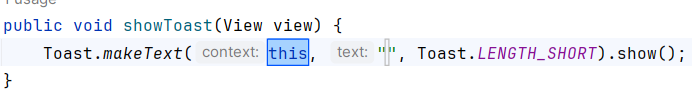
##### Chỉnh sửa trình xử lý sự kiện của Button Toast

Bạn sẽ chỉnh sửa phương thức showToast()—trình xử lý nhấp vào nút Toast trong MainActivity—để hiển thị một thông báo.

Toast cung cấp cách hiển thị một thông báo đơn giản trong một cửa sổ bật lên nhỏ. Nó chỉ chiếm không gian cần thiết cho thông báo. Hoạt động hiện tại vẫn hiển thị và có thể tương tác. Toast có thể hữu ích để kiểm tra tính tương tác trong ứng dụng của bạn—thêm một thông báo Toast để hiển thị kết quả của việc nhấn một nút hoặc thực hiện một hành động.

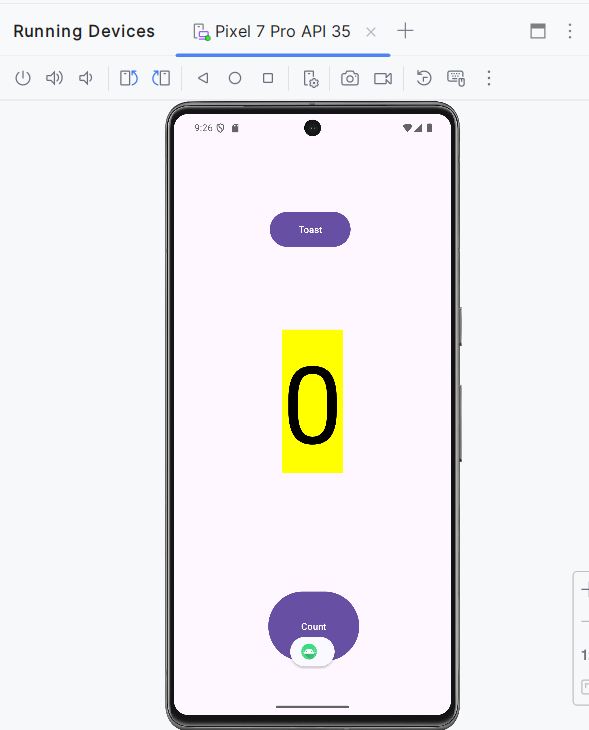
Thực hiện các bước sau để chỉnh sửa trình xử lý nhấp vào nút Toast:

1. Xác định vị trí phương thức showToast() vừa tạo.
2. Để tạo một thể hiện của Toast, hãy gọi phương thức tạo makeText() của lớp Toast. Câu lệnh này chưa hoàn chỉnh cho đến khi bạn hoàn thành tất cả các bước.
3. Cung cấp ngữ cảnh của Activity ứng dụng. Vì Toast hiển thị trên giao diện Activity, hệ thống cần thông tin về Activity hiện tại. Khi bạn đã ở trong Activity cần ngữ cảnh, hãy sử dụng this làm lối tắt.
4. Cung cấp thông báo để hiển thị, chẳng hạn như một tài nguyên chuỗi (chuỗi toast\_message bạn đã tạo ở bước trước). Tài nguyên chuỗi toast\_message được xác định bởi R.string.
5. Cung cấp thời gian hiển thị cho Toast. Ví dụ, Toast.LENGTH\_SHORT sẽ hiển thị thông báotrongmộtkhoảngthờigian Cung cấp thời gian hiển thị cho **Toast**. Ví dụ, **Toast.LENGTH\_SHORT** sẽ hiển thị thông báo trong một khoảng thời gian ngắn.  
   Thời gian hiển thị của **Toast** có thể là **Toast.LENGTH\_LONG** hoặc **Toast.LENGTH\_SHORT**.  
   Thời gian thực tế khoảng **3.5 giây** đối với **Toast.LENGTH\_LONG** và **2 giây** đối với **Toast.LENGTH\_SHORT**.



1. Hiển thị **Toast** bằng cách gọi phương thức **show()**.

Chạy ứng dụng và kiểm tra xem thông báo **Toast** có xuất hiện khi nhấn nút **Toast** hay không.



##### Chỉnh sửa trình xử lý sự kiện của Button Toast

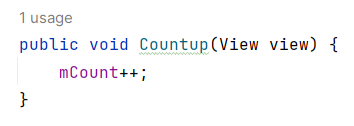
Bây giờ bạn sẽ chỉnh sửa phương thức countUp()—trình xử lý khi nhấn nút **Count** trong MainActivity—để hiển thị số đếm hiện tại sau mỗi lần nhấn. Mỗi lần nhấn sẽ tăng số đếm lên một.

Mã xử lý phải:

* Theo dõi sự thay đổi của số đếm.
* Gửi số đếm đã cập nhật đến TextView để hiển thị.

Thực hiện các bước sau để chỉnh sửa trình xử lý khi nhấn nút **Count**:

1. Xác định vị trí phương thức countUp() vừa tạo.
2. Để theo dõi số đếm, bạn cần một biến thành viên riêng. Mỗi lần nhấn nút **Count** sẽ tăng giá trị của biến này. Nhập đoạn sau, phần này sẽ được tô đỏ và hiển thị biểu tượng bóng đèn đỏ:

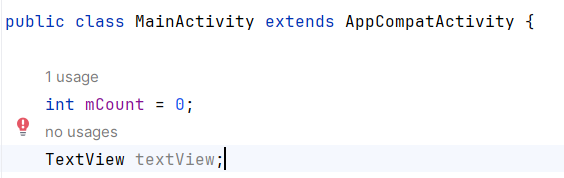


Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy chọn biểu thức mCount++. Biểu tượng bóng đèn sẽ xuất hiện sau đó.

1. Nhấp vào biểu tượng bóng đèn đỏ và chọn **Create field 'mCount'** từ menu bật lên. Thao tác này sẽ tạo một biến thành viên riêng ở đầu MainActivity, và Android Studio sẽ tự động đặt kiểu dữ liệu là int.



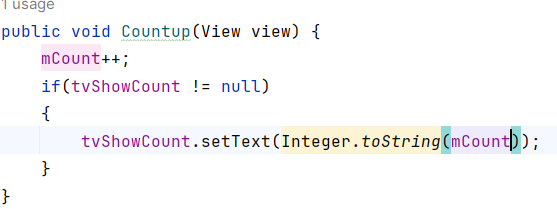
1. Thay đổi câu lệnh khai báo biến thành viên riêng để khởi tạo giá trị ban đầu là 0.
2. Bên cạnh biến trên, bạn cũng cần một biến thành viên riêng để tham chiếu đến TextView **show\_count**, biến này sẽ được thêm vào trình xử lý khi nhấn nút. Đặt tên biến là mShowCount.



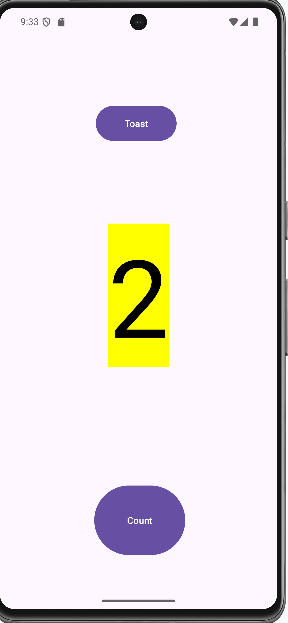
1. Khi đã có mShowCount, bạn có thể lấy tham chiếu đến TextView bằng ID đã đặt trong tệp giao diện. Để tránh gọi lại nhiều lần, hãy khai báo tham chiếu này trong phương thức onCreate(). Như bạn sẽ học trong bài học khác, phương thức onCreate() được dùng để **inflate layout**, nghĩa là đặt nội dung hiển thị của màn hình theo tệp XML. Nó cũng có thể dùng để lấy tham chiếu đến các thành phần UI khác trong giao diện, như TextView. Xác định vị trí phương thức onCreate() trong MainActivity.
2. Thêm câu lệnh findViewById vào cuối phương thức:  
   Một View, giống như một chuỗi, là một tài nguyên có thể có id. Lệnh findViewById nhận ID của một View làm tham số và trả về View. Vì phương thức này trả về một View, bạn cần ép kiểu kết quả về kiểu mong đợi, trong trường hợp này là (TextView).



1. Khi đã gán TextView vào mShowCount, bạn có thể sử dụng biến này để đặt văn bản trong TextView thành giá trị của biến mCount. Thêm dòng lệnh này vào phương thức countUp().



1. Chạy ứng dụng để kiểm tra xem số đếm có tăng khi nhấn nút **Count** hay không.



**Coding challenge**

Ứng dụng **HelloToast** hiển thị tốt khi thiết bị hoặc trình giả lập ở chế độ dọc. Tuy nhiên, khi chuyển sang chế độ ngang, nút **Count** có thể chồng lên **TextView** ở phía dưới, như hình minh họa.

Android Studio project:

**Thử thách: Thay đổi bố cục để hiển thị tốt ở cả chế độ ngang và dọc**

1. Trên máy tính, sao chép thư mục dự án **HelloToast** và đổi tên thành **HelloToastChallenge**.
2. Mở **HelloToastChallenge** trong Android Studio và thực hiện refactor. (Xem **Phụ lục: Tiện ích** để biết cách sao chép và refactor dự án.)
3. Thay đổi bố cục sao cho nút **Toast** và **Count** xuất hiện ở bên trái, như trong hình minh họa. **TextView** nằm bên cạnh nhưng chỉ đủ rộng để hiển thị nội dung. *(Gợi ý: Dùng wrap\_content.)*
4. Chạy ứng dụng ở cả chế độ ngang và dọc để kiểm tra.

**Challenge solution code**

Android Studio project:

**Tóm tắt**

**View, ViewGroup và Layouts:**

* Tất cả các phần tử giao diện người dùng (UI) đều là lớp con của View, kế thừa nhiều thuộc tính từ lớp View cha.
* Các phần tử View có thể được nhóm bên trong ViewGroup, đóng vai trò như một container. Mối quan hệ giữa chúng là **parent-child**: **parent** là ViewGroup, còn **child** là View hoặc một ViewGroup khác.
* Phương thức onCreate() được dùng để **inflate layout**, tức là đặt nội dung màn hình theo tệp XML. Nó cũng có thể dùng để lấy tham chiếu đến các phần tử UI khác trong layout.
* Một View, giống như một chuỗi, là một tài nguyên có thể có id. Phương thức findViewById nhận ID của View làm tham số và trả về View tương ứng.

**Sử dụng trình chỉnh sửa layout**

* Nhấn vào tab **Design** để thao tác trực tiếp trên giao diện, hoặc tab **Text** để chỉnh sửa mã XML.
* **Palettes pane** hiển thị các phần tử UI có thể sử dụng, còn **Component tree pane** hiển thị cây phân cấp của giao diện.
* **Blueprint pane** và **Design pane** hiển thị các phần tử UI trong layout.
* Tab **Attributes** hiển thị các thuộc tính của phần tử UI.
* **Constraint handle**: Nhấp vào điểm tròn trên mỗi cạnh của phần tử, kéo đến một điểm khác hoặc cạnh của ViewGroup để tạo ràng buộc (constraint), hiển thị dưới dạng đường zigzag.
* **Resizing handle**: Kéo góc vuông để thay đổi kích thước phần tử.
* **Autoconnect** tự động tạo ràng buộc khi kéo phần tử vào layout.
* Xóa ràng buộc bằng nút **Clear Constraints** hoặc nhấp vào điểm cụ thể của constraint cần xóa.
* **View inspector** trong tab **Attributes** giúp thay đổi chiều rộng và chiều cao của phần tử.

**Thiết lập chiều rộng và chiều cao layout**

layout\_width và layout\_height thay đổi khi chỉnh sửa trong **view inspector**. Trong ConstraintLayout, có 3 giá trị:

* + **match\_constraint**: Mở rộng phần tử theo chiều rộng hoặc chiều cao để lấp đầy phần tử cha (trừ khi có margin).
  + **wrap\_content**: Thu nhỏ phần tử vừa đủ để chứa nội dung. Nếu không có nội dung, phần tử sẽ ẩn.
  + **Sử dụng dp (density-independent pixels)** để đặt kích thước cố định, điều chỉnh phù hợp với màn hình thiết bị.

