

TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN



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

Hà Nội, 2.2025

MỤC LỤC

CHƯƠNG 1. Làm quen	3
Bài 1) Tạo ứng dụng đầu tiên.....	3
1.1) Android Studio và Hello World.....	3
1.2) Giao diện người dùng tương tác đầu tiên.....	5
1.3) Trình chỉnh sửa bố cục.....	11
1.4) Văn bản và các chế độ cuộn.....	11
1.5) Tài nguyên có sẵn.....	11
Bài 2) Activities	11
2.1) Activity và Intent.....	11
2.2) Vòng đời của Activity và trạng thái.....	11
2.3) Intent ngầm định.....	11
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ	11
3.1) Trình gỡ lỗi.....	11
3.2) Kiểm thử đơn vị.....	11
3.3) Thư viện hỗ trợ.....	11
CHƯƠNG 2. Trải nghiệm người dùng	12
Bài 1) Tương tác người dùng.....	12
1.1) Hình ảnh có thể chọn.....	12
1.2) Các điều khiển nhập liệu	12
1.3) Menu và bộ chọn.....	12
1.4) Điều hướng người dùng	12
1.5) RecycleView.....	12
Bài 2) Trải nghiệm người dùng thú vị.....	12
2.1) Hình vẽ, định kiểu và chủ đề.....	12
2.2) Thẻ và màu sắc	12
2.3) Bố cục thích ứng.....	12
Bài 3) Kiểm thử giao diện người dùng.....	12

3.1) Espresso cho việc kiểm tra UI.....	12
CHƯƠNG 3. Làm việc trong nền.....	12
Bài 1) Các tác vụ nền.....	12
1.1) AsyncTask.....	12
1.2) AsyncTask và AsyncTaskLoader.....	12
1.3) Broadcast receivers	12
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền.....	12
2.1) Thông báo.....	12
2.2) Trình quản lý cảnh báo.....	12
2.3) JobScheduler	12
CHƯƠNG 4. Lưu trữ dữ liệu người dùng.....	13
Bài 1) Tùy chọn và cài đặt.....	13
1.1) Shared preferences	13
1.2) Cài đặt ứng dụng	13
Bài 2) Lưu trữ dữ liệu với Room	13
2.1) Room, LiveData và ViewModel	13
2.2) Room, LiveData và ViewModel	13
3.1) Trinhf gowx loi

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

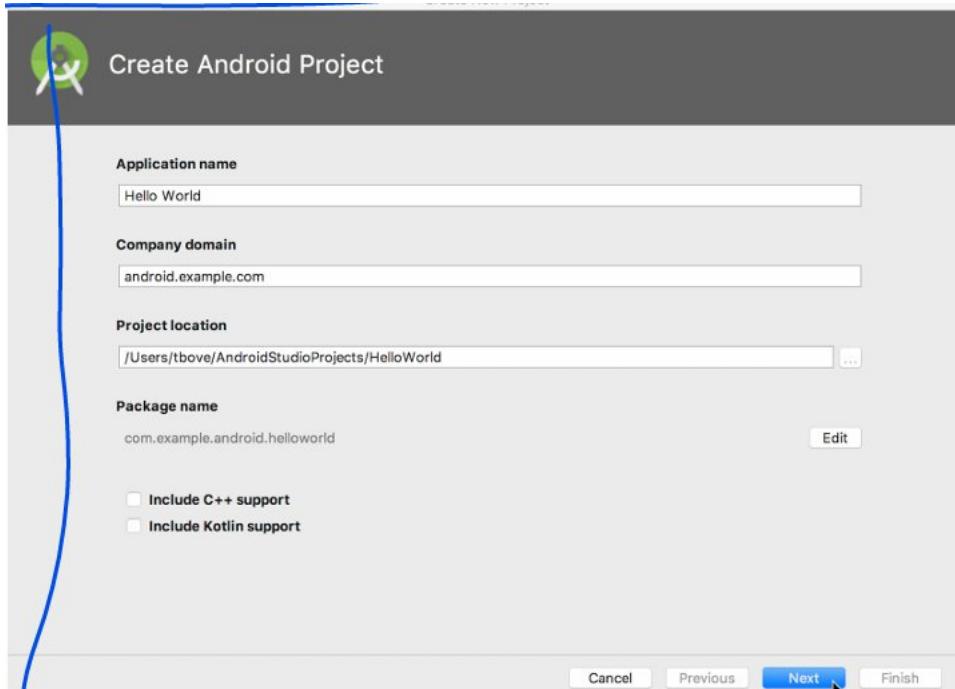
Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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



Những gì Bạn sẽ cần:

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

Những gì bạn sẽ học

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

Những gì bạn sẽ làm

- Cài đặt môi trường phát triển **Android Studio**.
- Tạo một trình giả lập (thiết bị ảo) để chạy ứng dụng của bạn trên máy tính.

- Tạo và chạy ứng dụng **Hello World** trên các thiết bị ảo và vật lý.
- Khám phá cấu trúc dự án.
- Tạo và xem các thông điệp ghi lại từ ứng dụng của bạn.
- Khám phá tệp **AndroidManifest.xml**

1.2) Giao diện người dùng tương tác đầu tiên

Giới thiệu

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

- **TextView** để hiển thị văn bản.
- **EditText** cho phép người dùng nhập và chỉnh sửa văn bản.
- **Button** và các phần tử có thể nhấp khác (chẳng hạn như **RadioButton**, **CheckBox** và **Spinner**) để cung cấp hành vi tương tác.
- **ScrollView** và **RecyclerView** để hiển thị các mục có thể cuộn.
- **ImageView** để hiển thị hình ảnh.
- **ConstraintLayout** và **LinearLayout** để chứa các phần tử View khác và định vị chúng.

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

Ví dụ, mã **MainActivity** trong ứng dụng **Hello World** hiển thị một bộ cục được định nghĩa trong tệp **activity_main.xml**, trong đó có một **TextView** với nội dung văn bản "Hello World".

Trong các ứng dụng phức tạp hơn, một **Activity** có thể thực hiện các hành động như phản hồi khi người dùng nhấn vào màn hình, vẽ nội dung đồ họa hoặc yêu cầu dữ liệu từ cơ sở dữ liệu hoặc internet. Bạn sẽ tìm hiểu thêm về lớp **Activity** trong bài học khác.

Trong bài thực hành này, bạn sẽ học cách tạo ứng dụng tương tác đầu tiên một ứng dụng cho phép người dùng tương tác. Bạn sẽ tạo một ứng dụng bằng mẫu **Empty Activity**. Ngoài ra, bạn cũng sẽ học cách sử dụng trình chỉnh sửa bố cục (**layout editor**) để thiết kế giao diện và chỉnh sửa bố cục.

Những gì bạn nên biết

Bạn nên làm quen với:

- Cách cài đặt và mở Android Studio
- Cách tạo ứng dụng Hello World
- Cách chạy ứng dụng Hello World

Những gì bạn sẽ học

- Cách tạo một ứng dụng có hành vi tương tác
- Cách sử dụng layout editor để thiết kế giao diện
- Cách chỉnh sửa bố cục trong XML
- Nhiều thuật ngữ mới

Những gì bạn sẽ làm

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

Tổng quan về ứng dụng

Ứng dụng **HelloToast** bao gồm hai phần tử **Button** và một **TextView**. Khi người dùng nhấp vào **Button** đầu tiên, một thông báo ngắn (**Toast**) sẽ hiển thị trên màn hình. Nhấp vào **Button** thứ hai sẽ tăng giá trị của bộ đếm số lần nhấp (**click counter**) hiển thị trong **TextView**, bắt đầu từ số **0**.

Dưới đây là giao diện của ứng dụng sau khi hoàn thành:

Nhiệm vụ 1: Tạo và khám phá một dự án mới

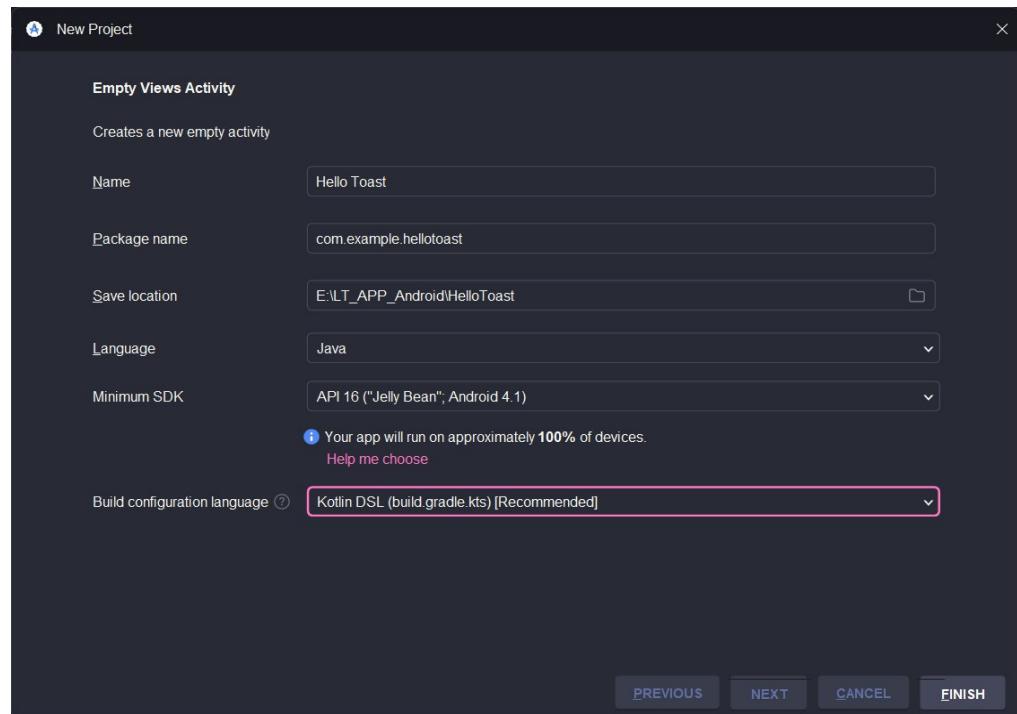
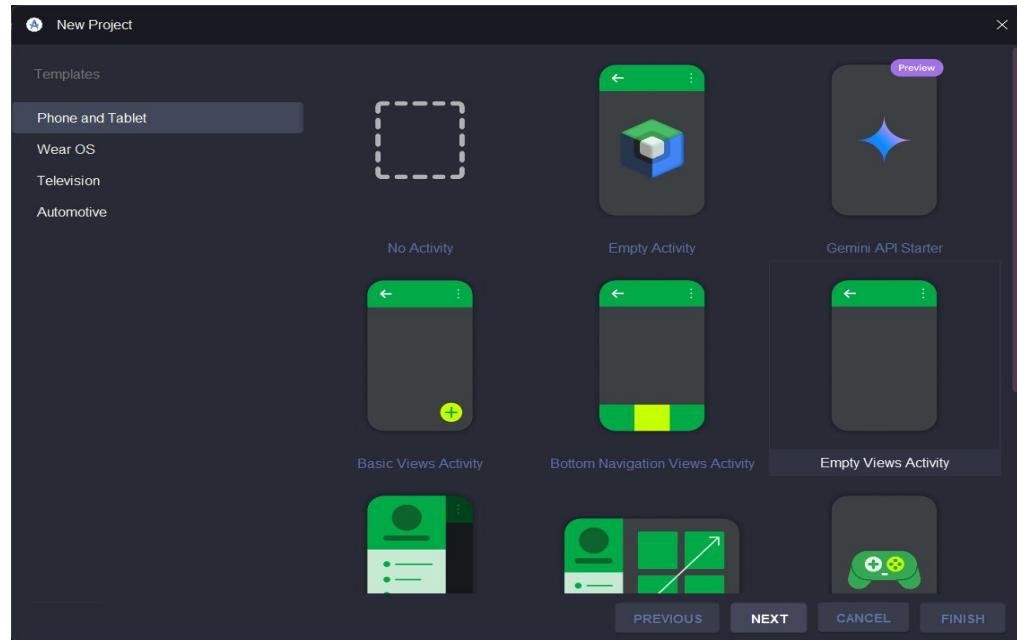
Trong bài thực hành này, bạn cần thiết kế và triển khai một dự án cho ứng dụng HelloToast.

1.1. Tạo dự án Android Studio

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

Attribute	Value
Application Name	Hello Toast
Company Name	com.example.android (or your own domain)
Phone and Tablet Minimum SDK	API15: Android 4.0.3 IceCreamSandwich
Template	Empty Activity
Generate Layout file box Selected	Selected
Backwards Compatibility box Selected	Selected

- Chọn Run > Run app hoặc nhấn và biểu tượng  trên thanh công cụ để biên dịch và chạy ứng dụng trên trình giả lập (**emulator**) hoặc thiết bị của bạn

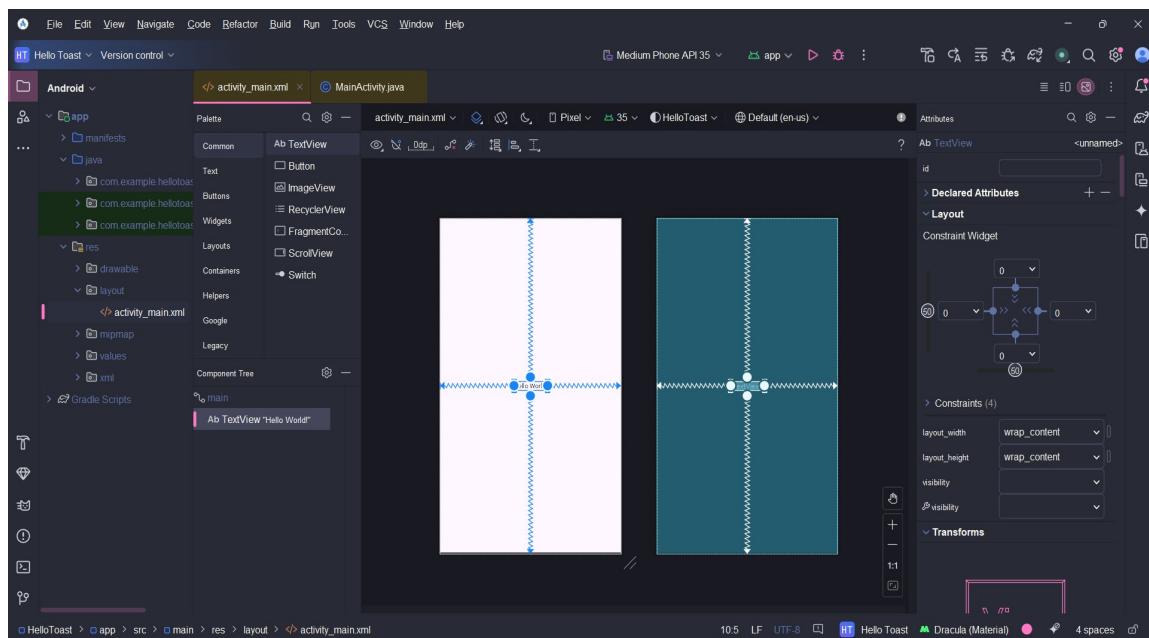


1.2. Khám phá trình chỉnh sửa bố cục

Android Studio cung cấp **trình chỉnh sửa bố cục (layout editor)** để nhanh chóng xây dựng giao diện người dùng (UI) của ứng dụng. Trình chỉnh sửa này cho phép

bạn kéo thả các phần tử vào chế độ xem thiết kế trực quan và bản vẽ (**blueprint view**), định vị chúng trong bố cục, thêm ràng buộc (**constraints**) và thiết lập thuộc tính. **Ràng buộc** xác định vị trí của một phần tử giao diện người dùng trong bố cục. Một **ràng buộc** thể hiện một kết nối hoặc cản chỉnh với một phần tử khác, bố cục cha, hoặc một đường hướng dẫn vô hình.