**Trích xuất chuỗi tài nguyên**

Không nên hard-code chuỗi, thay vào đó nên sử dụng **string resources**:

* 1. Nhấp vào chuỗi cần trích xuất, nhấn Alt + Enter (Option + Enter trên Mac) và chọn **Extract string resources**.
  2. Đặt tên tài nguyên (Resource name).
  3. Nhấn **OK**, chuỗi sẽ được lưu vào res/values/strings.xml và trong mã nguồn sẽ thay thế bằng @string/resource\_name.

**Xử lý sự kiện nhấn (Click Handler)**

* **Click handler** là phương thức được gọi khi người dùng nhấn vào phần tử UI.
* Cách đặt click handler cho Button:
  + Nhập tên phương thức vào **onClick** trong tab **Attributes** của **Design**.
  + Hoặc thêm thuộc tính android:onClick vào XML của Button.
* Tạo phương thức xử lý trong MainActivity
* Xem tài liệu của lớp **Button** để biết các thuộc tính của nút, và tài liệu của **TextView** để biết các thuộc tính của văn bản.

**Toast** hiển thị một thông báo ngắn trong cửa sổ popup nhỏ, chỉ chiếm không gian vừa đủ cho nội dung.

**Cách tạo Toast:**

* Gọi phương thức makeText() từ lớp Toast.
* Truyền **context của Activity** và **chuỗi thông báo** (có thể dùng string resource).
* Truyền thời gian hiển thị:
* Toast.LENGTH\_SHORT (ngắn)
* Toast.LENGTH\_LONG (dài)
* Gọi show() để hiển thị **Toast**.

### Trình chỉnh sửa bố cục

**Giới thiệu**

Như bạn đã học trong **Phần 1.3: Giao diện người dùng (UI) tương tác đầu tiên**, bạn có thể tạo UI bằng **ConstraintLayout** trong trình chỉnh sửa bố cục. ConstraintLayout cho phép sắp xếp các phần tử UI bằng cách thiết lập ràng buộc với các phần tử khác hoặc với mép của bố cục. Nó được thiết kế để giúp bạn dễ dàng kéo thả các phần tử vào giao diện.

**ConstraintLayout** là một **ViewGroup**, tức là một View đặc biệt có thể chứa các View khác (gọi là **child views**). Trong bài thực hành này, bạn sẽ tìm hiểu thêm về các tính năng của **ConstraintLayout** và trình chỉnh sửa bố cục. Ngoài ra, bài học cũng giới thiệu hai lớp con khác của **ViewGroup**:

* **LinearLayout**: Bố cục sắp xếp các phần tử con theo chiều ngang hoặc dọc.
* **RelativeLayout**: Bố cục cho phép sắp xếp các phần tử con dựa trên vị trí của các phần tử khác hoặc của chính **ViewGroup** cha.

**Kiến thức cần có**

Bạn nên biết cách:

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

**Bạn sẽ học được**

* Cách tạo bố cục riêng cho màn hình ngang (**landscape**).
* Cách tạo bố cục riêng cho máy tính bảng và màn hình lớn.
* Cách sử dụng **baseline constraint** để căn chỉnh phần tử UI với văn bản.
* Cách dùng nút **pack** và **align** để sắp xếp phần tử trong bố cục.
* Cách định vị phần tử trong **LinearLayout**.
* Cách định vị phần tử trong **RelativeLayout**.

**Bạn sẽ thực hiện**

* Tạo bố cục cho màn hình ngang.
* Tạo bố cục cho máy tính bảng và màn hình lớn.
* Điều chỉnh bố cục và thêm ràng buộc cho các phần tử UI.
* Dùng **baseline constraint** trong **ConstraintLayout** để căn chỉnh phần tử với văn bản.
* Sử dụng nút **pack** và **align** trong **ConstraintLayout** để sắp xếp phần tử.
* Chuyển bố cục sang **LinearLayout**.
* Định vị các phần tử trong **LinearLayout**.
* Chuyển bố cục sang **RelativeLayout**.
* Sắp xếp lại các phần tử trong bố cục chính để căn chỉnh tương đối với nhau.

**Tổng quan về ứng dụng**

Ứng dụng **Hello Toast** trong bài học trước sử dụng **ConstraintLayout** để sắp xếp các thành phần giao diện trong **Activity**, như minh họa trong hình dưới đây.



Để thực hành thêm với **ConstraintLayout**, bạn sẽ tạo một phiên bản bố cục khác cho chế độ ngang, như hình minh họa.

Bạn cũng sẽ học cách sử dụng **ràng buộc đường cơ sở (baseline constraints)** và một số tính năng căn chỉnh của **ConstraintLayout** bằng cách tạo một phiên bản bố cục khác dành cho máy tính bảng.

Ngoài ra, bạn sẽ tìm hiểu về các lớp con khác của **ViewGroup**, chẳng hạn như **LinearLayout** và **RelativeLayout**, đồng thời thay đổi bố cục của ứng dụng **Hello Toast** để sử dụng chúng.

#### 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 hiển thị phù hợp ở chế độ ngang hoặc dọc. Trong nhiệm vụ này, bạn sẽ học cách dễ dàng tạo các biến thể bố cục cho **chế độ ngang** (còn gọi là landscape) và **chế độ dọc** (portrait) trên điện thoại, cũng như trên các thiết bị màn hình lớn như máy tính bảng.

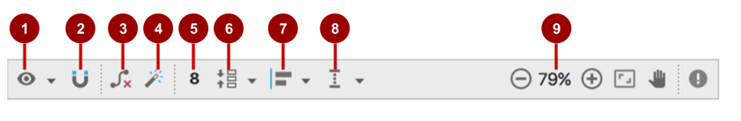


Bạn sẽ sử dụng một số nút trong **hai thanh công cụ trên cùng** của trình chỉnh sửa bố cục (**Layout Editor**):

Thanh này cho phép bạn tùy chỉnh giao diện xem trước bố cục trong trình chỉnh sửa:

1. **Chọn bề mặt thiết kế** (*Design Surface*):
   * Chọn **Design** để xem trước bố cục có màu sắc.
   * Chọn **Blueprint** để hiển thị các đường viền của các phần tử UI.
   * Chọn **Design + Blueprint** để hiển thị cả hai chế độ song song.
2. **Hướng trong trình chỉnh sửa** (*Orientation in Editor*):
   * Chọn **Portrait** hoặc **Landscape** để xem trước bố cục theo chiều dọc hoặc ngang.
   * Để tạo bố cục thay thế, chọn **Create Landscape Variation** hoặc các biến thể khác.
3. **Thiết bị trong trình chỉnh sửa** (*Device in Editor*):
   * Chọn loại thiết bị (điện thoại/máy tính bảng, Android TV, Android Wear).
4. **Phiên bản API trong trình chỉnh sửa** (*API Version in Editor*):
   * Chọn phiên bản Android để hiển thị xem trước.
5. **Chủ đề trong trình chỉnh sửa** (*Theme in Editor*):
   * Chọn chủ đề (ví dụ: **AppTheme**) để áp dụng cho bố cục xem trước.
6. **Ngôn ngữ trong trình chỉnh sửa** (*Locale in Editor*):
   * Chọn ngôn ngữ hiển thị. Danh sách này chỉ hiển thị các ngôn ngữ có trong tài nguyên chuỗi của ứng dụng.
   * Có thể chọn **Preview as Right To Left** để xem bố cục dưới dạng ngôn ngữ viết từ phải sang trái (RTL).

Thanh công cụ thứ hai cho phép bạn cấu hình giao diện của các phần tử UI trong ConstraintLayout và điều chỉnh thu phóng, di chuyển bản xem trước:



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**: Bật hoặc tắt **Autoconnect**. Khi bật, bạn có thể kéo bất kỳ phần tử nào (chẳng hạn như **Button**) đến bất kỳ phần nào của bố cục để tạo ràng buộc với bố cục cha.
3. **Xóa tất cả ràng buộc**: Xóa tất cả ràng buộc trong toàn bộ bố cục.
4. **Suy luận ràng buộc**: Tạo ràng buộc bằng suy luận.
5. **Lề mặc định**: Đặt lề mặc định.
6. **Đóng gói**: Đóng gói hoặc mở rộng các phần tử đã chọn.
7. **Căn chỉnh**: Căn chỉnh các phần tử đã chọn.
8. **Đường hướng dẫn**: Thêm các đường hướng dẫn theo chiều dọc hoặc ngang.
9. **Điều khiển thu phóng/di chuyển**: Phóng to hoặc thu nhỏ.

##### Xem trước bố cục ở chế độ ngang.

Để xem trước bố cục của ứng dụng **Hello Toast** ở chế độ ngang, thực hiện các bước sau:

1. Mở ứng dụng **Hello Toast** từ bài học trước.

**Lưu ý**: Nếu bạn đã tải xuống mã nguồn hoàn chỉnh của **HelloToast**, bạn cần xóa các bố cục **landscape** và **extra-large** đã hoàn thành trong nhiệm vụ này.

* + Chuyển từ **Project > Android** sang **Project > Project Files** trong bảng **Project**.
  + Mở rộng thư mục: **app > app > src/main > res**.
  + Chọn cả hai thư mục **layout-land** và **layout-xlarge**, sau đó chọn **Edit > Delete**.
  + Chuyển bảng **Project** trở lại **Project > Android**.

1. Mở tệp **activity\_main.xml**. Nhấp vào tab **Design** nếu nó chưa được chọn.
2. Nhấp vào nút **Orientation in Editor** trên thanh công cụ trên cùng.
3. Chọn **Switch to Landscape** trong menu thả xuống. Bố cục sẽ hiển thị ở chế độ ngang như hình bên dưới.
   * Để quay lại chế độ dọc, chọn **Switch to Portrait**.

##### . Tạo biến thể bố cục cho chế độ ngang.

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

Để khắc phục vấn đề này trong chế độ ngang mà không ảnh hưởng đến chế độ dọc, bạn có thể tạo một biến thể của bố cục ứng dụng **Hello Toast** dành riêng cho chế độ ngang. Thực hiện các bước sau:

1. Nhấp vào nút **Orientation in Editor** trên thanh công cụ trên cùng.
2. Chọn **Create Landscape Variation**.
   * Một cửa sổ chỉnh sửa mới sẽ mở với tab **land/activity\_main.xml**, hiển thị bố cục cho chế độ ngang.
   * Bạn có thể thay đổi bố cục này mà không làm thay đổi bố cục gốc ở chế độ dọc.
3. Trong bảng **Project > Android**, mở thư mục **res > layout**.
   * Bạn sẽ thấy **Android Studio** đã tự động tạo biến thể có tên **activity\_main.xml (land)**.

##### Xem trước bố cục trên các thiết bị khác nhau.

Bạn có thể xem trước bố cục trên các thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị hoặc trình giả lập. Thực hiện các bước sau:

1. **Tab** land/activity\_main.xml **vẫn nên được mở trong trình chỉnh sửa bố cục**; nếu không, hãy nhấp đúp vào tệp activity\_main.xml (land) trong thư mục layout.
2. Nhấp vào nút **Device in Editor** trên thanh công cụ trên cùng.
3. Chọn một thiết bị khác trong menu thả xuống. Ví dụ:
   * Chọn **Nexus 4**, **Nexus 5**, sau đó **Pixel** để xem sự khác biệt trong bản xem trước.
   * Những khác biệt này chủ yếu do kích thước văn bản cố định của **TextView**.

##### Thay đổi bố cục cho chế độ ngang.

Bạn có thể sử dụng bảng **Attributes** trong tab **Design** để thiết lập hoặc thay đổi thuộc tính, nhưng đôi khi việc chỉnh sửa trực tiếp mã XML trong tab **Text** sẽ nhanh hơn. Tab **Text** hiển thị mã XML và cung cấp tab **Preview** ở bên phải cửa sổ để xem trước bố cục, như hình dưới đây.

Hình minh họa phía trên hiển thị:

1. **Tab Preview**, dùng để hiển thị khung xem trước.
2. **Khung xem trước**.
3. **Mã XML**.

**Thực hiện thay đổi bố cục bằng các bước sau:**

1. **Tab** land/activity\_main.xml **vẫn nên được mở trong trình chỉnh sửa bố cục**; nếu không, hãy nhấp đúp vào tệp activity\_main.xml (land) trong thư mục layout.
2. Nhấp vào tab **Text** và tab **Preview** (nếu chưa được chọn).
3. Tìm phần tử **TextView** trong mã XML.
4. Thay đổi thuộc tính:
5. Chọn các thiết bị khác nhau trong menu **Device in Editor** để kiểm tra bố cục trên nhiều thiết bị ở chế độ ngang.
6. **Chạy ứng dụng trên trình giả lập hoặc thiết bị thật** và **chuyển đổi giữa chế độ dọc và ngang** để quan sát cả hai bố cục.

##### Tạo một biến thể bố cục cho máy tính bảng.

Như bạn đã học trước đó, bạn có thể xem trước bố cục trên các thiết bị khác nhau bằng cách nhấp vào nút **Device in Editor** trên thanh công cụ trên cùng.

Nếu bạn chọn một thiết bị như **Nexus 10** (máy tính bảng) từ menu, bạn sẽ thấy rằng bố cục không phù hợp với màn hình máy tính bảng:

* Văn bản trên mỗi **Button** quá nhỏ.
* Việc sắp xếp các phần tử **Button** ở trên và dưới không tối ưu cho màn hình lớn.

Để khắc phục điều này mà không ảnh hưởng đến bố cục trên điện thoại ở chế độ dọc và ngang, bạn có thể tạo một biến thể bố cục riêng dành cho máy tính bảng.

* **Thực hiện các bước sau:**

1. Nhấp vào tab **Design** (nếu chưa được chọn) để hiển thị khung thiết kế và bản vẽ bố cục.
2. Nhấp vào nút **Orientation in Editor** trên thanh công cụ trên cùng.
3. Chọn **Create layout x-large Variation**.
   * Một cửa sổ chỉnh sửa mới sẽ mở với tab **xlarge/activity\_main.xml**, hiển thị bố cục dành riêng cho thiết bị có màn hình lớn.
   * Trình chỉnh sửa cũng tự động chọn một thiết bị máy tính bảng như **Nexus 9** hoặc **Nexus 10** để xem trước.
   * Bạn có thể thay đổi bố cục này mà không ảnh hưởng đến các bố cục khác.

##### Thay đổi biến thể bố cục máy tính bảng

Bạn có thể sử dụng bảng **Attributes** trong tab **Design** để thay đổi thuộc tính cho bố cục này.

1. **Thực hiện các bước sau:**
2. **Tắt công cụ Autoconnect** trong thanh công cụ. Đảm bảo rằng công cụ này đã được vô hiệu hóa.
3. **Xóa tất cả các ràng buộc (constraints)** trong bố cục bằng cách nhấp vào nút **Clear All Constraints** trên thanh công cụ.
   * Khi đã xóa các ràng buộc, bạn có thể tự do di chuyển và thay đổi kích thước các phần tử trong bố cục.
4. **Di chuyển và thay đổi kích thước phần tử**
   * Trình chỉnh sửa bố cục cung cấp tay cầm thay đổi kích thước ở bốn góc của một phần tử.
   * Trong **Component Tree**, chọn **TextView** có ID là show\_count.
   * Để dọn chỗ và dễ dàng kéo các phần tử **Button**, kéo một góc của **TextView** để thay đổi kích thước của nó.

*Lưu ý:* Việc thay đổi kích thước một phần tử sẽ gán cứng (hardcode) giá trị width và height. Tránh đặt kích thước cố định vì nó có thể không hiển thị đúng trên các màn hình có kích thước và mật độ khác nhau. Bạn chỉ đang tạm thời di chuyển phần tử này và sẽ chỉnh sửa lại kích thước sau.

1. **Thay đổi thuộc tính của Button "Toast"**
   * Trong **Component Tree**, chọn **Button** có ID button\_toast.
   * Mở tab **Attributes**, thay đổi các giá trị sau:
     + textSize → **60sp**
     + layout\_width → **wrap\_content**
   * Ở phía bên phải, bạn có thể sử dụng **view inspector** để thay đổi **layout\_width** thành wrap\_content.

*Sử dụng wrap\_content giúp Button có thể tự động điều chỉnh kích thước nếu văn bản bên trong thay đổi khi được dịch sang ngôn ngữ khác.*

1. **Thay đổi thuộc tính của Button "Count"**
   * Trong **Component Tree**, chọn **Button** có ID button\_count.
   * Thay đổi các giá trị sau:
     + textSize → **60sp**
     + layout\_width → **wrap\_content**
   * Kéo Button này lên trên **TextView** đến một vị trí trống trong bố cục.

##### Sử dụng ràng buộc đường cơ sở.

Bạn có thể căn chỉnh một phần tử giao diện người dùng chứa văn bản (chẳng hạn như **TextView** hoặc **Button**) với một phần tử văn bản khác bằng cách sử dụng **Baseline Constraint**. Điều này giúp văn bản trong các phần tử này được căn chỉnh chính xác theo cùng một đường cơ sở.

1. **Thực hiện các bước sau:**
2. **Thêm ràng buộc vị trí cho các nút Button:**
   * Ràng buộc **button\_toast** vào mép trên và bên trái của bố cục.
   * Kéo **button\_count** đến gần **button\_toast**, sau đó ràng buộc **button\_count** vào mép trái của **button\_toast**.
3. **Sử dụng Baseline Constraint:**
   * Chọn phần tử **button\_count**.
   * Di chuyển con trỏ chuột lên phần tử này cho đến khi biểu tượng **baseline constraint** xuất hiện bên dưới.
   * Nhấp vào nút **baseline constraint** để hiển thị tay cầm căn chỉnh màu xanh lá cây.
   * Kéo tay cầm căn chỉnh đến đường cơ sở của **button\_toast**.

##### Mở rộng các Button theo chiều ngang.

Bạn có thể sử dụng nút **Pack** trên thanh công cụ để sắp xếp hoặc mở rộng các phần tử UI theo chiều ngang, giúp các **Button** được căn đều trong bố cục.

1. **Thực hiện các bước sau:**
2. Trong **Component Tree**, chọn **button\_count**.
3. Giữ phím **Shift**, sau đó chọn **button\_toast** để chọn cả hai Button.
4. Nhấp vào **Pack** trên thanh công cụ và chọn **Expand Horizontally**.
   * Lúc này, các Button sẽ mở rộng theo chiều ngang để lấp đầy bố cục.
5. **Hoàn thiện bố cục:**
   * Ràng buộc **show\_count** (**TextView**) vào phía dưới của **button\_toast**, cũng như vào hai cạnh bên và cạnh dưới của bố cục.
6. **Chỉnh sửa thuộc tính của show\_count:**
   * Đặt layout\_width và layout\_height thành **Match Constraints**.
   * Thay đổi textSize thành **200sp**.
7. **Xem trước ở chế độ ngang:**
   * Nhấp vào **Orientation in Editor** trên thanh công cụ.
   * Chọn **Switch to Landscape** để xem bố cục máy tính bảng trong chế độ ngang.
   * Chọn **Switch to Portrait** để quay lại chế độ dọc.
8. **Chạy ứng dụng trên nhiều thiết bị khác nhau:**
   * Chạy ứng dụng trên các trình giả lập khác nhau.
   * Chuyển đổi giữa chế độ ngang và dọc để kiểm tra giao diện trên các thiết bị có kích thước màn hình và mật độ điểm ảnh khác nhau.

**Coding challenge 1**

Thử thách: Để phù hợp với chế độ ngang (landscape) của máy tính bảng, bạn có thể căn giữa các phần tử **Button** trong **activity\_main.xml (xlarge)** để chúng hiển thị như trong hình dưới đây.

Gợi ý: Chọn các phần tử, nhấp vào nút **Align** trên thanh công cụ và chọn **Center Horizontally**.

#### Nhiệm vụ 2 : thay đổi bố cục thành LinearLayout

**LinearLayout** là một **ViewGroup** sắp xếp tập hợp các view theo hàng ngang hoặc dọc. Đây là một trong những bố cục phổ biến nhất vì đơn giản và nhanh chóng. **LinearLayout** thường được sử dụng bên trong một nhóm view khác để sắp xếp các phần tử giao diện người dùng theo chiều ngang hoặc dọc.

**LinearLayout yêu cầu các thuộc tính sau:**

* **layout\_width**
* **layout\_height**
* **orientation**

**Các giá trị có thể có cho layout\_width và layout\_height:**

* **match\_parent**: Mở rộng view để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao. Khi **LinearLayout** là root view, nó sẽ mở rộng theo kích thước của màn hình (phần tử cha).
* **wrap\_content**: Thu nhỏ kích thước của view sao cho nó chỉ vừa đủ để chứa nội dung của nó. Nếu không có nội dung, view sẽ trở nên vô hình.
* **Giá trị cố định theo dp (density-independent pixels)**: Xác định kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị. Ví dụ, **16dp** có nghĩa là 16 pixel độc lập với mật độ.

**Các giá trị có thể có cho orientation:**

* **horizontal**: Sắp xếp các view từ trái sang phải.
* **vertical**: Sắp xếp các view từ trên xuống dưới.
* **Thay đổi nhóm bố cục gốc thành LinearLayout**

Trong nhiệm vụ này, bạn sẽ thay đổi nhóm bố cục gốc **ConstraintLayout** của ứng dụng **Hello Toast** thành **LinearLayout** để thực hành sử dụng **LinearLayout**.

##### 2.1 Thay đổi nhóm bố cục gốc thành LinearLayout

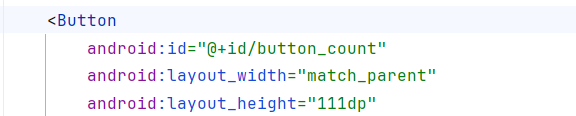
1. Mở ứng dụng **Hello Toast** từ nhiệm vụ trước.
2. Mở tệp **activity\_main.xml** (nếu chưa mở) và nhấp vào tab **Text** ở cuối khung chỉnh sửa để xem mã XML. Ở đầu mã XML có dòng thẻ sau:
3. Thay đổi thẻ **<android.support.constraint.ConstraintLayout** thành **<LinearLayout**, để mã trông như sau:
4. Đảm bảo rằng thẻ đóng ở cuối mã đã thay đổi thành **</LinearLayout>** (Android Studio sẽ tự động thay đổi thẻ đóng nếu bạn thay đổi thẻ mở). Nếu nó không thay đổi tự động, hãy thay đổi thủ công.
5. Dưới dòng thẻ **<LinearLayout**, thêm thuộc tính sau sau **android:layout\_height**:

Sau khi thực hiện các thay đổi này, một số thuộc tính XML của các phần tử khác sẽ bị gạch chân màu đỏ vì chúng được sử dụng với **ConstraintLayout** và không phù hợp với **LinearLayout**.

##### 2.2 Thay đổi thuộc tính phần tử cho LinearLayout

Làm theo các bước sau để thay đổi thuộc tính của phần tử giao diện người dùng sao cho phù hợp với **LinearLayout**:

1. Mở ứng dụng **Hello Toast** từ nhiệm vụ trước.
2. Mở tệp **activity\_main.xml** (nếu chưa mở) và nhấp vào tab **Text**.
3. Tìm phần tử **button\_toast** và thay đổi thuộc tính sau:



1. Xóa các thuộc tính sau khỏi phần tử **button\_toast**
2. Tìm phần tử **button\_count** và thay đổi thuộc tính sau
3. Xóa các thuộc tính sau khỏi phần tử **button\_count**



1. Tìm phần tử **show\_count** (**TextView**) và thay đổi các thuộc tính sau
2. Xóa các thuộc tính sau khỏi phần tử **show\_count**



1. Nhấp vào tab **Preview** ở bên phải cửa sổ **Android Studio** (nếu chưa được chọn) để xem trước bố cục cho đến thời điểm hiện tại.

##### 2.3 Thay đổi vị trí của các phần tử trong LinearLayout

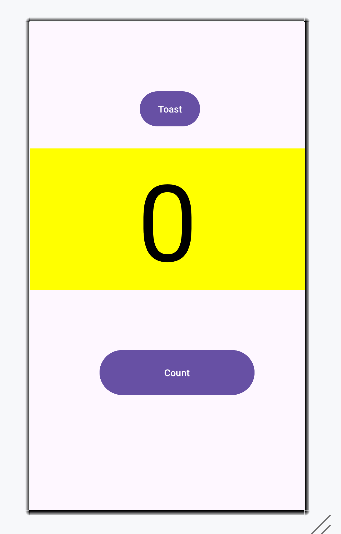
**LinearLayout** sắp xếp các phần tử theo hàng ngang hoặc dọc. Bạn đã thêm thuộc tính android:orientation="vertical" cho **LinearLayout**, vì vậy các phần tử được xếp chồng lên nhau theo chiều dọc như trong hình trước đó.