Khám phá trình chỉnh sửa bố cục và tham khảo hình minh họa bên dưới khi bạn thực hiện theo các bước được đánh số.



1. Trong bảng **Project > Android**, điều hướng đến 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 nó chưa được chọn. Bạn sử dụng tab **Design** để thao tác với các phần tử và bố cục, còn tab **Text** để chỉnh sửa mã XML của bố cục.
3. **Pane Palettes** hiển thị các phần tử giao diện người dùng (**UI elements**) mà bạn có thể sử dụng trong bố cục ứng dụng.
4. **Pane Component Tree** hiển thị hệ thống phân cấp của các phần tử giao diện người dùng. Các phần tử được tổ chức theo dạng cây, trong đó **con** kề thừa thuộc tính của **cha**. Trong hình minh họa, **TextView** là một phần tử con của **ConstraintLayout**. Bạn sẽ tìm hiểu thêm về các phần tử này trong bài học sau.
5. **Pane thiết kế và bản vẽ (Design & Blueprint panes)** trong trình chỉnh sửa bố cục 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 bảng thuộc tính (**Attributes pane**) để thiết lập các thuộc tính cho một phần tử UI.

Nhiệm vụ 2: Thêm vào các phần tử View trong chỉnh sửa bố cục

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

2.1. Kiểm tra các ràng buộc của phần tử

Thực hiện các bước sau:

1. Mở activity_main.xml từ ngăn Project > Android nếu tệp chưa được mở. Nếu tab Design chưa được chọn, hãy nhấp vào . Nếu không thấy Blueprint, nhấn vào nút Select Design Surface trên thanh công cụ và chọn Design + Blueprint.
2. Công cụ Autoconnect  cũng nằm trên thanh công cụ và được bật theo mặc định. Ở bước này, hãy đảm bảo công cụ không bị tắt.
3. Nhấn nút Zoom  để phóng to bảng thiết kế và bảng Blueprint để xem chi tiết hơn.
4. Chọn TextView trong ngăn Component Tree. TextView có nội dung "Hello World" sẽ được tô sáng trong bảng thiết kế và bảng Blueprint, đồng thời các ràng buộc (constraints) của phần tử cũng sẽ hiển thị.
5. Thực hiện theo hình động bên dưới cho bước này. Nhấp vào chấm tròn ở bên phải của TextView để xóa ràng buộc ngang đang liên kết nó với cạnh phải của bố cục. Khi đó, TextView sẽ dịch chuyển sang trái do không còn bị ràng buộc vào cạnh phải. Để thêm lại ràng buộc ngang, nhấp vào cùng một chấm tròn và kéo một đường đến cạnh phải của bố cục.

Trong bảng **Blueprint** hoặc **Design**, các điều khiển sau sẽ xuất hiện trên thẻ TextView:

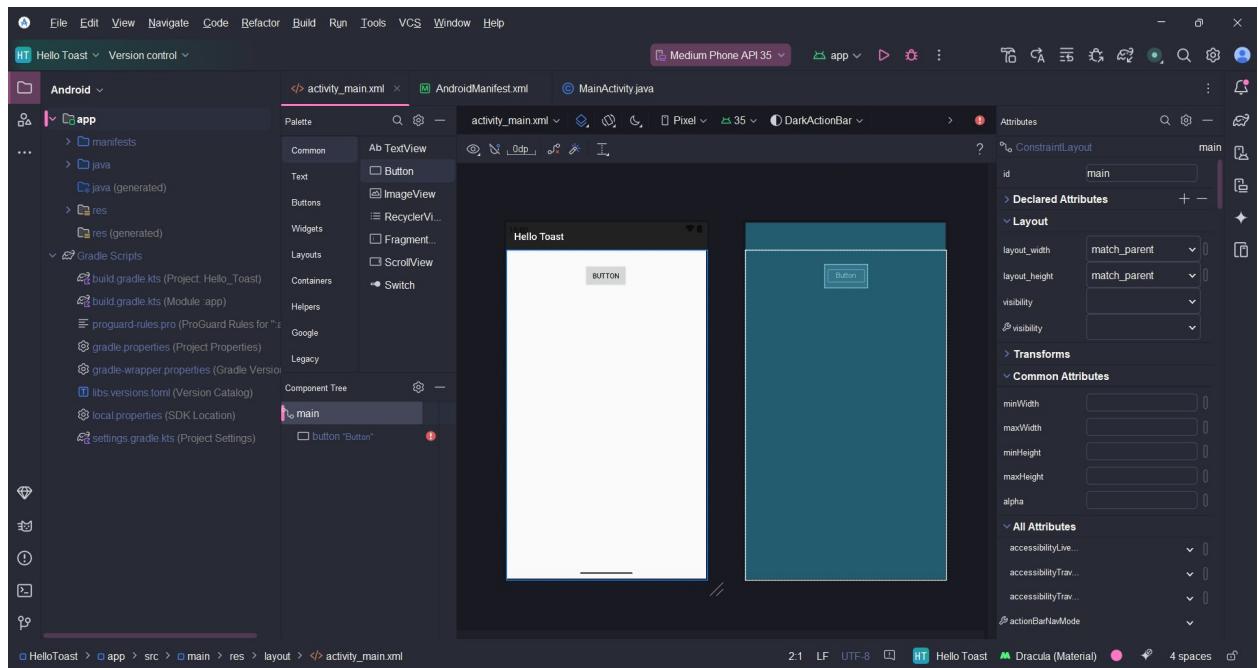
-  **Constraint handle:** Để tạo một ràng buộc như trong hình động trên, hãy nhấp vào một tay cầm ràng buộc, được hiển thị dưới dạng vòng tròn ở cạnh của một phần tử. Sau đó, kéo tay cầm này đến một tay cầm ràng buộc khác hoặc đến ranh giới của phần tử cha. Một đường ziczac sẽ đại diện cho ràng buộc đó.
-  **Resizing handle:** Để thay đổi kích thước phần tử, hãy kéo các tay cầm hình vuông. Khi bạn kéo, tay cầm sẽ chuyển thành một góc xiên.

2.2. Thêm một button vào bố cục

Khi được bật, công cụ **Autoconnect** sẽ tự động tạo hai hoặc nhiều ràng buộc cho một phần tử giao diện người dùng với bố cục cha. Sau khi bạn 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**:

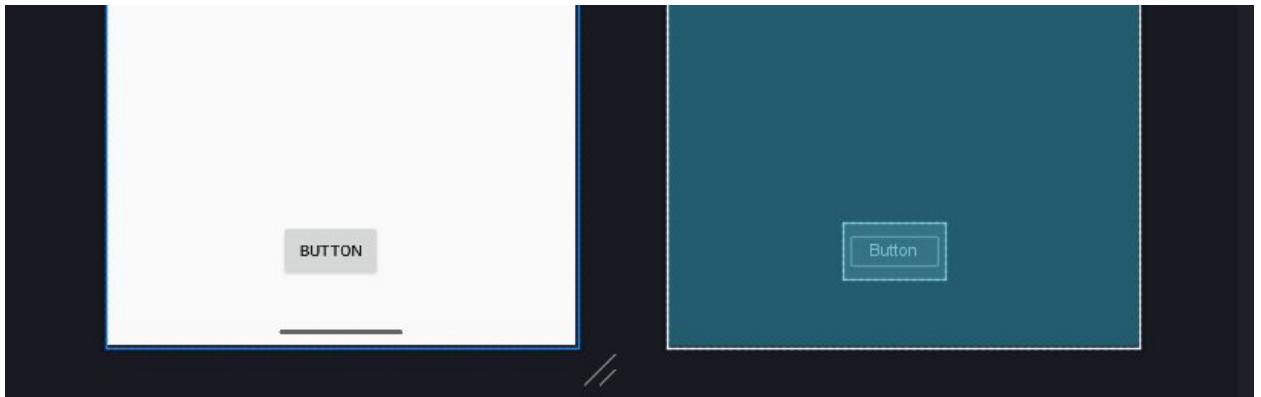
1. **Bắt đầu với một bố cục trống:** Phần tử **TextView** không cần thiết, vì vậy khi nó vẫn đang được chọn, nhấn phím **Delete** hoặc chọn **Edit > Delete**. Lúc này, bạn sẽ có một bố cục hoàn toàn trống.
2. Kéo một **Button** từ ngăn **Palette** vào bất kỳ vị trí nào trong bố cục. Nếu bạn thả **Button** vào khu vực giữa phía trên của bố cục, có thể các ràng buộc sẽ tự động xuất hiện. Nếu không, bạn có thể kéo các ràng buộc đến cạnh trên, cạnh trái và cạnh phải của bố cục như trong hình động bên dưới.



2.3. Thêm nút thứ 2 vào bố cục

1. Kéo một Button khác từ ngăn Palette vào giữa bố cục như trong hình động bên dưới. Autoconnect có thể tự động tạo các ràng buộc ngang cho bạn (nếu không, bạn có thể tự kéo chúng).

- Kéo một ràng buộc dọc từ Button xuống cạnh dưới của bố cục (tham khảo hình bên dưới).



Bạn có thể xóa ràng buộc khỏi 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, hãy nhấp vào tay cầm của ràng buộc đó.

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

Nhiệm vụ 3: Thay đổi thành phần UI

Ngăn **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện người dùng (**UI element**). Bạn có thể tìm thấy các thuộc tính (còn gọi là **properties**) chung cho tất cả các **View** trong tài liệu của lớp **View**.

Trong nhiệm vụ này, bạn sẽ nhập các 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** khác.

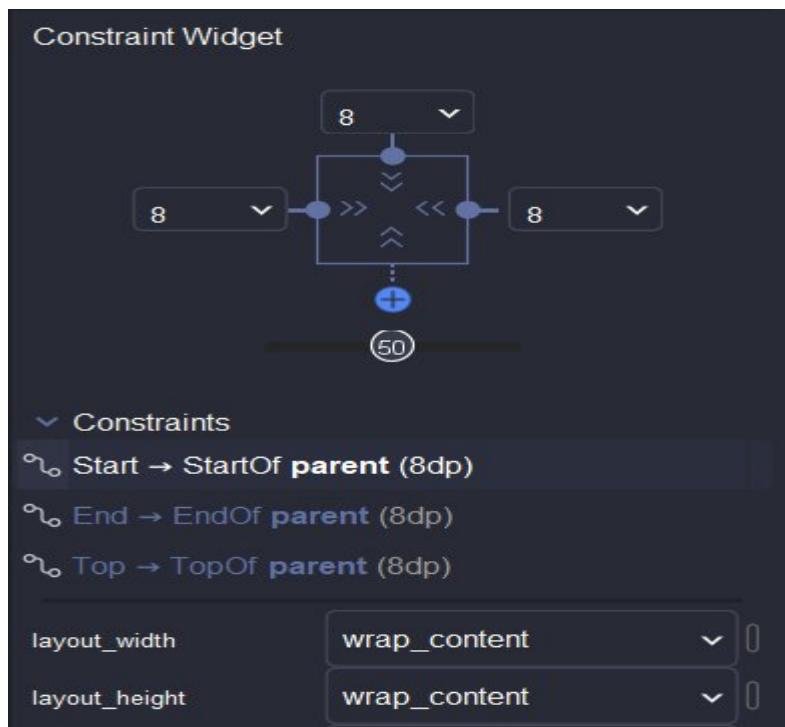
3.1 Thay đổi kích thước Button

Trình chỉnh sửa bộ cục cung cấp các tay cầm thay đổi kích thước ở bốn góc của một **View**, giúp bạn có thể điều chỉnh kích thước nhanh chóng. Bạn có thể kéo các tay cầm này để thay đổi kích thước của **View**, nhưng cách này sẽ đặt kích thước cố định (**hardcoded**).

Lưu ý: Tránh đặt kích thước cố định cho hầu hết các phần tử **View**, vì chúng không thể tự động điều chỉnh theo nội dung và kích thước màn hình khác nhau.

Thay vào đó, hãy sử dụng ngăn **Attributes** ở bên phải trình chỉnh sửa bộ cục để chọn một chế độ kích thước không sử dụng giá trị cố định.

- **Ngăn Attributes** bao gồm một bảng kích thước hình vuông gọi là **view inspector** ở phía trên.
- Các biểu tượng bên trong hình vuông này đại diện cho các cài đặt chiều cao và chiều rộng như sau:



Trong hình trên:

1. Điều khiển chiều cao (Height control):

- Điều khiển này xác định thuộc tính **layout_height** và xuất hiện dưới dạng hai đoạn ở cạnh trên và cạnh dưới của hình vuông.

- Nếu các góc được bo tròn, điều đó có nghĩa là thuộc tính này được đặt thành **wrap_content**, tức là View sẽ mở rộng theo chiều dọc tùy theo nội dung bên trong.
- Số 8 hiển thị trên điều khiển này cho biết một khoảng lè tiêu chuẩn là **8dp**.

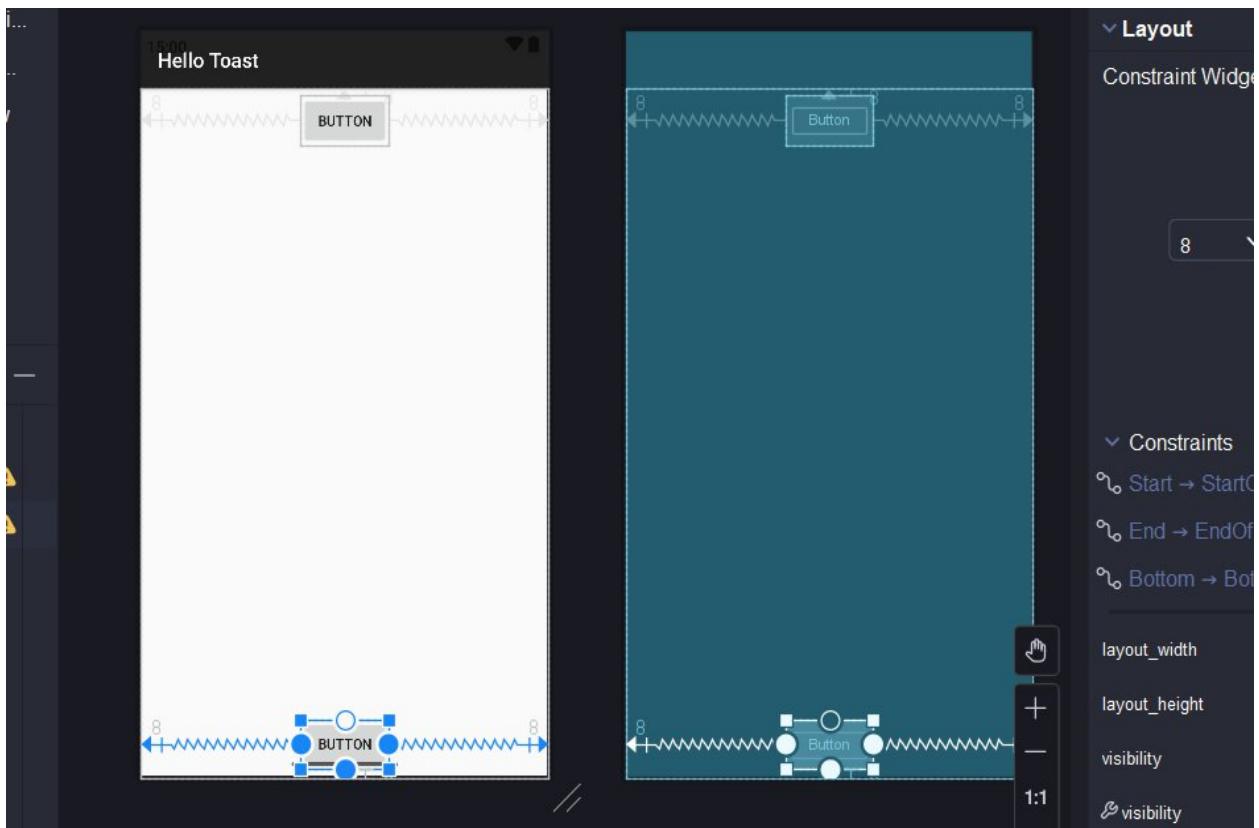
2. Điều khiển chiều rộng (Width control):

- Điều khiển này xác định thuộc tính **layout_width** và xuất hiện dưới dạng hai đoạn ở cạnh trái và cạnh phải của hình vuông.
- Nếu các góc được bo tròn, điều đó có nghĩa là thuộc tính này được đặt thành **wrap_content**, tức là View sẽ mở rộng theo chiều ngang tùy theo nội dung bên trong nghĩa là View sẽ mở rộng theo chiều ngang tùy theo nội dung bên trong, với khoảng lè tối đa là **8dp**.

2. Nút đóng ngăn Attributes (Attributes pane close button): Nhấp vào nút này để đóng ngăn Attributes.

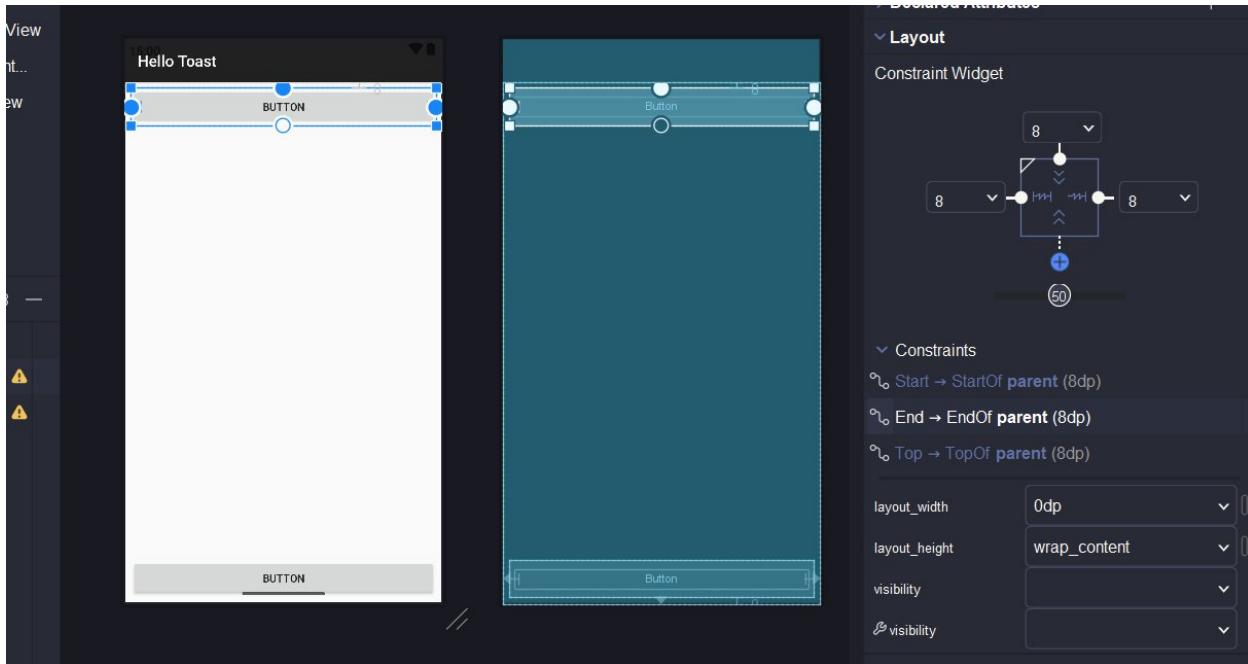
Thực hiện các bước sau:

1. Chọn **Button** ở phía trên trong ngăn **Component Tree**.
2. Nhấp vào tab **Attributes** ở bên phải cửa sổ trình chỉnh sửa bố cục.
3. Nhấp vào **điều khiển chiều rộng (width control)** hai lần:
 - Lần nhấp đầu tiên sẽ thay đổi nó thành **Fixed** (Cố định), được biểu thị bằng các đường thẳng.
 - Lần nhấp thứ hai sẽ thay đổi nó thành **Match Constraints** (Khớp ràng buộc), được biểu thị bằng các lò xo, như trong hình động bên dưới.



Do thay đổi điều khiển chiều rộng, thuộc tính **layout_width** trong ngăn **Attributes** hiển thị giá trị **match_constraint**, và phần tử **Button** sẽ kéo giãn theo chiều ngang để lấp đầy khoảng trống giữa cạnh trái và cạnh phải của bố cục.

4. Chọn **Button** thứ hai và thực hiện các thay đổi tương tự cho **layout_width** như ở bước trước, như hiển thị trong hình bên dưới.

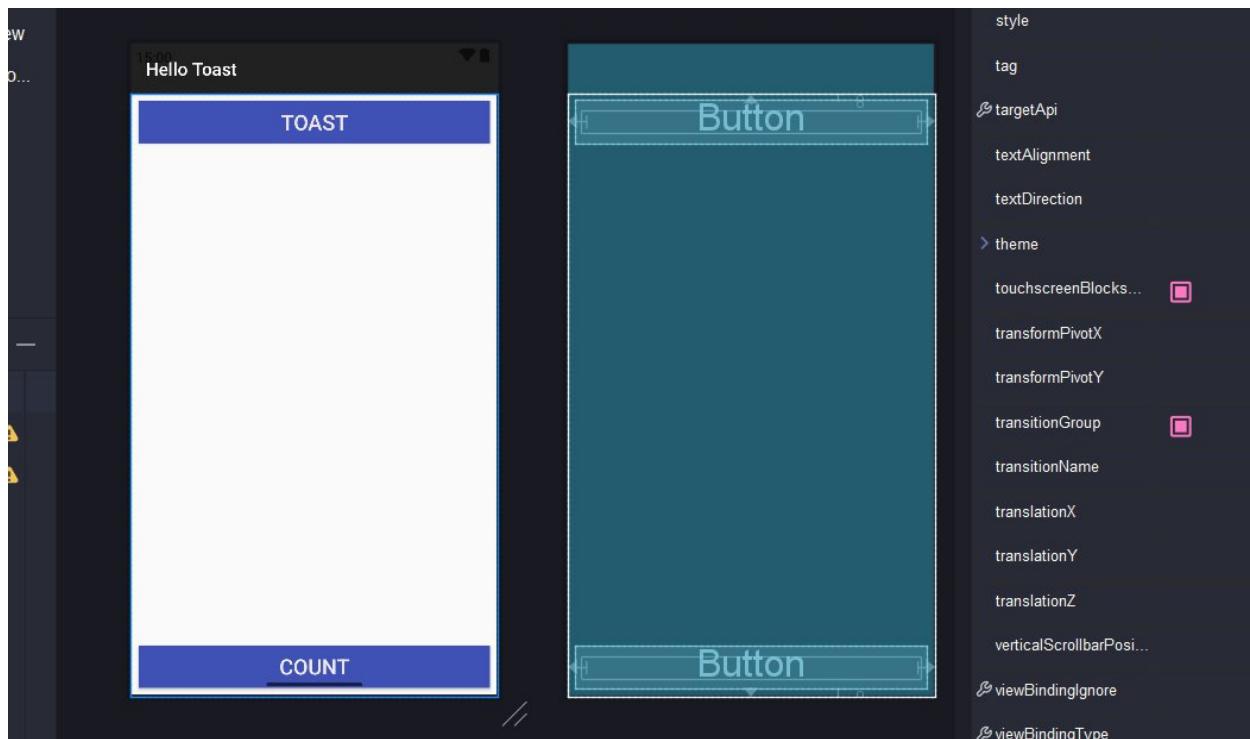


Như đã thấy trong các bước trước, các thuộc tính **layout_width** và **layout_height** trong ngăn **Attributes** sẽ thay đổi khi bạn điều chỉnh các điều khiển chiều cao và chiều rộng trong **inspector**. Các thuộc tính này có thể nhận một trong ba giá trị trong **ConstraintLayout**:

- **match_constraint** mở rộng phần tử **View** để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao, tối đa đến khoảng lè nếu có. Trong trường hợp này, phần tử cha là **ConstraintLayout**. Bạn sẽ tìm hiểu thêm về **ConstraintLayout** trong nhiệm vụ tiếp theo.
- **wrap_content** thu nhỏ kích thước của phần tử **View** sao cho vừa đủ chứa nội dung bên trong. Nếu không có nội dung, phần tử **View** sẽ trở nên vô hình.
- Để chỉ định kích thước cố định nhưng vẫn điều chỉnh theo kích thước màn hình của thiết bị, hãy sử dụng một số cố định với đơn vị **dp** (**density-independent pixels**). Ví dụ, **16dp** có nghĩa là **16 pixel độc lập với mật độ màn hình**.

Nhiệm vụ 4: Thêm một **TextView** và thiết lập thuộc tính

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



4.1 Thêm một TextView và thiết lập ràng buộc

1. Như trong hình động bên dưới, kéo một **TextView** từ ngăn **Palette** vào phần trên của bố cục, sau đó kéo một ràng buộc từ cạnh trên của **TextView** đến tay cầm ở cạnh dưới của nút **Toast Button**. Việc này sẽ ràng buộc **TextView** nằm ngay bên dưới **Button**.
2. Như trong hình động bên dưới, kéo một ràng buộc từ cạnh dưới của **TextView** đến tay cầm ở cạnh trên của nút **Count Button**, và kéo các ràng buộc từ hai cạnh của **TextView** đến hai cạnh của bố cục. Việc này sẽ ràng buộc **TextView** nằm ở giữa bố cục, giữa hai **Button**.

4.2 Đặt thuộc tính TextView

Với **TextView** đã chọn, hãy mở ngăn **Thuộc tính**, nếu ngăn này chưa mở. Đặt thuộc tính cho

TextView như trong hình động bên dưới. Các thuộc tính bạn chưa gắp phải được giải thích sau hình:

1. Đặt ID thành show_count.
2. Đặt văn bản thành 0.
3. Đặt textSize thành 160sp.
4. Đặt textStyle thành B (in đậm) và textAlign thành ALIGNCENTER (căn giữa đoạn văn).
5. Thay đổi các điều khiển kích thước ché độ xem theo chiều ngang và chiều dọc (layout_width và layout_height) thành match_constraint.
5. Đặt textColor thành @color/colorPrimary.
7. Cuộn xuống ngăn và nhập vào Xem tất cả các thuộc tính, cuộn xuống trang thứ hai của thuộc tính đến nền, sau đó nhập #FFF00 để có màu vàng.
8. Cuộn xuống trọng lực, mở rộng trọng lực và chọn center_ver (cho center-vertical).

textSize: Kích thước văn bản của TextView. Đối với bài học này, kích thước được đặt thành 160sp. Sp viết tắt của pixel không phụ thuộc tỷ lệ, và giống như dp, là một đơn vị tỷ lệ với mật độ màn hình và tùy chọn kích thước phông chữ của người dùng. Sử dụng đơn vị dp khi bạn chỉ định kích thước phông chữ để kích thước được điều chỉnh cho cả mật độ màn hình và tùy chọn của người dùng.

- textStyle và textAlign: Kiểu văn bản, được đặt thành B (in đậm) trong bài học này và căn chỉnh văn bản, được đặt thành ALIGNCENTER (căn giữa đoạn văn).

- gravity: Thuộc tính gravity chỉ định cách View được căn chỉnh trong View hoặc ViewGroup cha của nó. Trong bước này, bạn căn giữa TextView để được căn giữa theo chiều dọc trong ConstraintLayout cha.

Bạn có thể nhận thấy rằng thuộc tính nền nằm trên trang đầu tiên của ngăn Thuộc tính cho một Nút, nhưng nằm trên trang thứ hai của ngăn Thuộc tính cho một TextView. Ngăn Thuộc tính thay đổi cho từng loại Chế độ xem: Các thuộc tính phổ biến nhất cho loại Chế độ xem xuất hiện trên trang đầu tiên, và các thuộc tính còn lại được liệt kê trên trang thứ hai. Để quay lại trang đầu tiên của ngăn Thuộc tính, hãy nhập vào biểu tượng trên thanh công cụ ở đầu ngăn.

Nhiệm vụ 5: Chính sửa bộ cục trong XML

Bố cục ứng dụng Hello Toast gần hoàn thiện! Tuy nhiên, một dấu chấm than xuất hiện bên cạnh mỗi thành phần UI trong Cây thành phần. Di con trỏ qua các dấu chấm than này để xem các thông báo cảnh báo, như hiển thị bên dưới. Cảnh báo tương tự xuất hiện cho cả ba thành phần: chuỗi được mã hóa cứng phải sử dụng tài nguyên.

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

5.1 Mở mã XML cho bố cục

Trình soạn thảo XML xuất hiện, thay thế các ngăn thiết kế và bản thiết kế. Như bạn có thể thấy trong hình bên dưới, hiển thị một phần mã XML cho bố cục, các cảnh báo được tô sáng—các chuỗi được mã hóa cứng "Toast" và "Count". (Chuỗi "0" được mã hóa cứng cũng được tô sáng nhưng không hiển thị trong hình.) Di con trỏ qua chuỗi được mã hóa cứng "Toast" để xem thông báo cảnh báo.