Để thay đổi vị trí sao cho nút **Count** nằm ở dưới cùng, hãy làm theo các bước sau:

1. Mở ứng dụng **Hello Toast** từ nhiệm vụ trước.
2. Mở tệp **activity\_main.xml** (nếu chưa mở) và nhấp vào tab **Text**.
3. Chọn phần tử **button\_count** (**Button**) và tất cả thuộc tính của nó, từ thẻ <Button> đến thẻ đóng />, sau đó chọn **Edit > Cut**.
4. Nhấp sau thẻ đóng /> của phần tử **TextView** nhưng trước thẻ đóng </LinearLayout>, sau đó chọn **Edit > Paste**.
5. *(Tùy chọn)* Để sửa lỗi thụt lề hoặc khoảng cách cho mã XML, chọn **Code > Reformat Code** để định dạng lại với khoảng cách và thụt lề phù hợp.

Bằng cách di chuyển nút **button\_count** (**Button**) xuống dưới **TextView**, bố cục hiện tại gần giống với bố cục ban đầu, với nút **Count** nằm ở phía dưới.

Bản xem trước của bố cục hiện tại trông như sau:



##### Thêm thuộc tính weight cho phần tử TextView

Việc chỉ định các thuộc tính **gravity** và **weight** giúp bạn kiểm soát tốt hơn việc sắp xếp các **View** và nội dung trong **LinearLayout**.

Thuộc tính **android:gravity** xác định cách căn chỉnh nội dung bên trong một **View**. Trong bài học trước, bạn đã đặt thuộc tính này cho **show\_count** (**TextView**) để căn giữa nội dung (chữ số 0) trong **TextView**.



Thuộc tính **android:layout\_weight** chỉ định lượng không gian bổ sung trong **LinearLayout** sẽ được phân bổ cho **View**. Nếu chỉ có một **View** có thuộc tính này, nó sẽ chiếm toàn bộ không gian còn lại. Nếu nhiều **View** có thuộc tính **weight**, không gian sẽ được chia theo tỷ lệ. Ví dụ: nếu hai **Button** có **weight = 1** và **TextView** có **weight = 2** (tổng cộng là 4), mỗi **Button** nhận ¼ không gian, trong khi **TextView** nhận một nửa.

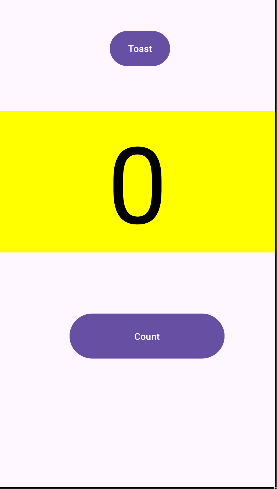
Trên các thiết bị khác nhau, bố cục có thể hiển thị phần tử **show\_count** (**TextView**) với kích thước thay đổi giữa hai nút **Toast** và **Count**. Để đảm bảo **TextView** mở rộng để lấp đầy không gian trống trên mọi thiết bị, bạn cần chỉ định thuộc tính **android:gravity** cho **TextView**.

* **Thực hiện:**

1. Mở ứng dụng **Hello Toast** từ bài trước.
2. Mở tệp **activity\_main.xml** (nếu chưa mở), sau đó nhấp vào tab **Text**.
3. Tìm phần tử **show\_count** (**TextView**) và thêm thuộc tính sau:



Sau khi thực hiện, bản xem trước sẽ hiển thị như hình bên dưới.



* Phần tử **show\_count** (**TextView**) sẽ chiếm toàn bộ không gian giữa hai nút.
* Bạn có thể xem trước bố cục trên các thiết bị khác nhau bằng cách nhấp vào nút **Device in Editor** trên thanh công cụ của bảng xem trước và chọn thiết bị khác.
* Dù chọn thiết bị nào, **show\_count** (**TextView**) vẫn sẽ lấp đầy không gian giữa hai nút.

#### Nhiệm vụ 3: Thay đổi bố cục sang RelativeLayout

**RelativeLayout** là một nhóm giao diện trong đó mỗi thành phần được định vị và căn chỉnh tương đối với các thành phần khác trong cùng nhóm. Trong nhiệm vụ này, bạn sẽ học cách xây dựng bố cục bằng **RelativeLayout**.

##### 3.1 Thay đổi LinearLayout thành RelativeLayout

Một cách dễ dàng để thay đổi **LinearLayout** thành **RelativeLayout** là chỉnh sửa các thuộc tính XML trong tab **Text**.

1. Mở tệp **activity\_main.xml** và nhấp vào tab **Text** ở cuối bảng chỉnh sửa để xem mã XML.
2. Thay đổi <LinearLayout ở đầu thành <RelativeLayout.>
3. Cuộn xuống để đảm bảo rằng thẻ đóng </LinearLayout> đã được thay đổi thành </RelativeLayout>. Nếu chưa, hãy thay đổi thủ công.

##### 3.2 Sắp xếp lại các thành phần trong RelativeLayout

Một cách dễ dàng để sắp xếp và định vị các thành phần trong **RelativeLayout** là chỉnh sửa các thuộc tính XML trong tab **Text**.

1. Nhấp vào tab **Preview** ở bên cạnh trình chỉnh sửa (nếu chưa được chọn) để xem trước bố cục.

Sau khi thay đổi sang **RelativeLayout**, trình chỉnh sửa bố cục cũng đã tự động thay đổi một số thuộc tính của thành phần giao diện, cụ thể:

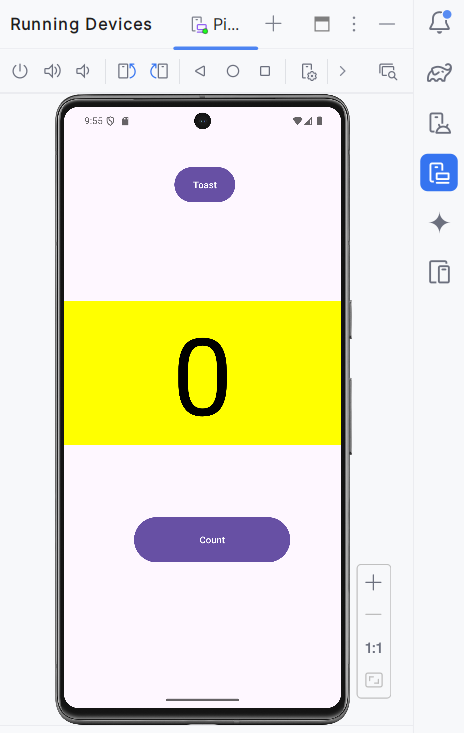
* + Nút **Count** (**button\_count**) đang chồng lên nút **Toast** (**button\_toast**), khiến nút **Toast** không hiển thị.
  + Phần trên của **TextView** (**show\_count**) đang chồng lên các nút **Button**.

1. Thêm thuộc tính **android:layout\_below** vào nút **button\_count** để đặt nút này ngay bên dưới **show\_count** (**TextView**)
2. Thêm thuộc tính **android:layout\_centerHorizontal** vào **button\_count** để căn giữa nút theo chiều ngang trong **RelativeLayout**.



1. Thêm các thuộc tính sau vào **show\_count** (**TextView**):
   * **android:layout\_alignParentLeft** để căn trái theo **RelativeLayout**.
   * **android:layout\_alignParentStart** để căn chỉnh theo hướng ngôn ngữ, giúp hiển thị đúng trên các thiết bị có ngôn ngữ từ phải sang trái.
2. Xóa thuộc tính **android:layout\_weight="1"** khỏi **show\_count**, vì thuộc tính này không áp dụng cho **RelativeLayout**.

Sau khi thực hiện các thay đổi, bố cục sẽ hiển thị :



**Gợi ý**: **RelativeLayout** giúp dễ dàng định vị các thành phần trong bố cục. Để tìm hiểu thêm về cách định vị các thành phần trong **RelativeLayout**, hãy tham khảo tài liệu **"Positioning Views"** trong hướng dẫn về **RelativeLayout** của API.

**Tóm tắt**

**Sử dụng trình chỉnh sửa bố cục để xem trước và tạo các biến thể**

* Để xem trước bố cục ứng dụng theo hướng ngang trong trình chỉnh sửa bố cục, nhấn vào nút **Orientation in Editor** trên thanh công cụ và chọn **Switch to Landscape**. Chọn **Switch to Portrait** để quay lại hướng dọc.
* Để tạo một biến thể bố cục dành riêng cho chế độ ngang, nhấn vào **Orientation in Editor** và chọn **Create Landscape Variation**. Một cửa sổ chỉnh sửa mới sẽ mở ra với tab **land/activity\_main.xml**, hiển thị bố cục cho chế độ ngang.
* Để xem trước bố cục trên các thiết bị khác nhau mà không cần chạy ứng dụng trên thiết bị thật hoặc trình giả lập, nhấn vào nút **Device in Editor** trên thanh công cụ và chọn một thiết bị.
* Để tạo một biến thể bố cục dành cho máy tính bảng (màn hình lớn hơn), nhấn **Orientation in Editor** và chọn **Create layout x-large Variation**. Một cửa sổ chỉnh sửa mới sẽ mở với tab **xlarge/activity\_main.xml**, hiển thị bố cục cho thiết bị có màn hình lớn.

**Sử dụng ConstraintLayout**

* Để xóa tất cả ràng buộc trong một bố cục có **ConstraintLayout** làm gốc, nhấn vào nút **Clear All Constraints** trên thanh công cụ.
* Có thể căn chỉnh một phần tử giao diện chứa văn bản (như **TextView** hoặc **Button**) với một phần tử khác có văn bản bằng cách sử dụng **baseline constraint** để căn chỉnh các dòng cơ sở của văn bản.
* Để tạo **baseline constraint**, di chuột qua phần tử giao diện cho đến khi nút **baseline constraint** xuất hiện bên dưới phần tử.
* Nút **Pack** trên thanh công cụ cung cấp tùy chọn để gom nhóm hoặc mở rộng các phần tử giao diện đã chọn, giúp sắp xếp đều các nút **Button** theo chiều ngang trong bố cục.

**Sử dụng LinearLayout**

* **LinearLayout** là một **ViewGroup** sắp xếp các thành phần giao diện theo hàng ngang hoặc dọc.
* **LinearLayout** yêu cầu các thuộc tính **layout\_width**, **layout\_height**, và **orientation**.
* **match\_parent** (cho **layout\_width** hoặc **layout\_height**) giúp phần tử mở rộng toàn bộ chiều rộng hoặc chiều cao của phần tử cha. Nếu **LinearLayout** là gốc, nó sẽ mở rộng ra toàn bộ màn hình.
* **wrap\_content** (cho **layout\_width** hoặc **layout\_height**) giúp phần tử chỉ chiếm diện tích vừa đủ để chứa nội dung bên trong. Nếu không có nội dung, phần tử sẽ trở nên vô hình.
* **Giá trị dp cố định** (density-independent pixels) cho **layout\_width** hoặc **layout\_height** giúp đặt kích thước cố định, tự động điều chỉnh theo mật độ màn hình. Ví dụ, **16dp** có nghĩa là 16 pixel độc lập với mật độ màn hình.
* **orientation** có thể là **horizontal** để sắp xếp theo chiều ngang (trái sang phải) hoặc **vertical** để sắp xếp theo chiều dọc (trên xuống dưới).
* **gravity** và **weight** cung cấp thêm khả năng kiểm soát cách sắp xếp các phần tử:
  + **android:gravity** xác định vị trí nội dung bên trong **View**.
  + **android:layout\_weight** xác định phần trăm không gian bổ sung mà một **View** sẽ chiếm trong **LinearLayout**. Nếu chỉ có một phần tử có **layout\_weight**, nó sẽ chiếm toàn bộ không gian thừa. Nếu có nhiều phần tử, không gian được phân bổ theo tỷ lệ. Ví dụ, nếu hai **Button** có **weight = 1** và một **TextView** có **weight = 2**, tổng là **4**, thì mỗi **Button** chiếm **1/4** diện tích thừa, còn **TextView** chiếm **1/2** diện tích thừa.

**Sử dụng RelativeLayout**

* **RelativeLayout** là một **ViewGroup** trong đó mỗi thành phần được định vị và căn chỉnh tương đối với các thành phần khác trong nhóm.
* **android:layout\_alignParentTop** căn chỉnh phần tử với mép trên của **RelativeLayout** cha.
* **android:layout\_alignParentLeft** căn chỉnh phần tử với mép trái của **RelativeLayout** cha.
* **android:layout\_alignParentStart** giúp cạnh bắt đầu của phần tử khớp với cạnh bắt đầu của phần tử cha, hỗ trợ bố cục phù hợp với ngôn ngữ hiển thị. Nếu thiết bị sử dụng ngôn ngữ **trái sang phải (LTR)**, "start" là mép trái màn hình. Nếu sử dụng **phải sang trái (RTL)**, "start" là mép phải màn hình.

**Bài tập về nhà**

**Thay đổi ứng dụng**

Mở ứng dụng **HelloToast** và thực hiện các thay đổi sau:

1. Đổi tên dự án thành **HelloConstraint** và refactor dự án theo tên mới. (Xem **Phụ lục: Tiện ích** để biết hướng dẫn sao chép và refactor dự án).
2. Chỉnh sửa bố cục trong **activity\_main.xml** để căn chỉnh các nút **Toast** và **Count** dọc theo mép trái của **TextView show\_count**, hiển thị số "0". Xem hình minh họa bên dưới để tham khảo.
3. Thêm một nút **Zero** ở giữa hai nút **Toast** và **Count**.
4. Phân bổ đều các nút **Button** theo chiều dọc từ trên xuống dưới trong **TextView show\_count**.
5. Đặt nền ban đầu của nút **Zero** là màu xám.
6. Đảm bảo nút **Zero** xuất hiện trong bố cục chế độ ngang (**activity\_main.xml (land)**) và trên màn hình máy tính bảng (**activity\_main.xml (xlarge)**).
7. Khi nhấn vào nút **Zero**, giá trị hiển thị trong **TextView show\_count** sẽ trở về **0**.
8. Cập nhật trình xử lý sự kiện cho nút **Count** sao cho khi nhấn, nền của nút thay đổi tùy theo giá trị **count** là số chẵn hay lẻ.
   * **Gợi ý**: Không sử dụng **findViewById** để lấy tham chiếu đến nút **Count**. Hãy tìm cách khác phù hợp hơn.
   * Có thể sử dụng hằng số trong lớp **Color** để đặt màu nền cho số chẵn và lẻ.
9. Khi nhấn vào nút **Count**, nền của nút **Zero** phải thay đổi thành một màu khác ngoài màu xám, để hiển thị rằng nó đã được kích hoạt.
   * **Gợi ý**: Trong trường hợp này, có thể sử dụng **findViewById**.
10. Khi nhấn vào nút **Zero**, màu nền của nó sẽ trở lại màu xám nếu giá trị **count** bằng **0**.

### 1.3)Văn bản và chế độ xem cuộn

Giới thiệu

Lớp TextView là lớp con của lớp View hiển thị văn bản trên màn hình. Bạn có thể kiểm soát cách văn bản xuất hiện bằng các thuộc tính TextView trong tệp bố cục XML. Bài thực hành này cho thấy cách làm việc với nhiều phần tử TextView, bao gồm một phần tử mà người dùng có thể cuộn nội dung của nó theo chiều dọc.

Nếu bạn có nhiều thông tin hơn mức hiển thị trên màn hình của thiết bị, bạn có thể tạo chế độ xem cuộn để người dùng có thể cuộn theo chiều dọc bằng cách vuốt lên hoặc xuống hoặc theo chiều ngang bằng cách vuốt sang phải hoặc sang trái.

Thông thường, bạn sẽ sử dụng chế độ xem cuộn cho các tin tức, bài viết hoặc bất kỳ văn bản dài nào không hoàn toàn vừa trên màn hình. Bạn cũng có thể sử dụng chế độ xem cuộn để cho phép người dùng nhập nhiều dòng văn bản hoặc kết hợp các phần tử UI (chẳng hạn như trường văn bản và nút) trong chế độ xem cuộn.

Lớp ScrollView cung cấp bố cục cho chế độ xem cuộn. ScrollView là một lớp con của FrameLayout. Bên trong nó, bạn chỉ nên đặt một phần tử con duy nhất, phần tử này sẽ chứa toàn bộ nội dung cần cuộn. Phần tử con này có thể là một ViewGroup (chẳng hạn như LinearLayout) chứa các thành phần giao diện người dùng.

Các bố cục phức tạp có thể gặp vấn đề về hiệu suất khi chứa các phần tử con như hình ảnh. Một lựa chọn tốt cho View bên trong ScrollView là LinearLayout được sắp xếp theo hướng dọc, giúp hiển thị các mục mà người dùng có thể cuộn qua (chẳng hạn như các phần tử TextView).

Với ScrollView, tất cả các phần tử giao diện người dùng đều được lưu trữ trong bộ nhớ và nằm trong cây giao diện, ngay cả khi chúng không được hiển thị trên màn hình. Điều này làm cho ScrollView trở thành một lựa chọn lý tưởng để cuộn các trang văn bản tự do một cách mượt mà, vì toàn bộ văn bản đã được tải vào bộ nhớ. Tuy nhiên, ScrollView có thể tiêu tốn nhiều bộ nhớ, ảnh hưởng đến hiệu suất của ứng dụng.

Nếu bạn cần hiển thị danh sách dài các mục mà người dùng có thể thêm, xóa hoặc chỉnh sửa, hãy cân nhắc sử dụng RecyclerView, được giới thiệu trong một bài học riêng.

**Những kiến thức 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.

● Triển khai TextView trong bố cục của ứng dụng.

● Tạo và sử dụng tài nguyên chuỗi (string resources).

**Những gì bạn sẽ học**

● Cách sử dụng mã XML để thêm nhiều phần tử TextView.

● Cách sử dụng mã XML để định nghĩa một View có thể cuộn.

● Cách hiển thị văn bản tự do với một số thẻ định dạng HTML.

● Cách tùy chỉnh màu nền và màu chữ của TextView.

● Cách chèn liên kết web vào trong văn bản.

**Những gì bạn sẽ làm**

● Tạo ứng dụng ScrollingText.

● Thay đổi ViewGroup ConstraintLayout thành RelativeLayout.

● Thêm hai phần tử TextView cho tiêu đề và tiêu đề phụ của bài viết.

● Sử dụng kiểu TextAppearance và màu sắc cho tiêu đề và tiêu đề phụ.

● Sử dụng thẻ HTML trong chuỗi văn bản để định dạng nội dung.

● Dùng thuộc tính lineSpacingExtra để tăng khoảng cách dòng, giúp cải thiện khả năng đọc.

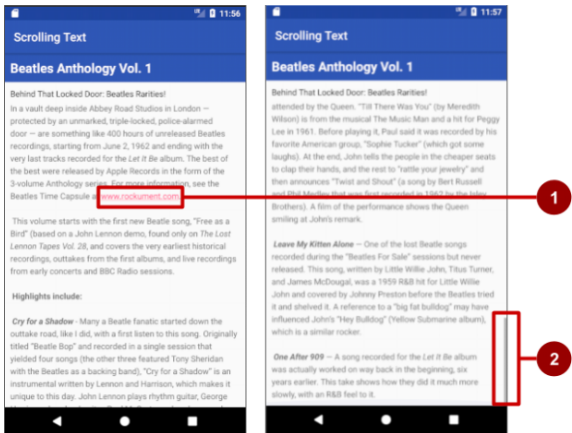
● Thêm ScrollView vào bố cục để cho phép cuộn nội dung trong TextView.

● Thêm thuộc tính autoLink để kích hoạt các URL trong văn bản, giúp chúng có thể nhấp vào.

**Tổng quan về ứng dụng**

Ứng dụng Scrolling Text minh họa cách sử dụng thành phần giao diện ScrollView. ScrollView là một ViewGroup, trong ví dụ này, nó chứa một TextView. Ứng dụng hiển thị một trang văn bản dài — trong trường hợp này là bài đánh giá album nhạc — mà người dùng có thể cuộn dọc để đọc bằng cách vuốt lên và xuống. Một thanh cuộn sẽ xuất hiện ở lề phải.

Ứng dụng này cũng cho thấy cách bạn có thể sử dụng văn bản được định dạng với các thẻ HTML đơn giản để tạo chữ đậm, nghiêng và sử dụng ký tự xuống dòng để tách đoạn. Ngoài ra, bạn có thể chèn các liên kết web có thể nhấp vào trong văn bản.



Trong hình trên, các thành phần sau xuất hiện:

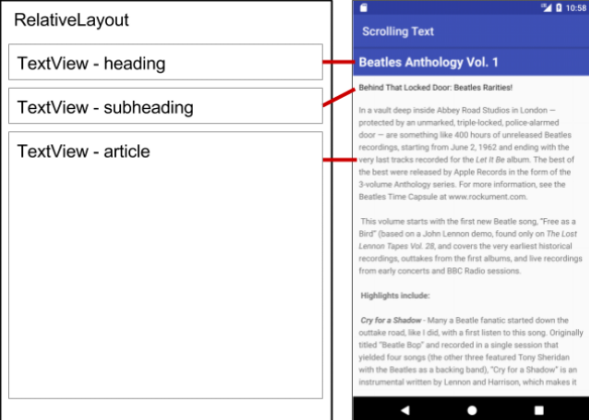
1.Một liên kết web có thể nhấp được nhúng trong văn bản tự do.

2.Thanh cuộn xuất hiện khi cuộn văn bản.

**Nhiệm vụ 1: Thêm và chỉnh sửa các phần tử TextView**

Trong bài thực hành này, bạn sẽ tạo một dự án Android cho ứng dụng ScrollingText, thêm các phần tử TextView vào bố cục để hiển thị tiêu đề và tiêu đề phụ của bài viết. Đồng thời, bạn sẽ thay đổi phần tử TextView mặc định "Hello World" để hiển thị một bài viết dài.

Hình dưới đây minh họa sơ đồ bố cục của ứng dụng.



Bạn sẽ thực hiện tất cả các thay đổi này trong mã XML và trong tệp strings.xml. Bạn sẽ chỉnh sửa mã XML cho bố cục trong Text pane bằng cách nhấp vào tab Text, thay vì sử dụng tab Design trong Design pane.

Một số thay đổi đối với các phần tử giao diện người dùng và thuộc tính có thể dễ dàng thực hiện trực tiếp trong Text pane bằng cách chỉnh sửa mã nguồn XML.

**1.1 Tạo dự án và các phần tử TextView**

Trong nhiệm vụ này, bạn sẽ tạo dự án và các phần tử **TextView**, đồng thời sử dụng các thuộc tính của **TextView** để định dạng văn bản và nền.

**Mẹo: Để tìm hiểu thêm về các thuộc tính này, hãy tham khảo tài liệu TextView.**

1. Trong **Android Studio**, tạo một dự án mới với các thông số sau:

| **Thuộc tính** | **Giá trị** |
| --- | --- |
| **Application Name** | Scrolling Text |
| **Company Name** | android.example.com (hoặc tên miền của bạn) |
| **Phone and Tablet Minimum SDK** | API 15: Android 4.0.3 IceCreamSandwich |
| **Template** | Empty Activity |
| **Generate Layout File** | Được chọn |
| **Backwards Compatibility (AppCompat)** | Được chọn |

1. Trong **Project > Android pane**, điều hướng đến thư mục **app > res > layout**, mở tệp **activity\_main.xml**, sau đó nhấp vào tab **Text** để xem mã XML.

Ở phần đầu, hay còn gọi là **gốc** của cây **View hierarchy**, là **ViewGroup ConstraintLayout**:

android.support.constraint.ConstraintLayout

1. Thay đổi ViewGroup này thành RelativeLayout. Dòng thứ hai của mã bây giờ sẽ trông giống như sau:



RelativeLayout cho phép bạn sắp xếp các phần tử giao diện người dùng tương đối với nhau hoặc tương đối với chính RelativeLayout cha.