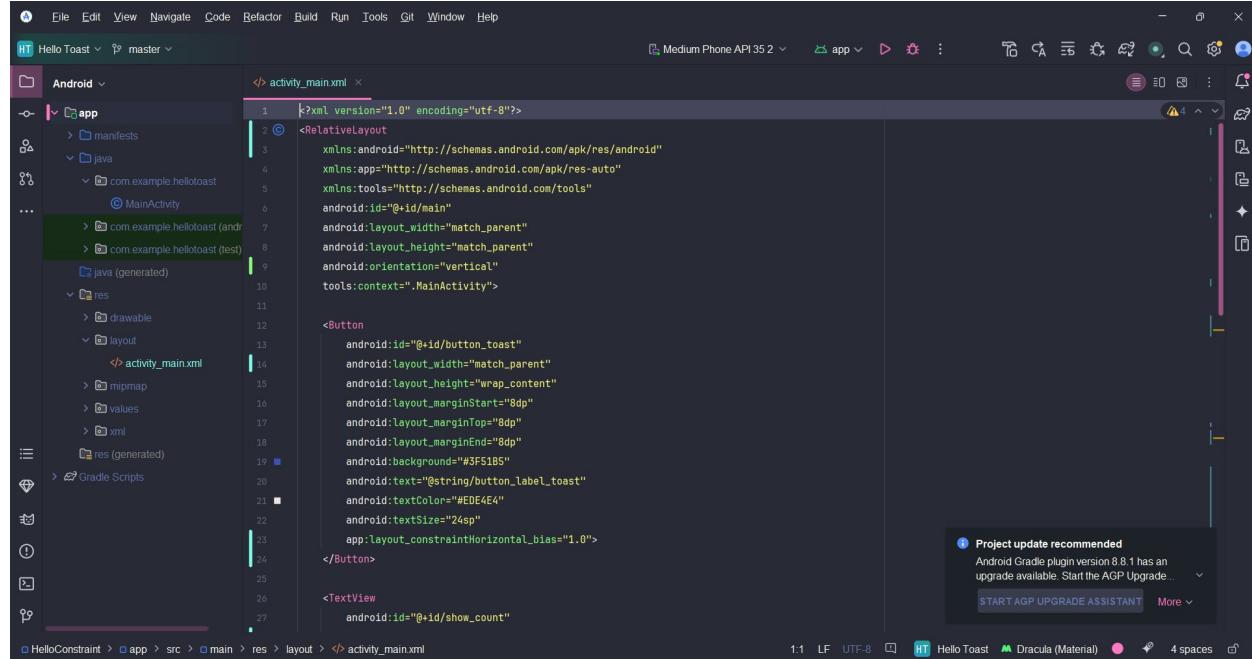
5.2 Trích xuất tài nguyên chuỗi

Thay vì mã hóa cứng chuỗi, cách tốt nhất là sử dụng tài nguyên chuỗi, đại diện cho các chuỗi. Việc có các chuỗi trong một tệp riêng giúp quản lý chúng dễ dàng hơn, đặc biệt là nếu bạn sử dụng các chuỗi này nhiều lần. Ngoài ra, tài nguyên chuỗi là bắt buộc để dịch và bản địa hóa ứng dụng của bạn, vì bạn cần tạo tệp tài nguyên chuỗi cho từng ngôn ngữ.

1. Nhập một lần vào từ "Toast" (cảnh báo được tô sáng đầu tiên).
2. Nhấn Alt-Enter trong Windows hoặc Option-Enter trong macOS và chọn Trích xuất tài nguyên chuỗi từ menu bật lên.
3. Nhập button_label_toast cho tên Tài nguyên.
4. Nhập vào OK. Một tài nguyên chuỗi được tạo trong tệp values/res/string.xml và chuỗi trong mã của bạn được thay thế bằng tham chiếu đến tài nguyên: @string/button_label_toast
5. Trích xuất các chuỗi còn lại: button_label_count cho "Count" và count_initial_value cho "0".

6. Trong ngăn Project > Android, hãy mở rộng các giá trị trong res, sau đó nhấp đúp vào strings.xml để xem tài nguyên chuỗi của bạn trong tệp strings.xml:

7. Bạn cần một chuỗi khác để sử dụng trong tác vụ tiếp theo hiển thị thông báo. Thêm vào tệp strings.xml một tài nguyên chuỗi khác có tên toast_message cho cụm từ "Hello Toast!":

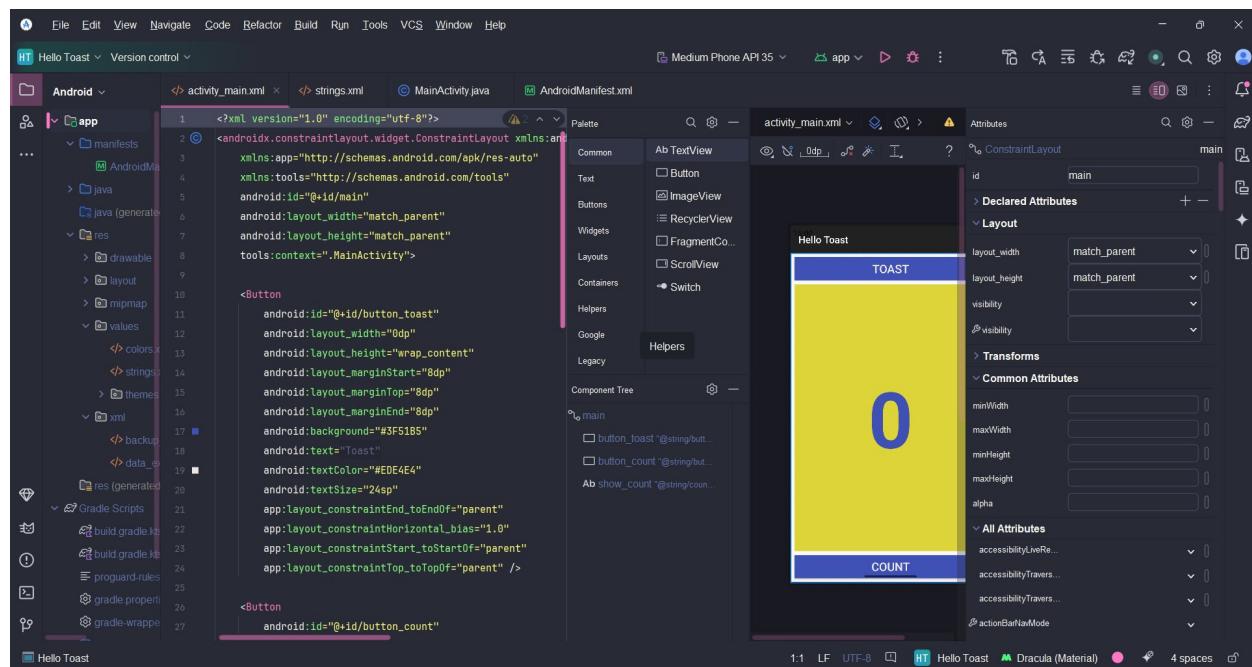


```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"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

    <Button
        android:id="@+id/button_toast"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:background="#3F51B5"
        android:text="@string/button_label_toast"
        android:textColor="#EDE4E4"
        android:textSize="24sp"
        app:layout_constraintHorizontal_bias="1.0"/>

    <TextView
        android:id="@+id/show_count"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"/>

```



Nhiệm vụ 6: Thêm trình xử lý onClick cho các nút

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

Trình xử lý nhấp là phương thức được gọi khi người dùng nhấp hoặc chạm vào phần tử UI có thể nhấp. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường onClick trong ngăn Thuộc tính của tab Thiết kế. Bạn cũng có thể chỉ định tên của phương thức xử lý trong trình soạn thảo XML bằng cách thêm thuộc tính android:onClick vào Button. Bạn sẽ sử dụng phương thức sau vì bạn chưa tạo các phương thức xử lý và trình soạn thảo XML cung cấp một cách tự động để tạo các phương thức đó.

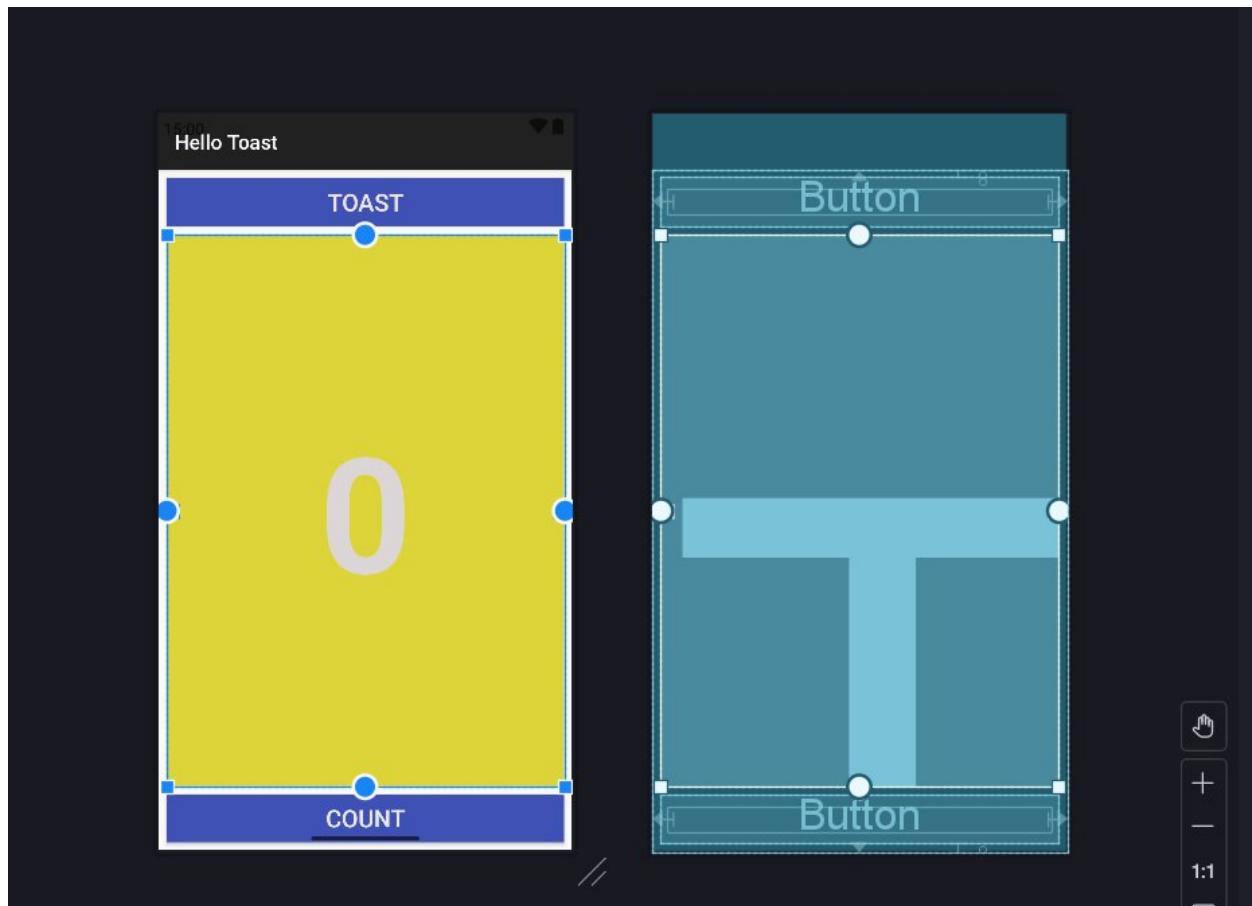
1. Khi trình soạn thảo XML mở (tab Văn bản), hãy tìm Button có android:id được đặt thành button_toast:
2. Thêm thuộc tính android:onClick vào cuối phần tử button_toast sau thuộc tính cuối cùng và trước chỉ báo kết thúc />:
3. Nhập vào biểu tượng bóng đèn màu đỏ xuất hiện bên cạnh thuộc tính. Chọn Tạo trình xử lý nhấp, chọn MainActivity và nhấp vào OK. Nếu biểu tượng bóng đèn màu đỏ không xuất hiện, hãy nhập vào tên phương thức ("showToast"). Nhấn Alt-Enter (Option-Enter trên máy Mac), chọn Tạo 'showToast(view)' trong MainActivity và nhấp vào OK. Hành động này tạo một stub phương thức giữ chỗ cho phương thức showToast() trong MainActivity, như được hiển thị ở cuối các bước này.
4. Lặp lại hai bước cuối cùng với Button_count: Thêm thuộc tính android:onClick vào cuối và thêm trình xử lý nhấp:

6.2 Chỉnh sửa trình xử lý Nút Toast

Bây giờ bạn sẽ chỉnh sửa phương thức showToast()—trình xử lý nhấp chuột Nút Toast trong MainActivity—để nó hiển thị một thông báo. Toast cung cấp một cách để hiển thị một thông báo đơn giản trong một cửa sổ bật lên nhỏ. Nó chỉ lấp đầy lượng không gian cần thiết cho thông báo. Hoạt động hiện tại vẫn hiển thị và 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 chạm vào Nút hoặc thực hiện một hành động.Làm theo các bước sau để chỉnh sửa trình xử lý nhấp chuột Nút Toast:

1. Xác định vị trí phương thức showToast() mới được tạo.
2. Để tạo một phiên bản của Toast, hãy gọi phương thức makeText() của lớp Toast.
3. Cung cấp ngữ cảnh của Activity ứng dụng. Vì Toast hiển thị ở trên cùng của Activity UI, hệ thống cần thông tin về Activity hiện tại. Khi bạn đã ở trong ngữ cảnh của Activity mà bạn cần ngữ cảnh, hãy sử dụng phương thức này như một phím tắt.



4. Cung cấp thông báo để hiển thị, chẳng hạn như tài nguyên chuỗi (taste_message bạn đã tạo ở bước trước). Tài nguyên chuỗi toast_message được xác định bởi R.string.

5. Cung cấp thời lượng hiển thị. Ví dụ: Toast.LENGTH_SHORT hiển thị toast trong thời gian tương đối ngắn. Thời lượng hiển thị Toast có thể là Toast.LENGTH_LONG hoặc Toast.LENGTH_SHORT. Độ dài thực tế là khoảng 3,5 giây cho Toast dài và 2 giây cho Toast ngắn.

6. Hiển thị Toast bằng cách gọi show(). Sau đây là toàn bộ phương thức showToast():

6.3 Chính sửa trình xử lý Nút đếm

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

Mã cho trình xử lý phải:

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

Thực hiện theo các bước sau để chỉnh sửa trình xử lý nhấp vào Nút đếm:

1. Xác định vị trí phương thức countUp() mới được tạo.

2. Để theo dõi số đếm, bạn cần một biến thành viên riêng. Mỗi lần nhấn nút Đếm sẽ làm tăng giá trị của biến này. Nhập nội dung sau, nội dung này sẽ được tô sáng bằng màu đỏ và hiển thị biểu tượng bóng đèn màu đỏ:

Nếu biểu tượng bóng đèn đỏ không xuất hiện, hãy chọn biểu thức mCount++. Bóng đèn đỏ cuối cùng xuất hiện.