Phần tử TextView mặc định "Hello World" được tạo bởi mẫu Empty Layout vẫn có các thuộc tính ràng buộc (constraint attributes), chẳng hạn như:app:layout\_constraintBottom\_toBottomOf="parent".Đừng lo lắng—bạn sẽ xóa chúng ở bước tiếp theo.

1. Xóa dòng mã XML sau, vì nó liên quan đến **ConstraintLayout**:



1. Thêm một phần tử **TextView** phía trên **TextView "Hello World"** bằng cách nhập:<TextView. Một khối **TextView** sẽ xuất hiện với dấu **/>** ở cuối, hiển thị các thuộc tính **layout\_width** và **layout\_height**, đây là các thuộc tính bắt buộc cho **TextView**.
2. Nhập các thuộc tính sau cho **TextView**. Khi bạn nhập từng thuộc tính và giá trị, Android Studio sẽ hiển thị các gợi ý để hoàn thành tên thuộc tính hoặc giá trị.

**TextView #1 - Thuộc tính và Giá trị**

| **Thuộc tính** | **Giá trị** |
| --- | --- |
| **android:layout\_width** | "match\_parent" (Chiều rộng toàn màn hình) |
| **android:layout\_height** | "wrap\_content" (Chiều cao phù hợp với nội dung) |
| **android:id** | "@+id/article\_heading" (ID của TextView) |
| **android:background** | "@color/colorPrimary" (Màu nền) |
| **android:textColor** | "@android:color/white" (Màu chữ trắng) |
| **android:padding** | "10dp" (Khoảng đệm xung quanh) |
| **android:textAppearance** | "@android:style/TextAppearance.DeviceDefault.Large" (Kiểu chữ lớn) |
| **android:textStyle** | "bold" (Chữ in đậm) |
| **android:text** | "Article Title" (Tiêu đề bài viết) |

7.Trích xuất tài nguyên chuỗi cho thuộc tính android:text với chuỗi cố định "Article Title" trong TextView để tạo một mục trong strings.xml.

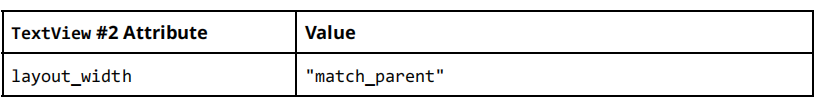
Đặt con trỏ vào chuỗi cố định, nhấn Alt-Enter (hoặc Option-Enter trên Mac), sau đó chọn Extract string resource. Đảm bảo tùy chọn Create the resource in directories được chọn, rồi chỉnh sửa tên tài nguyên thành article\_title.

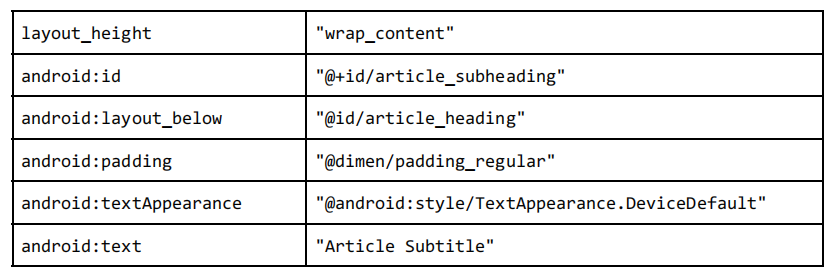
Tài nguyên chuỗi (String resources) được mô tả chi tiết trong String Resources..

8.Trích xuất tài nguyên kích thước (dimension resource) cho thuộc tính android:padding với giá trị cố định "10dp" trong TextView để tạo tệp dimens.xml và thêm một mục vào đó.

Đặt con trỏ vào chuỗi cố định, nhấn Alt-Enter (hoặc Option-Enter trên Mac), sau đó chọn Extract dimension resource. Đảm bảo tùy chọn Create the resource in directories được chọn, rồi chỉnh sửa tên tài nguyên thành padding\_regular.

9.Thêm một phần tử TextView khác phía trên TextView "Hello World" và bên dưới TextView bạn đã tạo ở các bước trước. Thêm các thuộc tính sau vào TextView:

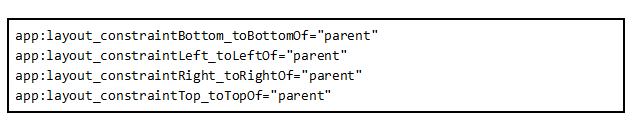




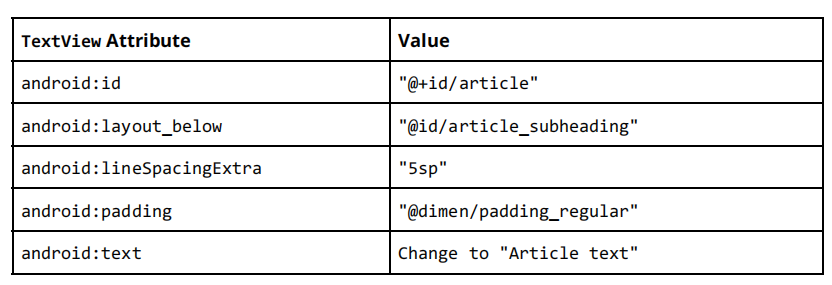
Vì bạn đã trích xuất tài nguyên kích thước cho chuỗi "10dp" thành **padding\_regular** trong **TextView** đã tạo trước đó, bạn có thể sử dụng @dimen/padding\_regular cho thuộc tính android:padding trong **TextView** này.

8.Trích xuất tài nguyên chuỗi cho thuộc tính android:text với chuỗi cố định "Article Subtitle" trong **TextView** thành **article\_subtitle**.

9.Trong phần tử **TextView "Hello World"**, xóa các thuộc tính **layout\_constraint**.



1. Thêm các thuộc tính **TextView** sau vào phần tử **TextView "Hello World"**, và thay đổi thuộc tính **android:text**.



1. Trích xuất tài nguyên chuỗi cho "Article text" thành **article\_text**, và trích xuất tài nguyên kích thước cho "5sp" thành **line\_spacing**.
2. Định dạng lại và căn chỉnh mã bằng cách chọn **Code > Reformat Code**. Việc định dạng và căn chỉnh mã giúp dễ đọc và dễ hiểu hơn.

1.2 Thêm nội dung văn bản của bài viết

Trong một ứng dụng thực tế truy cập các bài báo từ tạp chí hoặc báo, nội dung bài viết có thể đến từ một nguồn trực tuyến thông qua content provider, hoặc được lưu sẵn trong cơ sở dữ liệu trên thiết bị.

Trong bài thực hành này, bạn sẽ tạo bài viết dưới dạng một chuỗi dài duy nhất trong tài nguyên strings.xml.

Các bước thực hiện:

1.Trong thư mục app > res > values, mở tệp strings.xml.

2.Mở một tệp văn bản có nội dung dài hoặc mở strings.xml của ứng dụng ScrollingText đã hoàn thành.

3.Nhập giá trị cho các chuỗi article\_title và article\_subtitle với tiêu đề và phụ đề tùy chọn, hoặc sử dụng giá trị từ strings.xml của ứng dụng ScrollingText đã hoàn thành. Đảm bảo rằng các giá trị chuỗi này là văn bản một dòng, không chứa thẻ HTML hoặc xuống dòng.

4.Nhập hoặc sao chép nội dung văn bản vào chuỗi article\_text. Bạn có thể sử dụng nội dung từ tệp văn bản của mình hoặc sử dụng nội dung có sẵn trong strings.xml của ứng dụng ScrollingText đã hoàn thành. Yêu cầu duy nhất là văn bản phải đủ dài để không hiển thị hết trên màn hình.

Lưu ý:

Khi nhập hoặc dán văn bản vào strings.xml, các dòng văn bản sẽ không tự động xuống dòng mà sẽ kéo dài sang lề phải. Đây là hành vi đúng—mỗi dòng văn bản bắt đầu từ lề trái sẽ đại diện cho một đoạn văn. Nếu muốn xuống dòng, bạn có thể nhấn Return để nhập dấu ngắt dòng cứng hoặc định dạng văn bản trước bằng trình chỉnh sửa văn bản.

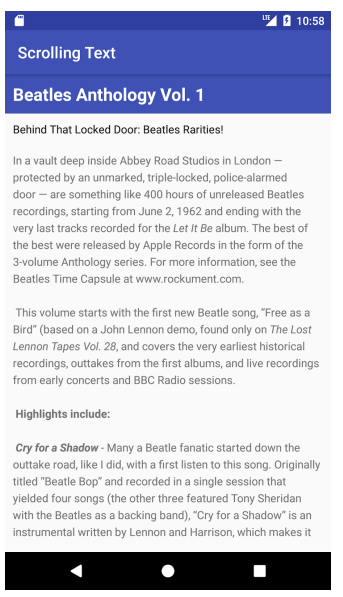
Sử dụng \n để biểu thị dấu kết thúc dòng và nhập thêm \n để tạo một dòng trống giữa các đoạn văn. Điều này giúp giữ định dạng văn bản và tránh các đoạn văn bị dính vào nhau.

* Nếu văn bản chứa dấu nháy đơn ('), bạn phải thêm dấu gạch chéo ngược (\') trước nó để tránh lỗi. Tương tự, nếu có dấu nháy kép ("), bạn cũng cần thoát ký tự bằng cách thêm \". Ngoài ra, các ký tự không thuộc bảng mã ASCII cũng cần được thoát. Xem phần **Định dạng và tạo kiểu** trong **String resources** để biết thêm chi tiết.
* Sử dụng thẻ HTML <b> và </b> để làm đậm các từ cần nhấn mạnh.
* Sử dụng thẻ HTML <i> và </i> để làm nghiêng văn bản. Nếu sử dụng dấu nháy đơn kiểu cong trong một cụm chữ in nghiêng, hãy thay thế chúng bằng dấu nháy đơn thẳng.
* Có thể kết hợp cả chữ đậm và chữ nghiêng bằng cách lồng các thẻ, ví dụ: <b><i>... từ cần định dạng ...</i></b>. Các thẻ HTML khác sẽ bị bỏ qua.
* Bao bọc toàn bộ nội dung bài viết trong thẻ <string name="article\_text"> ... </string> trong tệp **strings.xml**.
* Chèn một liên kết web để kiểm tra, ví dụ: www.google.com. (Ví dụ dưới đây sử dụng www.rockument.com.)
* Không sử dụng thẻ HTML, vì tất cả các thẻ HTML khác ngoài <b> và <i> sẽ bị bỏ qua và hiển thị dưới dạng văn bản, điều này không đúng với mục đích của bạn.



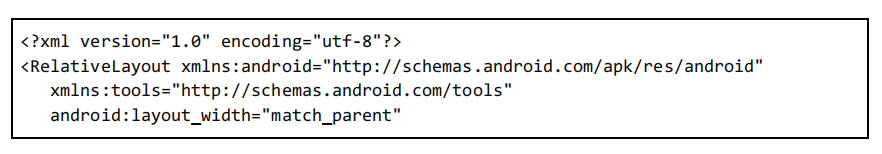
**1.3 Chạy ứng dụng**

Chạy ứng dụng. Bài viết sẽ hiển thị, nhưng người dùng chưa thể cuộn vì bạn vẫn chưa thêm ScrollView (bạn sẽ làm điều này trong nhiệm vụ tiếp theo). Ngoài ra, khi nhấn vào liên kết web, nó vẫn chưa hoạt động. Bạn cũng sẽ khắc phục điều này trong nhiệm vụ tiếp theo.

****

**Giải pháp mã nguồn cho Task 1**

Tệp layout activity\_main.xml trông như sau:





**Nhiệm vụ 2: Thêm ScrollView và liên kết web có thể nhấp**

Trong nhiệm vụ trước, bạn đã tạo ứng dụng ScrollingText với các phần tử TextView cho tiêu đề bài viết, phụ đề và nội dung bài viết dài. Bạn cũng đã thêm một liên kết web, nhưng liên kết đó chưa hoạt động. Bạn sẽ thêm mã để kích hoạt liên kết này.

Ngoài ra, TextView tự nó không thể cho phép người dùng cuộn nội dung bài viết để xem toàn bộ. Bạn sẽ thêm một ViewGroup mới có tên là ScrollView vào bố cục XML để làm cho TextView có thể cuộn được.