3. Nhấp vào biểu tượng bóng đèn đỏ và chọn Tạo trường 'mCount' từ menu bật lên. Điều này tạo một biến thành viên riêng tư tại

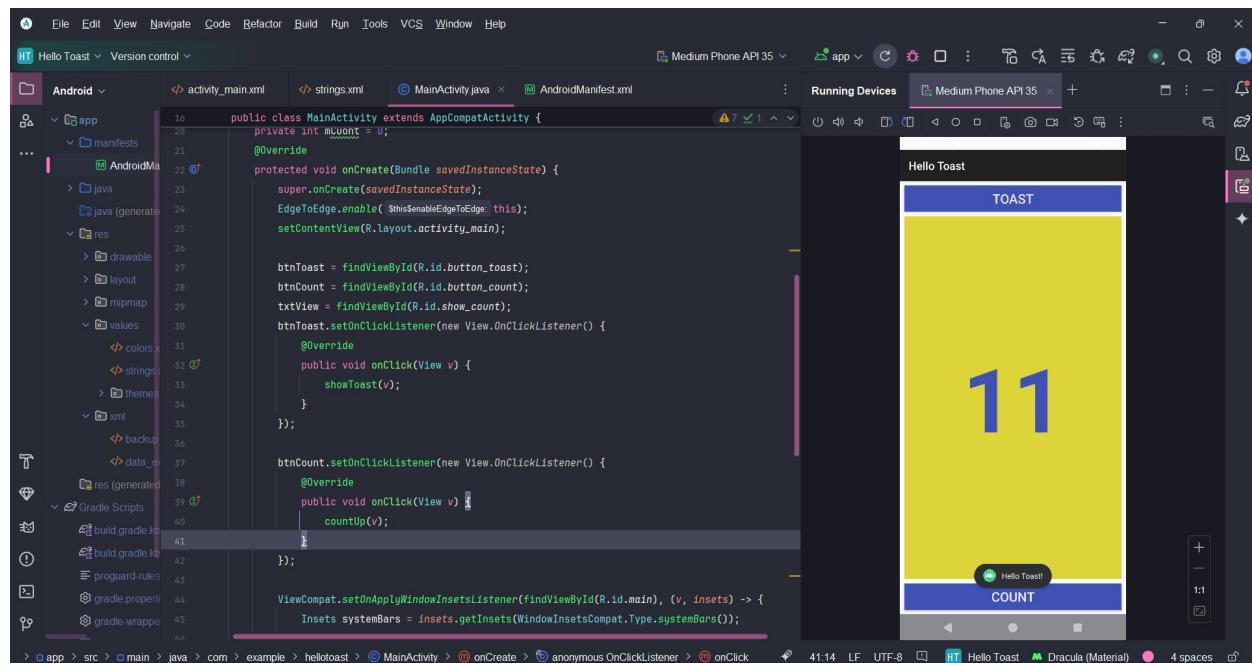
4. Thay đổi câu lệnh biến thành viên riêng tư để khởi tạo biến thành số không:

5. Cùng với biến ở trên, bạn cũng cần một biến thành viên riêng tư để tham chiếu đến TextView show_count, mà bạn sẽ thêm vào trình xử lý nhấp chuột. Gọi biến này mShowCount:6. Bây giờ bạn đã có mShowCount, bạn có thể lấy tham chiếu đến TextView bằng ID bạn đã đặt trong tệp bố cục. Để chỉ lấy tham chiếu này một lần, hãy chỉ định tham chiếu đó trong phương thức onCreate(). Như bạn đã tìm hiểu

trong bài học khác, phương thức `onCreate()` được sử dụng để làm phông bô cục, có nghĩa là đặt chế độ xem nội dung của màn hình thành bô cục XML. Bạn cũng có thể sử dụng phương thức này để lấy tham chiếu đến các thành phần UI khác trong bô cục, chẳng hạn như `TextView`. Xác định vị trí phương thức `onCreate()` trong `MainActivity`:

7. Thêm câu lệnh `findViewById` vào cuối phương thức `View`, giống như một chuỗi, là một tài nguyên có thể có một id. Lệnh gọi `findViewById` lấy ID của một view làm tham số và trả về `View`. Vì phương thức trả về một `View`, bạn phải ép kết quả thành kiểu view mà bạn mong đợi, trong trường hợp này là (`TextView`).

8. Nay bạn đã gán `TextView` cho `mShowCount`, bạn có thể sử dụng biến để đặt văn bản trong `TextView` thành giá trị của biến `mCount`. Thêm nội dung sau vào phương thức `countUp()`



Mã giải pháp

Ứng dụng `HelloToast` trông ổn khi thiết bị hoặc trình giả lập được định hướng theo chiều dọc. Tuy nhiên, nếu bạn chuyển thiết bị hoặc trình giả lập sang định hướng theo chiều ngang, Nút Đếm có thể chòng lên `TextView` dọc theo phía dưới như minh họa trong hình bên dưới.

Thử thách: Thay đổi bô cục sao cho đẹp theo cả hướng ngang và hướng dọc:

- Trên máy tính của bạn, tạo một bản sao của thư mục dự án HelloToast và đổi tên thành HelloToastChallenge.
- Mở HelloToastChallenge trong Android Studio và sắp xếp lại. (Xem Phụ lục: Tiện ích để biết hướng dẫn về cách sao chép và sắp xếp lại dự án.)
- Thay đổi bố cục sao cho Nút Toast và Nút Count xuất hiện ở bên trái, như hiển thị trong hình bên dưới. TextView xuất hiện bên cạnh chúng, nhưng chỉ đủ rộng để hiển thị nội dung của nó. (Gợi ý: Sử dụng wrap_content.)
- Chạy ứng dụng theo cả hướng ngang và hướng dọc.

Bài 1.2 Phần B: Trình chỉnh sửa bố cục

Giới thiệu

Như bạn đã học trong 1.2 Phân A: Giao diện người dùng tương tác đầu tiên của bạn, bạn có thể xây dựng giao diện người dùng (UI) bằng cách sử dụng ConstraintLayout trong trình chỉnh sửa bố cục, nơi đặt các thành phần UI trong bố cục bằng cách sử dụng các kết nối ràng buộc với các thành phần khác và với các cạnh bố cục. ConstraintLayout được thiết kế để giúp dễ dàng kéo các thành phần UI vào trình chỉnh sửa bố cục. ConstraintLayout là ViewGroup, đây là View đặc biệt có thể chứa các đối tượng View khác (gọi là con hoặc chế độ xem con). Bài thực hành này sẽ trình bày thêm các tính năng của ConstraintLayout và trình chỉnh sửa bố cục. Bài thực hành này cũng giới thiệu hai lớp con ViewGroup khác:

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

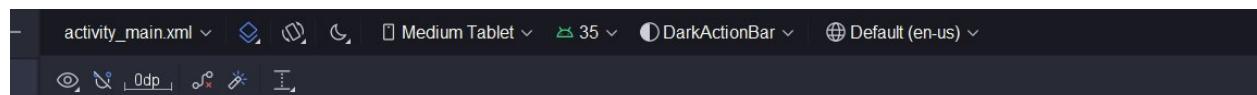
Nhiệm vụ 1: Tạo các biến có thể bố trí

Trong bài học trước, thử mã hóa yêu cầu thay đổi bố cục của ứng dụng Hello Toast để nó phù hợp với hướng ngang hoặc hướng dọc. Trong nhiệm vụ này, bạn sẽ học cách dễ dàng hơn để tạo ra các biến thể bố trí của mình theo hướng ngang (còn gọi là ngang) và hướng dọc (còn lại) gọi là dọc) cho điện thoại và cho màn hình lớn hơn như bảng máy tính. Trong nhiệm vụ này, bạn sẽ sử dụng một số nút trong hai thanh công cụ trên cùng trình chỉnh sửa bố cục. Công cụ này cho phép bạn định cấu hình giao diện của bản xem trước bố cục trong trình chỉnh sửa bố cục:

Trong hình trên:

1. Chọn Design Surface: Chọn Design để hiển thị bản xem trước màu của bố cục hoặc Blueprint để chỉ hiển thị phác thảo cho từng thành phần UI. Để xem cả hai ngăn cạnh nhau, hãy chọn Design + Blueprint.
2. Orientation in Editor: Chọn Portrait hoặc Landscape để hiển thị bản xem trước theo hướng dọc hoặc ngang. Điều này hữu ích khi xem trước bố cục mà không cần phải chạy ứng dụng trên trình giả lập hoặc thiết bị. Để tạo bố cục thay thế, hãy chọn Create Landscape Variation hoặc các biến thể khác.
3. Device in Editor: Chọn loại thiết bị (điện thoại/máy tính bảng, Android TV hoặc Android Wear).
4. API Version in Editor: Chọn phiên bản Android để sử dụng để hiển thị bản xem trước.
5. Theme in Editor: Chọn một chủ đề (như AppTheme) để áp dụng cho bản xem trước.
6. Locale in Editor: Chọn ngôn ngữ và locale cho bản xem trước. Danh sách này chỉ hiển thị các ngôn ngữ có sẵn trong tài nguyên chuỗi (xem bài học về bản địa hóa để biết chi tiết về cách thêm ngôn ngữ). Bạn cũng có thể chọn Xem trước từ Phải sang Trái để xem bố cục như thể ngôn ngữ RTL đã được chọn.

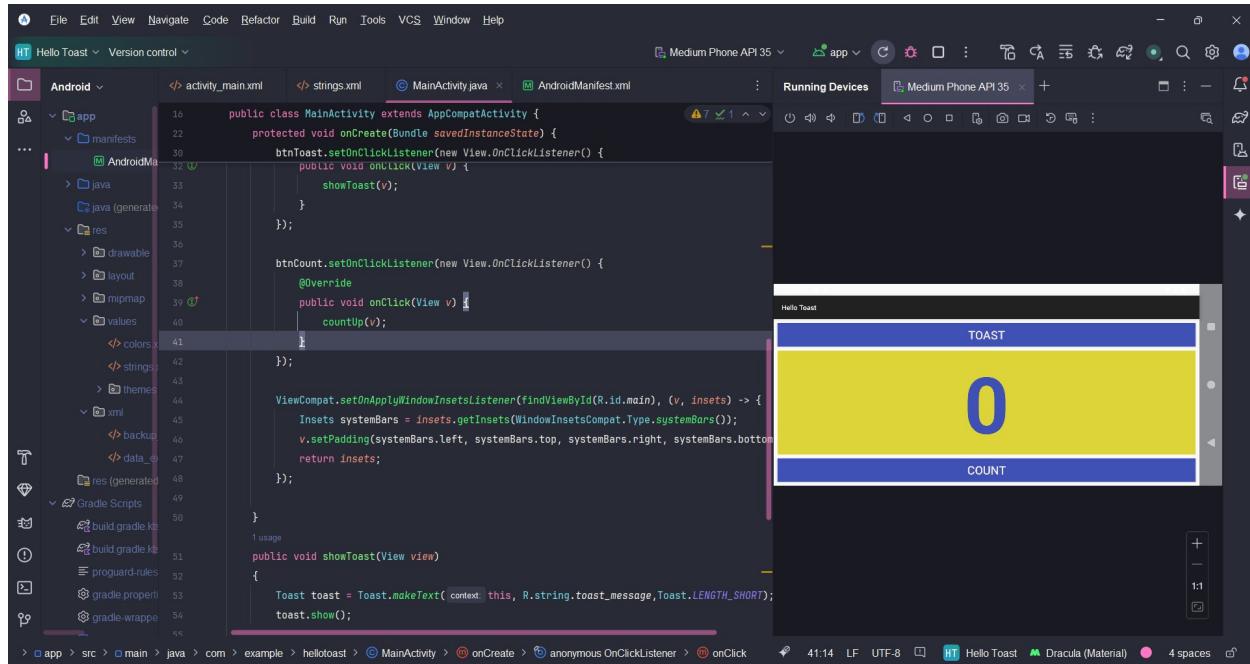
Thanh công cụ thứ hai cho phép bạn cấu hình giao diện của các thành phần UI trong ConstraintLayout và phóng to và thu nhỏ bản xem trước:



Trong hình trên:

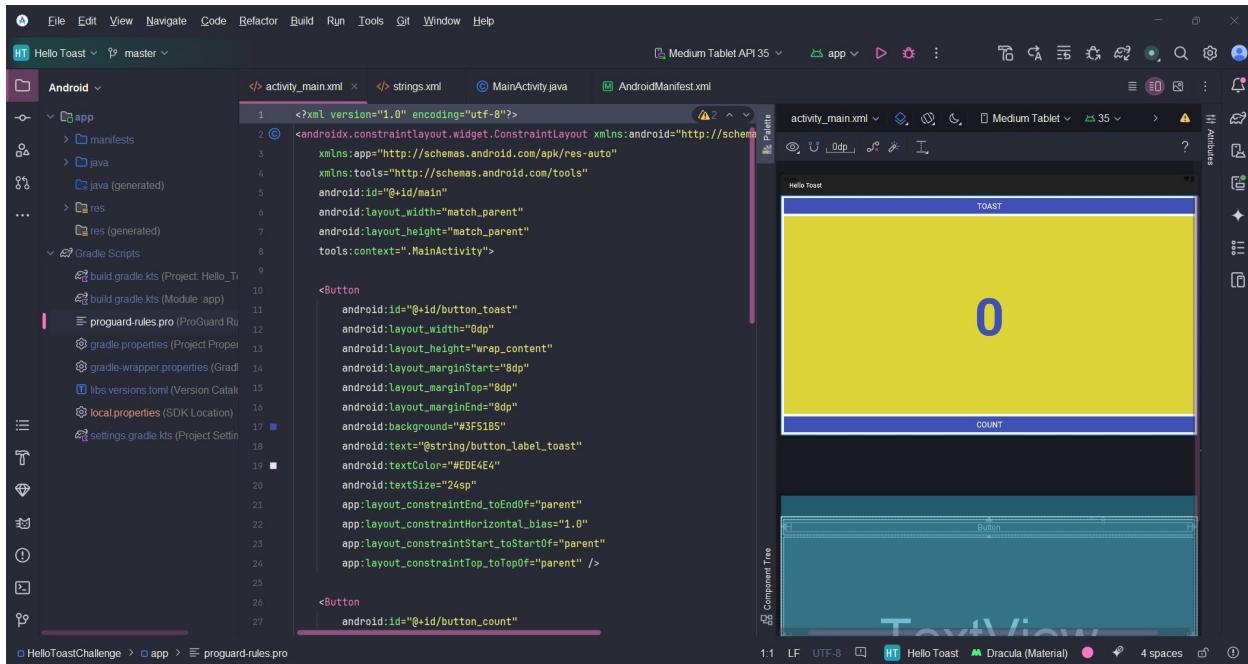
1. Hiển thị: Chọn Hiển thị ràng buộc và Hiển thị lề để hiển thị chúng trong bản xem trước hoặc dừng hiển thị chúng.
2. Tự động kết nối: Bật hoặc tắt Tự động kết nối. Khi Tự động kết nối được bật, bạn có thể kéo bất kỳ phần tử nào (chẳng hạn như Nút) đến bất kỳ phần nào của bố cục để tạo ràng buộc đối với bố cục cha.
3. Xóa tất cả ràng buộc: Xóa tất cả ràng buộc trong toàn bộ bố cục.
4. Suy ra ràng buộc: Tạo ràng buộc bằng suy luận.

5. Lè mặc định: Đặt lè 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 đường hướng dẫn theo chiều dọc hoặc chiều ngang.
9. Điều khiển thu phóng/xoay: Phóng to hoặc thu nhỏ



1.5 Tạo biến thể bộ cục cho máy tính bảng

Như bạn đã học trước đó, bạn có thể xem trước bộ cục cho các thiết bị khác nhau bằng cách nhấp vào nút Thiết bị trong Trình chỉnh sửa trên thanh công cụ trên cùng. Nếu bạn chọn một thiết bị như Nexus 10 (máy tính bảng) từ menu, bạn có thể thấy rằng bộ cục không lý tưởng cho màn hình máy tính bảng—văn bản của mỗi Nút quá nhỏ và cách sắp xếp các thành phần Nút ở trên cùng và dưới cùng không lý tưởng cho máy tính bảng màn hình lớn.