**2.1 Thêm thuộc tính autoLink để kích hoạt liên kết web**

Thêm thuộc tính android:autoLink="web" vào TextView của bài viết. Mã XML cho TextView này bây giờ sẽ trông như sau:

<TextView

android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:id="@+id/article" android:autoLink="web" android:layout\_below="@id/article\_subheading" android:lineSpacingExtra="@dimen/line\_spacing" android:padding="@dimen/padding\_regular" android:text="@string/article\_text" />

2.2 **Thêm một ScrollView vào bố cục.**

Để làm cho một View (chẳng hạn như TextView) có thể cuộn được, hãy nhúng View đó vào bên trong một ScrollView.

1. Thêm một ScrollView giữa TextView của article\_subheading và TextView của article. Khi bạn nhập <ScrollView, Android Studio sẽ tự động thêm </ScrollView> ở cuối và hiển thị các gợi ý cho thuộc tính android:layout\_width và android:layout\_height.
2. Chọn wrap\_content từ các gợi ý cho cả hai thuộc tính.

Mã XML cho hai TextView và ScrollView hiện trông như sau:



3.Di chuyển thẻ đóng </ScrollView> xuống sau TextView của article để tất cả các thuộc tính của TextView này nằm hoàn toàn bên trong ScrollView.

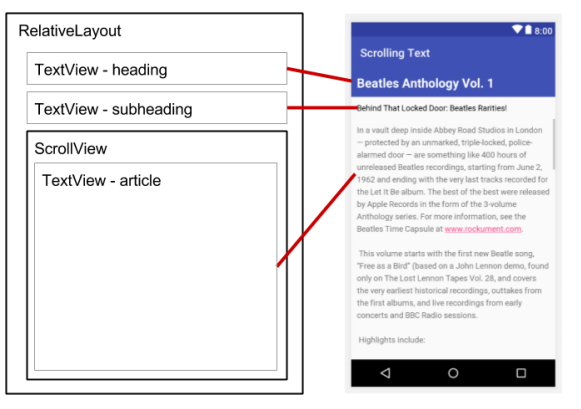
4.Xóa thuộc tính sau khỏi TextView của article và thêm nó vào ScrollView:



Với thuộc tính trên, phần tử ScrollView sẽ xuất hiện bên dưới phần phụ đề của bài viết. Bài viết sẽ nằm bên trong phần tử ScrollView.

1. Chọn Code > Reformat Code để định dạng lại mã XML sao cho TextView của bài viết hiển thị thụt vào bên trong thẻ <ScrollView>.
2. Nhấp vào tab Preview ở phía bên phải của trình chỉnh sửa layout để xem bản xem trước của bố cục.

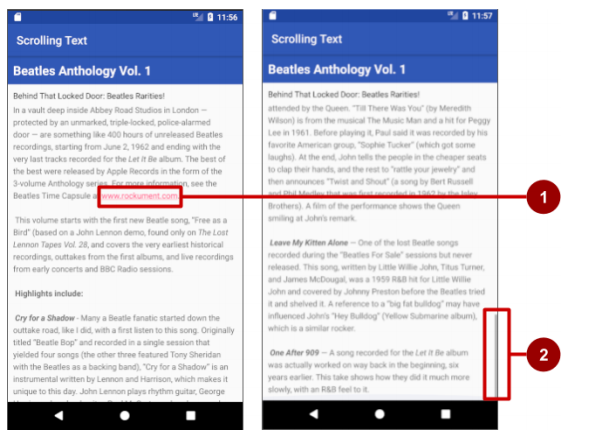
Bố cục bây giờ sẽ trông giống như phần bên phải của hình minh họa sau.



2.3 Chạy ứng dụng

Để kiểm tra cách văn bản cuộn:

1. Chạy ứng dụng trên thiết bị hoặc trình giả lập.  
   Vuốt lên và xuống để cuộn bài viết. Thanh cuộn xuất hiện ở lề phải khi bạn cuộn.  
   Nhấn vào liên kết web để truy cập trang web. Thuộc tính android:autoLink tự động biến bất kỳ URL nào có thể nhận diện trong TextView (chẳng hạn như [www.rockument.com](http://www.rockument.com/)) thành một liên kết web.
2. Xoay thiết bị hoặc trình giả lập trong khi ứng dụng đang chạy. Lưu ý rằng vùng cuộn mở rộng để sử dụng toàn bộ màn hình và vẫn cuộn đúng cách.
3. Chạy ứng dụng trên máy tính bảng hoặc trình giả lập máy tính bảng. Lưu ý rằng vùng cuộn mở rộng để sử dụng toàn bộ màn hình và vẫn cuộn đúng cách.



Trong hình trên, xuất hiện các yếu tố sau:

1. Một liên kết web đang hoạt động được nhúng trong văn bản tự do.
2. Thanh cuộn xuất hiện khi cuộn văn bản.

**Nhiệm vụ 2 mã giải pháp**

Mã XML cho bố cục với chế độ cuộn như sau:

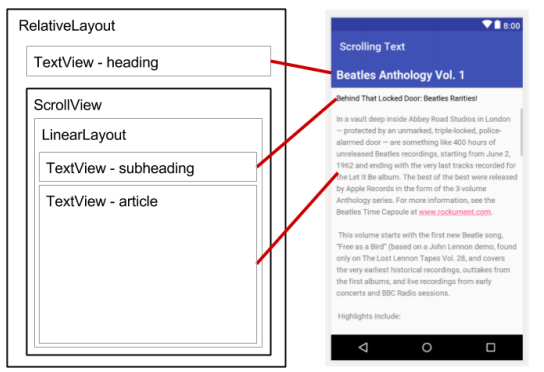




**Nhiệm vụ 3: Cuộn nhiều phần tử**

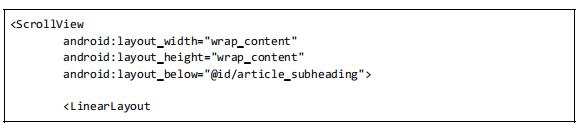
Như đã lưu ý trước đó, một ScrollView chỉ có thể chứa một View con (chẳng hạn như TextView của bài viết mà bạn đã tạo). Tuy nhiên, View đó có thể là một ViewGroup khác chứa các phần tử View, chẳng hạn như LinearLayout. Bạn có thể lồng một ViewGroup như LinearLayout vào trong ScrollView, do đó có thể cuộn tất cả nội dung bên trong LinearLayout.

Ví dụ, nếu bạn muốn phần tiêu đề phụ của bài viết cuộn cùng với bài viết, hãy thêm một LinearLayout vào trong ScrollView, rồi di chuyển tiêu đề phụ và bài viết vào trong LinearLayout. Khi đó, LinearLayout trở thành phần tử View con duy nhất trong ScrollView, như được minh họa trong hình dưới đây, và người dùng có thể cuộn toàn bộ LinearLayout: bao gồm cả tiêu đề phụ và bài viết.



**3.1 Thêm LinearLayout vào ScrollView**

1. Mở tệp activity\_main.xml của dự án ScrollingText app và chọn tab Text để chỉnh sửa mã XML (nếu chưa được chọn).
2. Thêm một LinearLayout phía trên TextView của bài viết trong ScrollView. Khi bạn nhập <LinearLayout, Android Studio sẽ tự động thêm </LinearLayout> vào cuối và hiển thị các gợi ý cho thuộc tính android:layout\_width và android:layout\_height. Chọn match\_parent cho chiều rộng và wrap\_content cho chiều cao từ danh sách gợi ý. Mã XML ở phần đầu của ScrollView bây giờ sẽ trông như thế này:



Bạn sử dụng match\_parent để chiều rộng khớp với ViewGroup cha. Bạn sử dụng wrap\_content để LinearLayout có kích thước vừa đủ để bao bọc nội dung bên trong.

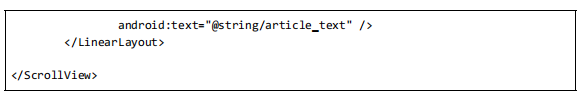
1. Di chuyển thẻ </LinearLayout> xuống sau thẻ article TextView nhưng trước thẻ đóng </ScrollView>.

LinearLayout hiện bao gồm TextView của bài viết và nằm hoàn toàn bên trong ScrollView.

1. Thêm thuộc tính android:orientation="vertical" vào LinearLayout để thiết lập hướng của nó theo chiều dọc..
2. Chọn Code > Reformat Code để thụt lề mã nguồn một cách chính xác.

LinearLayout bây giờ trông như thế này:



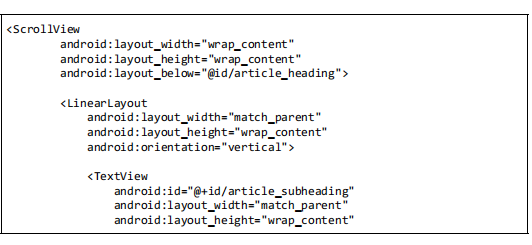


**3.2 Di chuyển các phần tử giao diện người dùng vào bên trong LinearLayout**

LinearLayout hiện tại chỉ có một phần tử giao diện người dùng—TextView của bài viết. Bạn cần đưa TextView của article\_subheading vào LinearLayout để cả hai có thể cuộn cùng nhau.

1. Để di chuyển TextView của article\_subheading, hãy chọn đoạn mã, chọn Chỉnh sửa > Cắt, nhấp vào phía trên TextView của bài viết bên trong LinearLayout, sau đó chọn Chỉnh sửa > Dán.
2. Xóa thuộc tính android:layout\_below="@id/article\_heading" khỏi TextView của article\_subheading. Vì TextView này hiện nằm trong LinearLayout, thuộc tính này sẽ xung đột với các thuộc tính của LinearLayout.
3. Thay đổi thuộc tính bố cục của ScrollView từ android:layout\_below="@id/article\_subheading" thành android:layout\_below="@id/article\_heading". Vì subheading hiện là một phần của LinearLayout, ScrollView phải được đặt bên dưới heading chứ không phải subheading.

Mã XML cho ScrollView hiện như sau:





1. Chạy ứng dụng.  
   Vuốt lên và xuống để cuộn bài viết, và lưu ý rằng subheading giờ đây cuộn cùng với bài viết, trong khi heading vẫn giữ nguyên vị trí.

**Mã nguồn giải pháp**

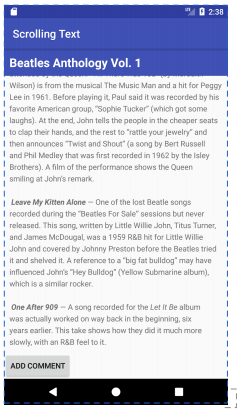
Dự án Android Studio: ScrollingText

Thử thách lập trình

Note: All coding challenges are optional and are not prerequisites for later lessons.

Thách thức: Thêm một thành phần UI khác—Nút—vào LinearLayout bên trong ScrollView để nó cuộn cùng với văn bản.

● Làm cho nút (Button) xuất hiện bên dưới bài viết. Người dùng cần cuộn đến cuối bài viết để nhìn thấy nút.  
● Sử dụng văn bản "Add Comment" cho nút. Trong thử thách này, không cần tạo phương thức xử lý nút; bạn chỉ cần đặt phần tử Button vào đúng vị trí trong bố cục.



**Mã giải pháp cho thử thách**

Dự án Android Studio: ScrollingTextChallenge

**Tóm tắt:**

* Sử dụng **ScrollView** để cuộn một View con duy nhất (chẳng hạn như **TextView**). **ScrollView** chỉ có thể chứa một View con hoặc một **ViewGroup**.
* Sử dụng một **ViewGroup** như **LinearLayout** làm View con bên trong **ScrollView** để cuộn nhiều phần tử View. Bao bọc các phần tử trong **LinearLayout**.
* Hiển thị văn bản tự do trong **TextView** với các thẻ định dạng HTML cho chữ **đậm** và *nghiêng*.
* Sử dụng \n làm ký tự xuống dòng trong văn bản tự do để ngăn các đoạn văn bị dính vào nhau.
* Dùng thuộc tính android:autoLink="web" để làm cho các liên kết web trong văn bản có thể nhấp vào.