Để khắc phục lỗi này cho máy tính bảng trong khi vẫn giữ nguyên hướng ngang và hướng dọc của kích thước điện thoại, bạn có thể tạo biến thể của bộ cục hoàn toàn khác cho máy tính bảng. Thực hiện theo các bước sau:

1. Nhấp vào tab Thiết kế (nếu chưa chọn) để hiển thị ngăn thiết kế và bản thiết kế.
2. Nhấp vào nút Định hướng trong Trình chỉnh sửa trên thanh công cụ trên cùng.
3. Chọn Tạo biến thể bộ cục x-large.

Một cửa sổ trình chỉnh sửa mới mở ra với tab xlARGE/activity_main.xml hiển thị bộ cục cho thiết bị có kích thước máy tính bảng. Trình chỉnh sửa cũng chọn một thiết bị máy tính bảng, chẳng hạn như Nexus 9 hoặc Nexus 10, để xem trước. Bạn có thể thay đổi bộ cục này, dành riêng cho máy tính bảng, mà không cần thay đổi các bộ cục khác.

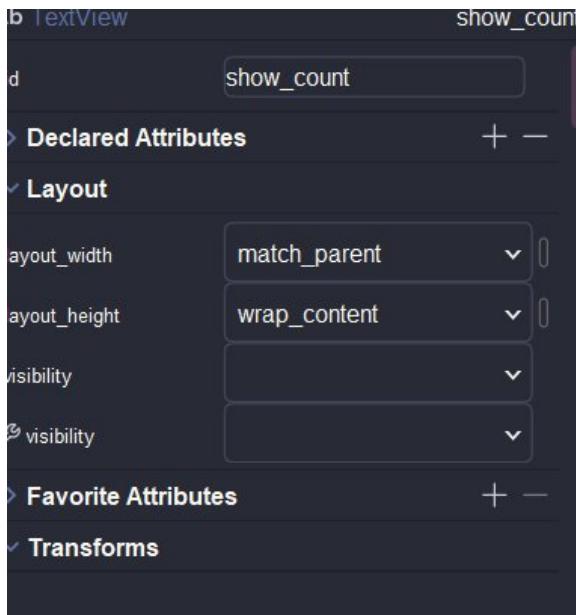
1.6 Thay đổi biến thể bộ cục cho máy tính bảng

Bạn có thể sử dụng ngăn Thuộc tính trong tab Thiết kế để thay đổi thuộc tính cho bộ cục này.

1. Tắt công cụ Tự động kết nối trên thanh công cụ. Đối với bước này, hãy đảm bảo rằng công cụ đã bị vô hiệu hóa:
2. Xóa tất cả các ràng buộc trong bộ cục bằng cách nhấp vào nút Xóa tất cả các ràng buộc trên thanh công cụ. Khi các ràng buộc đã được xóa, bạn có thể tự do di chuyển và thay đổi kích thước các thành phần trên bộ cục.

3. Trình chỉnh sửa bộ cục cung cấp các nút điều khiển thay đổi kích thước ở cả bốn góc của một thành phần để thay đổi kích thước của thành phần đó. Trong Cây thành phần, hãy chọn TextView có tên là show_count. Để TextView không còn cản trở bạn có thể tự do kéo các thành phần Nút, hãy kéo một góc của thành phần đó để thay đổi kích thước, như được hiển thị trong hình động bên dưới. Thay đổi kích thước của một phần tử sẽ mã hóa cùng các kích thước chiều rộng và chiều cao. Tránh mã hóa cùng các kích thước cho hầu hết các phần tử, vì bạn không thể dự đoán các kích thước được mã hóa cùng sẽ trông như thế nào trên các màn hình có kích thước và mật độ khác nhau. Nay giờ bạn chỉ thực hiện việc này để di chuyển phần tử ra khỏi đường đi và bạn sẽ thay đổi các kích thước trong một bước khác.

4. Chọn Nút button_toast trong Cây thành phần, nhập vào tab Thuộc tính để mở ngăn Thuộc tính và thay đổi textSize thành 60sp (#1 trong hình bên dưới) và layout_width thành wrap_content (#2 trong hình bên dưới).



Như được hiển thị ở phía bên phải của hình trên (2), bạn có thể nhập vào điều khiển chiều rộng của thanh tra chế độ xem, xuất hiện trong hai phân đoạn ở bên trái và bên phải của hình vuông, cho đến khi nó hiển thị Wrap Content. Ngoài ra, bạn có thể chọn wrap_content từ menu layout_width. Bạn sử dụng wrap_content để nếu văn bản Button được bản địa hóa sang ngôn ngữ khác, Button sẽ xuất hiện rộng hơn hoặc mỏng hơn để phù hợp với từ trong ngôn ngữ khác.

5. Chọn Button_count Button trong Component Tree, thay đổi textSize thành 60sp và layout_width thành wrap_content, rồi kéo Button phía trên TextView đến một khoảng trống trong bộ cục.

1.7 Sử dụng ràng buộc đường cơ sở

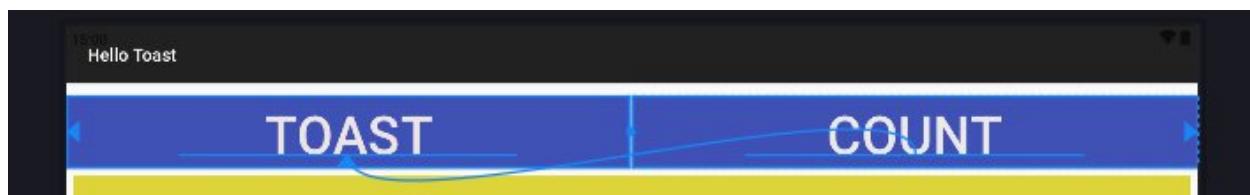
Bạn có thể căn chỉnh một phần tử UI có chứa văn bản, chẳng hạn như TextView hoặc Button, với một phần tử UI khác có chứa văn bản. Ràng buộc đường cơ sở cho phép bạn ràng buộc các phần tử sao cho các đường cơ sở văn bản khớp với nhau.

1. Ràng buộc Button_toast ở phía trên và bên trái của bố cục, kéo Button button_count đến một khoảng trống gần Button_toast và ràng buộc Button button_count ở phía bên trái của Button_toast, như được hiển thị trong hình động:
2. Sử dụng ràng buộc đường cơ sở, bạn có thể ràng buộc Button_count sao cho đường cơ sở văn bản của nó khớp với đường cơ sở văn bản của Button_toast. Chọn phần tử button_count, sau đó di con trỏ qua phần tử cho đến khi nút ràng buộc đường cơ sở xuất hiện bên dưới phần tử.
3. Nhấp vào nút ràng buộc đường cơ sở. Tay cầm đường cơ sở xuất hiện, nhấp nháy màu xanh lá cây như được hiển thị trong hình động. Nhấp và kéo một đường ràng buộc đường cơ sở đến đường cơ sở của phần tử button_toast.

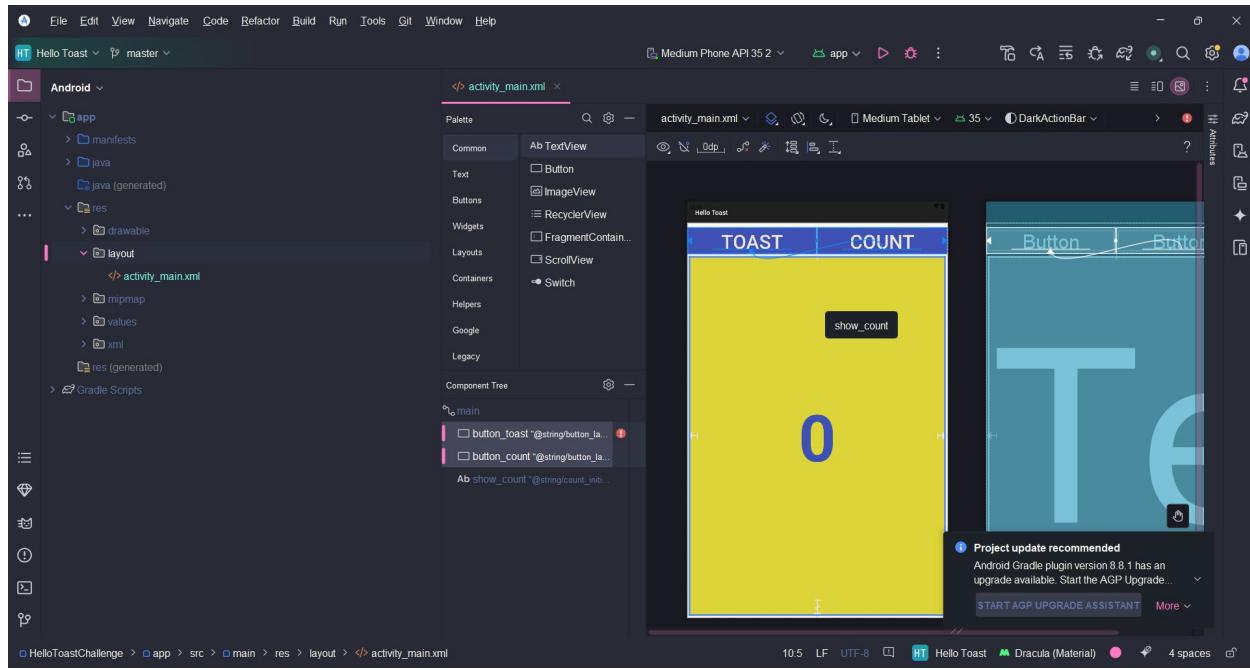
1.8 Mở rộng các nút theo chiều ngang

Nút pack trên thanh công cụ cung cấp các tùy chọn để đóng gói hoặc mở rộng các thành phần UI đã chọn. Bạn có thể sử dụng nút này để sắp xếp đều các thành phần Button theo chiều ngang trên toàn bộ bố cục.

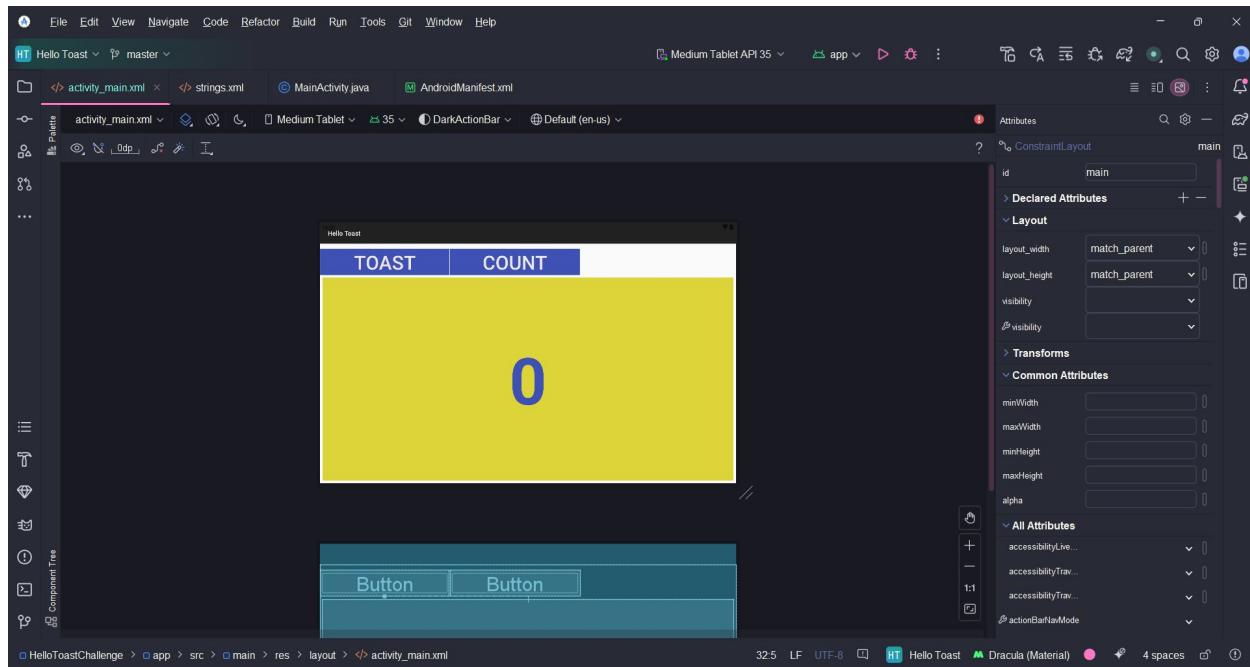
1. Chọn Button_count Button trong Component Tree và giữ Shift-select button_toast Button để cả hai đều được chọn.
2. Nhấp vào nút pack trên thanh công cụ và chọn Expand Horizontally như trong hình bên dưới.



3. Để hoàn thiện bố cục, hãy ràng buộc TextView show_count vào cuối button_toast Button và vào các cạnh và cuối của bố cục, như thể hiện trong hình động bên dưới.
4. Các bước cuối cùng là thay đổi layout_width và layout_height của TextView show_count thành Match Constraints và textSize thành 200sp. Bố cục cuối cùng trông giống như hình bên dưới.



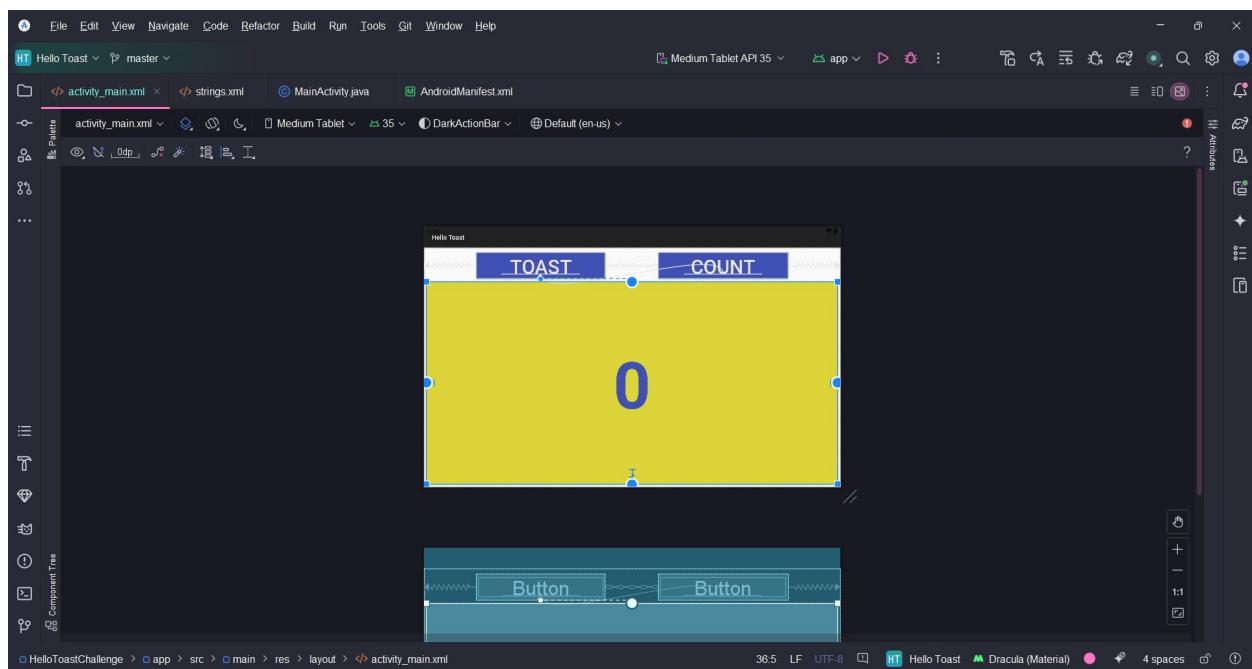
5. Nhập vào nút Định hướng trong Trình chỉnh sửa ở thanh công cụ trên cùng và chọn Chuyển sang Phong cảnh. Bố cục máy tính bảng xuất hiện theo hướng ngang như minh họa bên dưới. (Bạn có thể chọn Chuyển sang Chân dung để trở về hướng dọc.).



6. Chạy ứng dụng trên các trình giả lập khác nhau và thay đổi hướng sau khi chạy ứng dụng để xem ứng dụng trông như thế nào trên các loại thiết bị khác nhau. Bạn đã tạo thành công một ứng dụng có thể chạy với giao diện người dùng phù hợp trên điện thoại và máy tính bảng có kích thước và mật độ màn hình khác nhau.

Thách thức: Để phù hợp với hướng ngang (ngang) cho máy tính bảng, bạn có thể căn giữa các thành phần Nút trong activity_main.xml (cực lớn) để chúng xuất hiện như trong hình bên dưới.

Gợi ý: Chọn các thành phần, nhập vào nút căn chỉnh trên thanh công cụ và chọn Căn giữa theo chiều ngang.



Nhiệm vụ 2: Thay đổi bố cục thành LinearLayout <trang 93>

[LinearLayout](#) là ViewGroup sắp xếp tập hợp các chế độ xem của nó theo hàng ngang hoặc dọc. LinearLayout là một trong những bố cục phổ biến nhất vì nó đơn giản và nhanh. Nó thường được sử dụng trong một nhóm chế độ xem khác để sắp xếp các thành phần UI theo chiều ngang hoặc chiều dọc.

LinearLayout cần có các thuộc tính sau :

- chiều rộng bố cục
- chiều cao bố cục
- định hướng

layout_width và layout_height có thể sử dụng một trong các giá trị sau:

- match_parent : Mở rộng chế độ xem để lấp đầy chế độ xem cha theo chiều rộng hoặc chiều cao. Khi LinearLayout là chế độ xem gốc, nó sẽ mở rộng theo kích thước của màn hình (chế độ xem cha).
- wrap_content : Thu nhỏ kích thước chế độ xem để chế độ xem đủ lớn để bao gồm nội dung của nó. Nếu không có nội dung, chế độ xem sẽ trở nên vô hình.
- Số dp cố định ([pixel không phụ thuộc mật độ](#)): Chỉ định kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị. Ví dụ, 16dp nghĩa là 16 pixel không phụ thuộc mật độ.

Hướng có thể là:

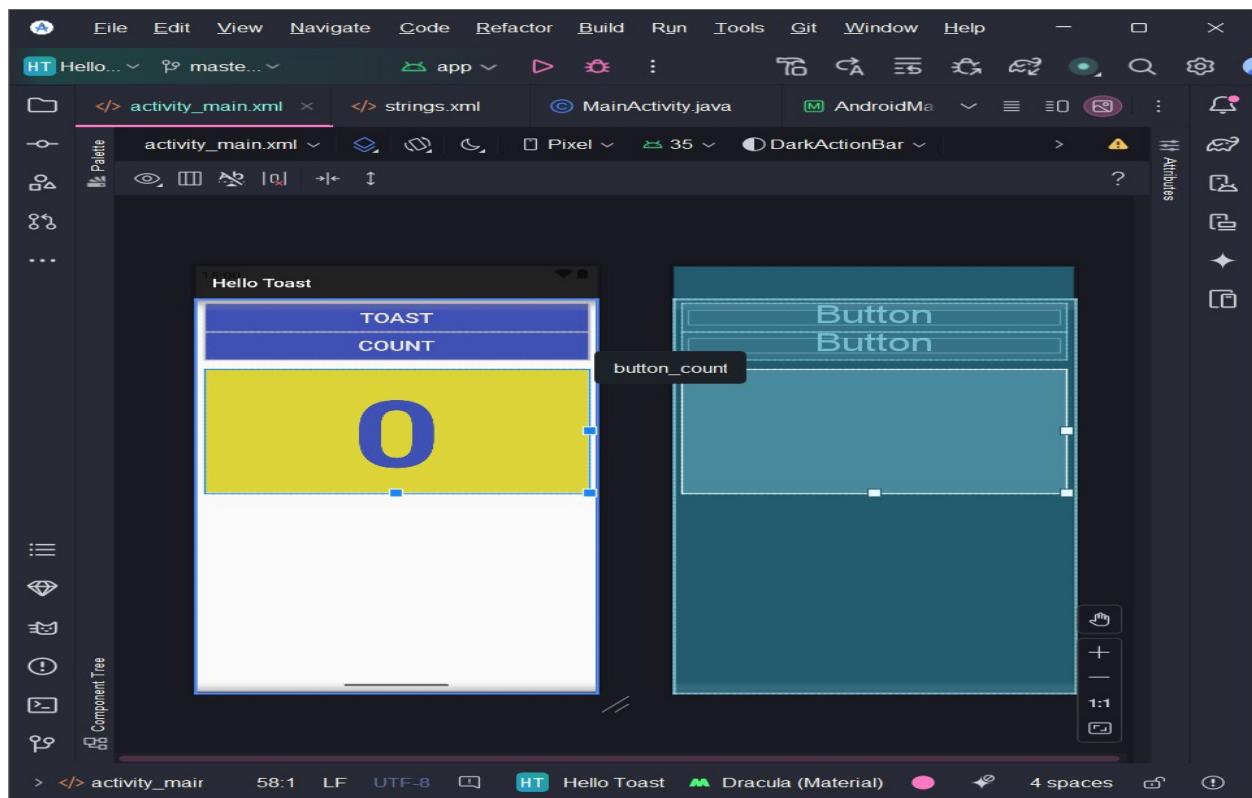
- ngang : Các chế độ xem được sắp xếp từ trái sang phải.
- dọc : Các chế độ xem được sắp xếp từ trên xuống dưới.

Trong nhiệm vụ này, bạn sẽ thay đổi nhóm chế độ xem gốc ConstraintLayout cho ứng dụng Hello Toast thành LinearLayout để bạn có thể thực hành sử dụng LinearLayout .

2.1 Thay đổi nhóm chế độ xem gốc thành LinearLayout

1. Mở ứng dụng Hello Toast từ tác vụ trước đó.
2. Mở tệp bô cục activity_main.xml (nếu tệp chưa được mở) và nhấp vào tab **Văn bản** ở cuối ngăn chỉnh sửa để xem mã XML. Ở đầu cùng của mã XML là dòng thẻ sau:

2.2 Thay đổi thuộc tính phần tử cho LinearLayout

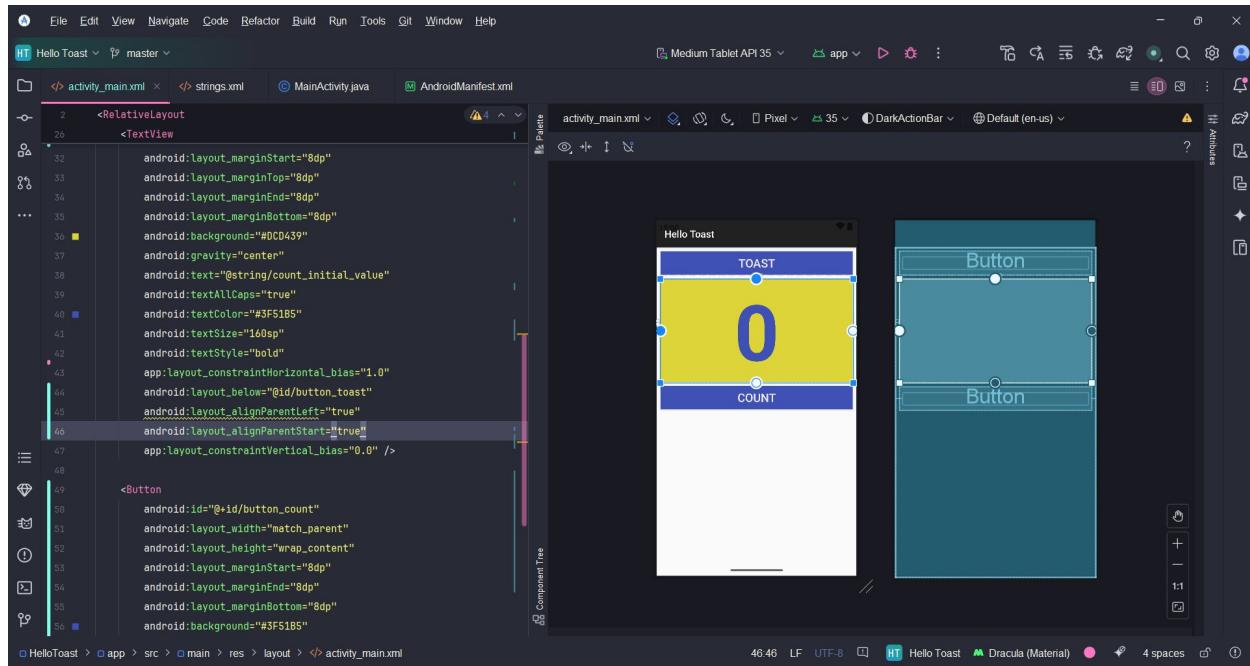


2.3 Thay đổi vị trí của các phần tử trong LinearLayout

[LinearLayout](#) sắp xếp các phần tử của nó theo hàng ngang hoặc dọc. Bạn đã thêm thuộc tính android:orientation="vertical" cho LinearLayout, do đó các phần tử được xếp chồng lên nhau theo chiều dọc như thể hiện trong hình trước.

Để thay đổi vị trí của chúng sao cho nút **Đếm** ở phía dưới, hãy làm theo các bước sau:

1. Mở ứng dụng Hello Toast từ tác vụ trước đó.
2. Mở tệp bố cục activity_main.xml (nếu tệp này chưa được mở) và nhấp vào tab **Văn bản**.
3. Chọn Nút button_count và tất cả các thuộc tính của nó, từ thẻ <Button> cho đến thẻ đóng />, rồi chọn **Edit > Cut**.
4. Nhấp vào sau thẻ đóng /> của phần tử TextView nhưng trước thẻ đóng </LinearLayout> và chọn **Chỉnh sửa > Dán**.
5. (Tùy chọn) Để khắc phục bất kỳ vấn đề thụt lề hoặc khoảng cách nào vì mục đích thẩm mỹ, hãy chọn **Mã > Định dạng lại mã** để định dạng lại mã XML với khoảng cách và thụt lề thích hợp.



2.4 Thêm trọng lượng cho phần tử TextView

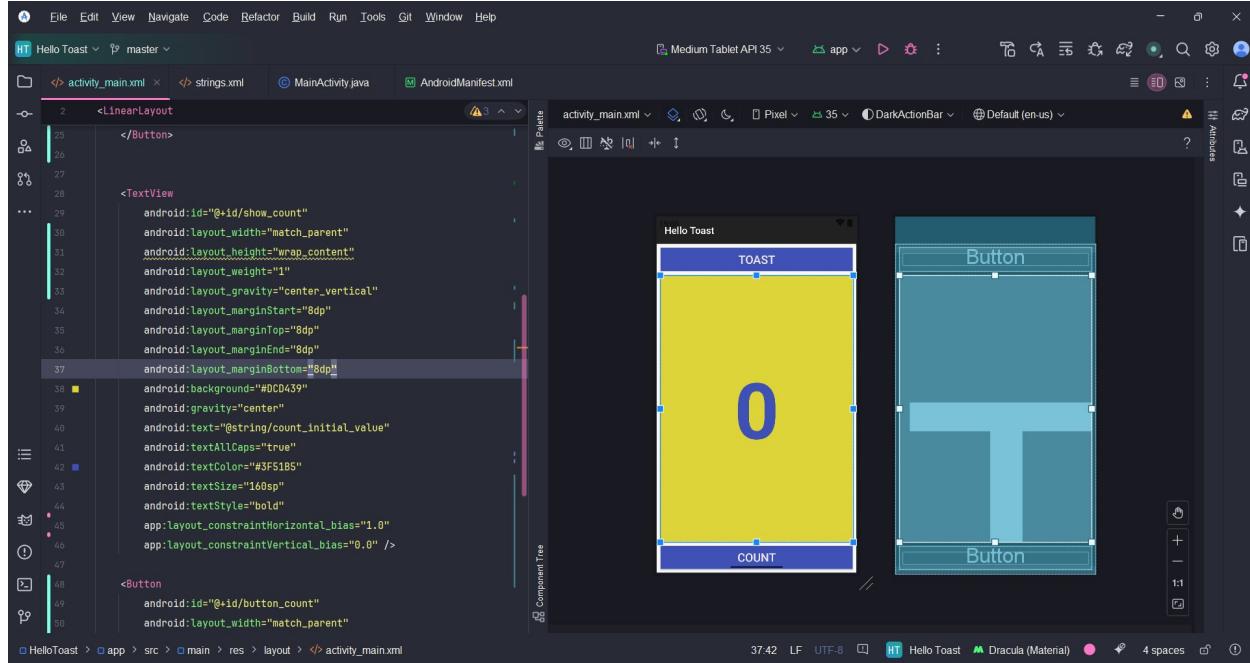
Chỉ định các thuộc tính trọng lực và trọng lượng cung cấp cho bạn khả năng kiểm soát bổ sung đối với việc sắp xếp chế độ xem và nội dung trong LinearLayout.

android :gravity chỉ định sự căn chỉnh nội dung của một View trong chính View đó. Trong bài học trước, bạn đã đặt thuộc tính này cho TextView show_count để căn giữa nội dung (chữ số 0) vào giữa Text View :

Thuộc tính android:layout_weight cho biết bao nhiêu không gian bổ sung trong LinearLayout sẽ được phân bổ cho View . Nếu chỉ có một View có thuộc tính này, nó sẽ nhận được toàn bộ không gian màn hình bổ sung. Đối với nhiều phần tử View , không gian được chia theo tỷ lệ. Ví dụ, nếu mỗi phần tử Button có trọng số là 1 và TextView là 2, tổng cộng là 4, thì mỗi phần tử Button sẽ nhận được $\frac{1}{4}$ không gian và TextView sẽ nhận được một nửa.