**Các khái niệm liên quan.**

Tài liệu về các khái niệm liên quan có trong: 1.3 Text and scrolling views

**Tìm hiểu thêm**

Tài liệu hướng dẫn Android Studio:

* Android Studio download page
* Meet Android Studio

Tài liệu dành cho nhà phát triển Android.:

* ScrollView
* LinearLayout
* RelativeLayout
* View
* Button
* TextView
* String resources
* Relative Layout

Khác:

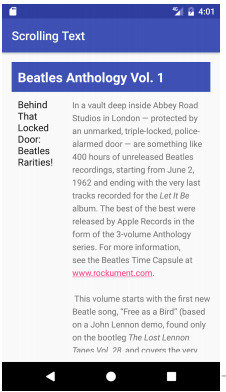
* Blog dành cho nhà phát triển Android. Linkify your Text!
* Codepath: Working with a TextView

Bài tập về nhà

**Thay đổi một ứng dụng.**

Mở ScrollingText2 app mà bạn đã tạo ra trong bài học Working with TextView Elements.

1. Thay đổi tiêu đề phụ để nó nằm gọn trong một cột bên trái rộng 100 dp, như hiển thị bên dưới.
2. Đặt văn bản của bài viết bên phải tiêu đề phụ như hình dưới đây.



Trả lời những câu hỏi này

Câu hỏi 1

Bạn có thể sử dụng bao nhiêu chế độ xem trong một ScrollView? Chọn một:

● Chỉ một view  
● Một view hoặc một ViewGroup  
● Bao nhiêu view tùy ý

Câu hỏi 2

Bạn sử dụng thuộc tính XML nào trong LinearLayout để hiển thị các chế độ xem cạnh nhau? Chọn một trong các tùy chọn sau:

* android:orientation="horizontal"
* android:orientation="vertical"
* android:layout\_width="wrap\_content"

Câu hỏi 3

Bạn sử dụng thuộc tính XML nào để xác định chiều rộng của LinearLayout bên trong chế độ xem cuộn? Chọn một:

* android:layout\_width="wrap\_content"
* android:layout\_width="match\_parent"
* android:layout\_width="200dp"

**Gửi ứng dụng của bạn để chấm điểm**

Hướng dẫn cho người chấm điểm

Kiểm tra xem ứng dụng có các tính năng sau không:

● Bố cục hiển thị tiêu đề phụ ở cột bên trái và văn bản bài viết ở cột bên phải,

như thể hiện trong hình trên.

● ScrollView bao gồm LinearLayout với hai phần tử TextView.

● Hướng LinearLayout được đặt thành ngang.

Bài 1.4: Học cách tự giúp mình

Giới thiệu

**Những gì bạn cần biết trước**

Bạn nên có khả năng:  
● Hiểu quy trình làm việc cơ bản của Android Studio.  
● Tạo một ứng dụng từ đầu bằng mẫu **Empty Activity**.  
● Sử dụng trình chỉnh sửa bố cục (**Layout Editor**).

**Những gì bạn sẽ học**

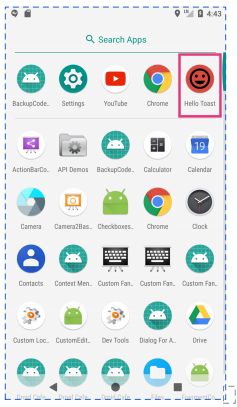
● Nơi tìm tài liệu và tài nguyên dành cho nhà phát triển.  
● Cách thêm biểu tượng khởi chạy (**launcher icon**) cho ứng dụng của bạn.  
● Cách tìm kiếm trợ giúp khi phát triển ứng dụng Android.

**Những gì bạn sẽ làm**

* Khám phá một số tài nguyên hữu ích dành cho nhà phát triển Android ở mọi cấp độ.
* Thêm biểu tượng khởi chạy (**launcher icon**) cho ứng dụng của bạn.

Tổng quan về ứng dụng

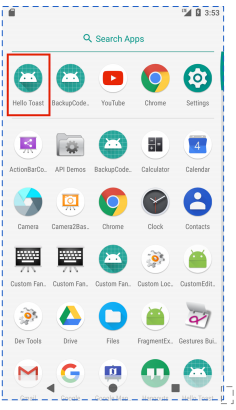
Bạn sẽ thêm biểu tượng khởi chạy vào ứng dụng **HelloToast** mà bạn đã tạo trước đó hoặc vào một ứng dụng mới.



**Nhiệm vụ 1: Thay đổi biểu tượng khởi chạy (launcher icon)**

Mỗi ứng dụng mới bạn tạo với Android Studio sẽ bắt đầu với một biểu tượng khởi chạy mặc định, đại diện cho ứng dụng. Biểu tượng khởi chạy xuất hiện trong danh sách ứng dụng trên Google Play. Khi người dùng tìm kiếm trên Google Play, biểu tượng của ứng dụng sẽ hiển thị trong kết quả tìm kiếm.

Khi người dùng đã cài đặt ứng dụng, biểu tượng khởi chạy sẽ xuất hiện trên thiết bị ở nhiều vị trí khác nhau, bao gồm **màn hình chính** và **màn hình tìm kiếm ứng dụng**. Ví dụ, ứng dụng **HelloToast** sẽ xuất hiện trong màn hình tìm kiếm ứng dụng của trình giả lập với biểu tượng mặc định dành cho các dự án ứng dụng mới, như hình dưới đây.



Thay đổi biểu tượng trình khởi chạy là một quy trình từng bước đơn giản giới thiệu cho bạn các tính năng tài sản hình ảnh của Android Studio. Trong nhiệm vụ này, bạn cũng tìm hiểu thêm về cách truy cập tài liệu Android chính thức.

**1.1 Khám phá tài liệu chính thức của Android**

Bạn có thể tìm thấy tài liệu chính thức dành cho nhà phát triển Android tại: developer.android.com. Tài liệu này chứa một lượng lớn thông tin và được Google cập nhật thường xuyên.

16. Đi đến developer.android.com/design/.

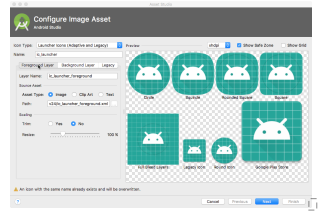
1. Phần này nói về Material Design, một triết lý thiết kế khái niệm phác thảo cách ứng dụng nên trông như thế nào và hoạt động ra sao trên thiết bị di động. Điều hướng các liên kết để tìm hiểu thêm về Material Design. Ví dụ, hãy truy cập phần Style để tìm hiểu thêm về cách sử dụng màu sắc và các chủ đề khác.
2. Đi đến developer.android.com/docs/ để tìm thông tin API, tài liệu tham khảo,hướng dẫn, hướng dẫn công cụ và mẫu mã.
3. Đi đến developer.android.com/distribute/ để tìm thông tin về việc đưa một ứng dụng vào Google

Play, Hệ thống phân phối kỹ thuật số của Google dành cho các ứng dụng được phát triển bằng Android SDK. Sử dụng Google Play Console để phát triển cơ sở người dùng của bạn và bắt đầu earning money.

1.2 Thêm nội dung hình ảnh cho biểu tượng trình khởi chạy

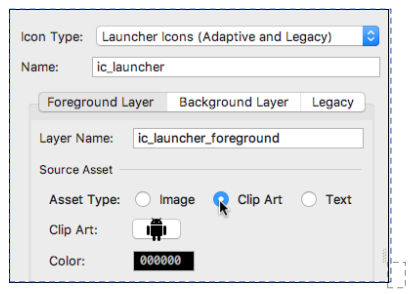
**Để thêm hình ảnh clip-art làm biểu tượng khởi chạy, hãy làm theo các bước sau:**

1. Mở dự án ứng dụng **HelloToast** từ bài học trước về trình chỉnh sửa bố cục hoặc tạo một dự án ứng dụng mới.
2. Trong **Project > Android**, nhấp chuột phải (hoặc Control-click) vào thư mục **res**, sau đó chọn **New > Image Asset**. Cửa sổ **Configure Image Asset** sẽ xuất hiện.



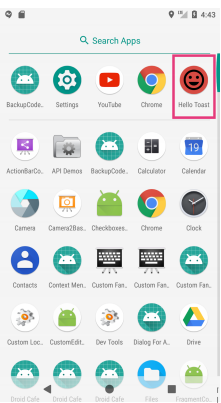
4.Trong trường **Icon Type**, chọn **Launcher Icons (Adaptive & Legacy)** nếu chưa được chọn.

5.Nhấp vào tab **Foreground Layer**, sau đó chọn **Clip Art** cho **Asset Type**.



1. Nhấp vào biểu tượng trong trường **Clip Art**. Các biểu tượng từ bộ biểu tượng **Material Design** sẽ xuất hiện.
2. Duyệt qua cửa sổ **Select Icon**, chọn một biểu tượng phù hợp (chẳng hạn như biểu tượng **mood** để gợi ý tâm trạng tốt), sau đó nhấp **OK**.
3. Nhấp vào tab **Background Layer**, chọn **Color** làm **Asset Type**, sau đó nhấp vào hộp màu để chọn màu làm nền cho biểu tượng.
4. Nhấp vào tab **Legacy** và xem lại các cài đặt mặc định. Xác nhận rằng bạn muốn tạo các biểu tượng **legacy, round**, và **Google Play Store**. Nhấp **Next** khi hoàn tất.
5. Chạy ứng dụng.

Android Studio tự động thêm các hình ảnh biểu tượng khởi chạy vào các thư mục **mipmap** cho các độ phân giải khác nhau. Do đó, sau khi bạn chạy ứng dụng, biểu tượng khởi chạy sẽ thay đổi thành biểu tượng mới, như hiển thị bên dưới.



**Mẹo: Xem** Launcher Icons để tìm hiểu thêm về cách thiết kế biểu tượng khởi chạy hiệu quả.

**Nhiệm vụ 2: Sử dụng mẫu dự án**

Android Studio cung cấp các mẫu cho các thiết kế ứng dụng và hoạt động (**activity**) phổ biến, được khuyến nghị. Sử dụng các mẫu có sẵn giúp tiết kiệm thời gian và giúp bạn tuân theo các phương pháp thiết kế tốt nhất.

Mỗi mẫu đều bao gồm một **activity** cơ bản và giao diện người dùng. Bạn đã từng sử dụng mẫu **Empty Activity**. Mẫu **Basic Activity** có nhiều tính năng hơn và tích hợp các tính năng ứng dụng được khuyến nghị, chẳng hạn như menu tùy chọn xuất hiện trên thanh ứng dụng (**app bar**).

**2.1 Khám phá kiến trúc của Basic Activity**

Mẫu **Basic** **Activity** là một mẫu linh hoạt do Android Studio cung cấp, giúp bạn khởi động nhanh quá trình phát triển ứng dụng.

Các bước thực hiện:

1. Trong **Android Studio,** tạo một dự án mới bằng mẫu Basic Activity.
2. **Build** và chạy ứng dụng.
3. Xác định các phần được đánh dấu trong hình và bảng bên dưới. Tìm các phần tương ứng trên thiết bị hoặc trình giả lập của bạn. Kiểm tra mã Java và tệp XML được mô tả trong bảng.

Việc làm quen với mã nguồn **Java** và tệp **XML** sẽ giúp bạn mở rộng và tùy chỉnh mẫu này theo nhu cầu của mình.

### Văn bản và các chế độ cuộn

### Tài nguyên có sẵn

## Activities

### Activity và Intent

### Vòng đời của Activity và trạng thái

### Intent ngầm định

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

### Trình gỡ lỗi

### Kiểm thử đơn vị

### Thư viện hỗ trợ

# TRẢI NGHIỆM NGƯỜI DÙNG

## Tương tác người dùng

### Hình ảnh có thể chọn

### Các điều khiển nhập liệu

### Menu và bộ chọn

### Điều hướng người dùng

### RecycleView

## Trải nghiệm người dùng thú vị

### Hình vẽ, định kiểu và chủ đề

### Thẻ và màu sắc

### Bố cục thích ứng

## Kiểm thử giao diện người dùng

### Espresso cho việc kiểm tra UI

# LÀM VIỆC TRONG NỀN

## Các tác vụ nền

### AsyncTask

### AsyncTask và AsyncTaskLoader

### Broadcast receivers

## Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền

### Thông báo

### Trình quản lý cảnh báo

### JobScheduler

# LƯU DỮ LIỆU NGƯỜI DÙNG

## Tùy chọn và cài đặt

### Shared preferences

### Cài đặt ứng dụng

## Lưu trữ dữ liệu với Room

### Room, LiveData và ViewModel

### Room, LiveData và ViewModel