Trên các thiết bị khác nhau, bộ cục có thể hiển thị phần tử TextView show_count lấp đầy một phần hoặc hầu hết khoảng trống giữa các nút Toast và Count . Để mở rộng TextView

để lấp đầy khoảng trống khả dụng bất kể thiết bị nào được sử dụng, hãy chỉ định thuộc tính android:gravity cho TextView .



Nhiệm vụ 3: Thay đổi bố cục thành RelativeLayout

MỘT [RelativeLayout](#) là một nhóm ché độ xem trong đó mỗi ché độ xem được định vị và căn chỉnh tương đối với các ché độ xem khác trong nhóm. Trong nhiệm vụ này, bạn sẽ học cách xây dựng bố cục bằng RelativeLayout.

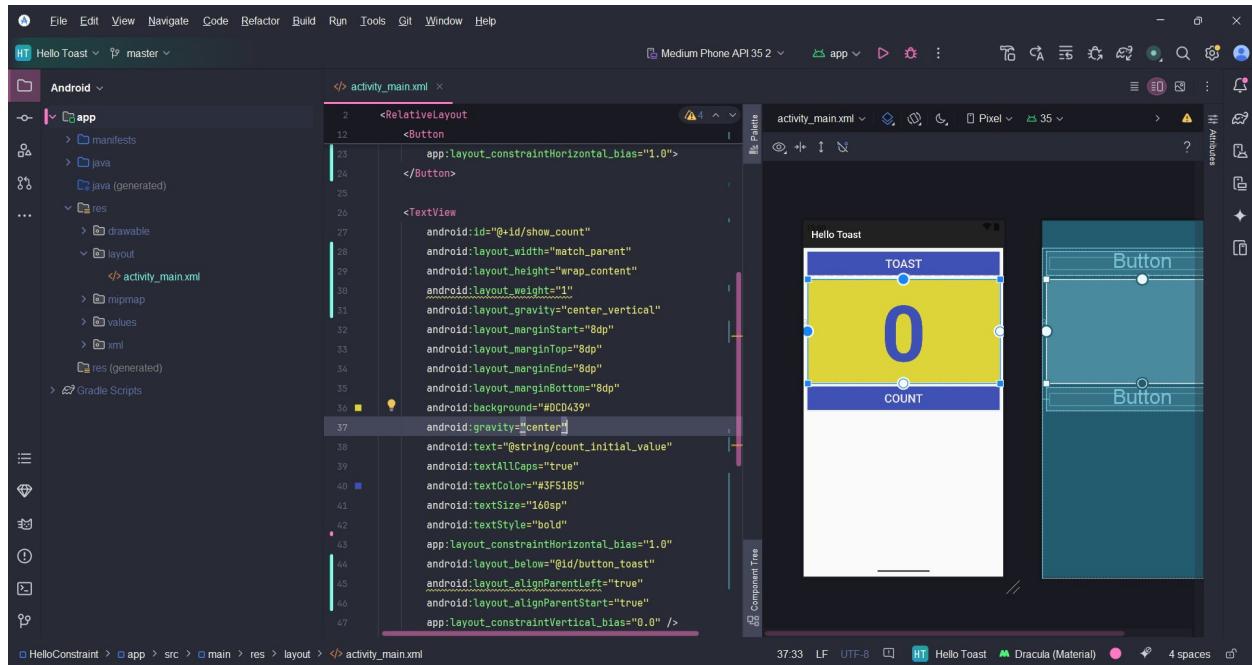
3.1 Đổi LinearLayout thành RelativeLayout

Một cách dễ dàng để thay đổi LinearLayout thành RelativeLayout là thêm thuộc tính XML vào tab Text .

1. Mở tệp bố cục **activity_main.xml** và nhập vào tab **Văn bản** ở cuối ngăn chỉnh sửa để xem mã XML.

2. Thay đổi <LinearLayout> ở trên cùng thành <**RelativeLayout**> để câu lệnh trông như thế này:
3. Cuộn xuống để đảm bảo rằng thẻ kết thúc </LinearLayout> cũng đã đổi thành </RelativeLayout>; nếu chưa, hãy thay đổi thủ công.

3.2 Sắp xếp lại chế độ xem trong RelativeLayout



Bản 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, hãy nhấp vào nút **Editor** ở thanh công cụ trên cùng và chọn **Chuyển sang chế độ ngang**. Chọn **Chuyển sang chế độ dọc** để trở về chế độ dọc.

- Để tạo biến thể của bộ cục khác với hướng ngang, hãy nhấp vào nút **Định hướng trong Trình chỉnh sửa** và chọn **Tạo Biến thể theo Phong cảnh**. Một trình chỉnh sửa mới cửa sổ mở ra với tab **land/activity_main.xml** hiển thị bộ cục cho hướng ngang (ngang).
- Để xem trước bộ cục cho các thiết bị khác nhau mà không cần phải chạy ứng dụng trên thiết bị hoặc trình giả lập, nhấp vào nút **Nexus 5** → **Thiết bị trong Trình chỉnh sửa** ở thanh công cụ trên cùng và chọn một thiết bị.

Để tạo biến thể của bộ cục khác biệt cho máy tính bảng (màn hình lớn hơn), hãy nhấp vào nút **Định hướng trong Trình chỉnh sửa** và chọn **Tạo biến thể bộ cục x-lớn**. Một trình chỉnh sửa mới cửa sổ mở ra với tab **xlarge/activity_main.xml** hiển thị bộ cục cho thiết bị có kích thước bảng máy tính bảng.

Sử dụng ConstraintLayout :

- Để xóa tất cả các ràng buộc trong một bộ cục với gốc ConstraintLayout , hãy nhấp vào nút **Xóa tất cả ràng buộc** trên thanh công cụ.
- Bạn có thể căn chỉnh một phần tử UI có chứa văn bản, chẳng hạn như TextView hoặc Button , với một phần tử UI khác có chứa văn bản. *Ràng buộc đường cơ sở* cho phép bạn ràng buộc các phần tử sao cho đường cơ sở văn bản khớp với nhau.
- Để tạo ràng buộc đường cơ sở, hãy di con trỏ của bạn qua phần tử UI cho đến khi đường cơ sở nút ràng buộc xuất hiện bên dưới phần tử.

- Nút đóng gói  trên thanh công cụ cung cấp các tùy chọn để đóng gói hoặc mở rộng các thành phần UI đã chọn. Bạn có thể sử dụng nó để sắp xếp đều các thành phần Nút theo chiều ngang trên toàn bộ bố cục.

Sử dụng LinearLayout :

- [LinearLayout](#) là một [ViewGroup](#) sắp xếp tập hợp các chế độ xem theo hàng ngang hoặc hàng dọc.
- LinearLayout phải có các thuộc tính `layout_width`, `layout_height` và `orientation`.
- `match_parent` cho `layout_width` hoặc `layout_height` : Mở rộng View để lấp đầy view cha theo chiều rộng hoặc chiều cao. Khi LinearLayout là View gốc, nó sẽ mở rộng đến kích thước của màn hình (View cha).
- `Wrap_content` cho `layout_width` hoặc `layout_height` : Thu nhỏ kích thước để View đủ lớn để bao quanh nội dung của nó. Nếu không có nội dung, View sẽ trở nên vô hình.
- Số dp cố định ([pixel không phụ thuộc mật độ](#)) cho `layout_width` hoặc `layout_height` : Chỉ định kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị. Ví dụ: 16dp nghĩa là 16 pixel không phụ thuộc mật độ.
- Hướng của LinearLayout có thể là theo chiều ngang để sắp xếp các thành phần từ trái sang phải hoặc theo chiều dọc để sắp xếp các thành phần từ trên xuống dưới.
- Chỉ định các thuộc tính trọng lực và trọng lượng cung cấp cho bạn khả năng kiểm soát bổ sung đối với việc sắp xếp chế độ xem và nội dung trong LinearLayout .

- android :gravity chỉ định cách căn chỉnh nội dung của một View trong chính View đó.
- Thuộc tính android:layout_weight chỉ ra lượng không gian bổ sung trong LinearLayout sẽ được phân bổ cho View . Nếu chỉ có một View có thuộc tính này, nó sẽ nhận được tất cả không gian màn hình bổ sung. Đối với nhiều phần tử View , không gian được tính theo tỷ lệ. Ví dụ, nếu hai phần tử Button mỗi phần tử có trọng số là 1 và TextView là 2, tổng cộng là 4, các phần tử Button mỗi bên nhận được $\frac{1}{4}$ không gian và một nửa TextView .

Sử dụng RelativeLayout :

- MỘT [RelativeLayout](#) là ViewGroup trong đó mỗi chế độ xem được định vị và căn chỉnh tương đối với các chế độ xem khác trong nhóm.
- Sử dụng android:layout_alignParentTop để căn chỉnh View lên trên cùng của phần tử cha.
- Sử dụng android:layout_alignParentLeft để căn chỉnh View về phía bên trái của phần tử cha.
- Sử dụng android:layout_alignParentStart để làm cho cạnh bắt đầu của View khớp với cạnh bắt đầu của parent. Thuộc tính này hữu ích nếu bạn muốn ứng dụng của mình hoạt động trên các thiết bị sử dụng ngôn ngữ hoặc tùy chọn địa phương khác nhau. Điểm *bắt đầu* là cạnh trái của màn hình nếu tùy chọn là từ trái sang phải hoặc là cạnh phải của màn hình nếu tùy chọn là từ phải sang trái.

1.3) Trình chỉnh sửa bố cục

Giới thiệu

Các [Lớp TextView](#) là một 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 theo chiều dọc.

Nếu có nhiều thông tin hơn mức hiển thị trên màn hình 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 trái.

Bạn thường sử dụng chế độ xem cuộn cho các tin tức, bài viết hoặc bất kỳ văn bản dài nào không vừa hoàn toàn 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 thành phần UI (như trường văn bản và nút) trong chế độ xem cuộn.

Các [Lớp ScrollView](#) cung cấp bố cục cho chế độ xem cuộn. ScrollView là lớp con của [FrameLayout](#) . Chỉ đặt *một* chế độ xem làm con bên trong nó—chế độ xem con chứa toàn bộ nội dung cuộn. Chế độ xem con này có thể tự nó là một [ViewGroup](#) (chẳng hạn như [LinearLayout](#)) chứa các thành phần UI.

Bố cục phức tạp có thể gặp phải các vấn đề về hiệu suất với các chế độ xem con như hình ảnh. Một lựa chọn tốt cho Chế độ xem trong ScrollView là LinearLayout được sắp xếp theo hướng dọc, trình bày các mục mà người dùng có thể cuộn qua (chẳng hạn như các phần tử TextView).

Với ScrollView , tất cả các thành phần UI đều nằm trong bộ nhớ và trong hệ thống phân cấp chế độ xem ngay cả khi chúng không được hiển thị trên màn hình. Điều này làm cho ScrollView trở nên lý tưởng để cuộn các trang văn bản dạng tự do một cách mượt mà, vì văn bản đã nằm trong bộ nhớ. Tuy nhiên, ScrollView có thể sử dụng rất nhiều bộ nhớ, điều này có thể ảnh hưởng đến hiệu suất của phần còn lại của ứng dụng. Để hiển thị danh sách dài các mục mà người dùng có thể thêm vào, xóa khỏi hoặc chỉnh sửa, hãy cân nhắc sử dụng [RecyclerView](#) được mô tả trong một bài học riêng.

Tổng quan về ứng dụng

Ứng dụng Scrolling Text thể hiện [Thành phần UI ScrollView](#). ScrollView là một ViewGroup trong ví dụ này chứa một TextView . Nó hiển thị một trang văn bản dài—trong trường hợp này là bài đánh giá album nhạc—mà người dùng có thể cuộn theo chiều dọc để đọc bằng cách vuốt lên và xuống. Một thanh cuộn xuất hiện ở lề phải. Ứng dụng cho thấy cách bạn có thể sử dụng văn bản được định dạng bằng các thẻ HTML tối thiểu để đặt văn bản thành in đậm hoặc in nghiêng và với các ký tự xuống dòng để phân tách các đoạn văn. Bạn cũng có thể bao gồm các liên kết web đang hoạt động trong văn bản.

Nhiệm vụ 1: Thêm và chỉnh sửa các thành phần TextView

Trong bài thực hành này, bạn sẽ tạo một dự án Android cho ứng dụng ScrollingText, thêm các thành phần TextView vào bố cục cho tiêu đề và phụ đề bài viết, và thay đổi thành phần TextView “Hello World” hiện có để hiển thị một bài viết dài. Hình bên dưới là sơ đồ bố cục.

Bạn sẽ thực hiện tất cả những thay đổi này trong mã XML và trong tệp strings.xml . Bạn sẽ chỉnh sửa mã XML cho bố cục trong ngăn Văn bản, mà bạn hiển thị bằng cách nhấp vào tab **Văn bản** , thay vì nhấp vào tab **Thiết kế** cho ngăn Thiết kế. Một số thay đổi đối với các thành phần và thuộc tính UI dễ thực hiện hơn trực tiếp trong ngăn Văn bản bằng cách sử dụng mã nguồn XML.

1.1 Tạo dự án và các thành phần TextView

Trong nhiệm vụ này, bạn sẽ tạo dự án và các thành phần TextView , đồng thời sử dụng các thuộc tính TextView để tạo kiểu cho văn bản và nền.

2. Trong thư mục **app > res > layout** trong ngăn **Project > Android** , hãy mở tệp **activity_main.xml** và nhấp vào tab **Text** để xem mã XML.

Ở trên cùng hoặc gốc của phân cấp View là [ConstraintLayoutViewGroup](#) :

ViewGroup này thành [RelativeLayout](#) . Dòng mã thứ hai bây giờ trông giống như thế này: RelativeLayout cho phép bạn đặt các thành phần UI tương đối với nhau hoặc tương đối với chính RelativeLayout cha .

TextView “Hello World” mặc định được tạo bởi mẫu Empty Layout vẫn có các thuộc tính ràng buộc (chẳng hạn như

app:layout_constraintBottom_toBottomOf="parent"). Đừng lo lắng—you sẽ xóa chúng ở bước tiếp theo.

5. Thêm phần tử TextView phía trên TextView “Hello World” bằng cách nhập <TextView . Một khung TextView xuất hiện kết thúc bằng /> và hiển thị các thuộc tính layout_width và layout_height , là những thuộc tính bắt buộc đối với TextView .
6. Nhập các thuộc tính sau cho TextView . Khi bạn nhập từng thuộc tính và giá trị, các gợi ý sẽ xuất hiện để hoàn thiện tên hoặc giá trị thuộc tính.



```
22     TextView
23         android:layout_width="match_parent"
24         android:layout_height="wrap_content"
25         android:id="@+id/article_subheading"
26         android:layout_below="@+id/article_heading"
27         android:padding="@dimen/padding_regular"
28         android:textAppearance="@android:style/TextAppearance.DeviceDefault"
29         android:text="Article Subtitle">
30     /TextView>
```

7. Trích xuất tài nguyên chuỗi cho chuỗi được mã hóa cứng "Article Title" của thuộc tính android:text trong TextView để tạo một mục cho chuỗi đó trong **strings.xml** .

Đặt con trỏ vào chuỗi được mã hóa cứng, nhấn Alt-Enter (Option-Enter trên máy Mac) và chọn **Extract string resource** . Đảm bảo rằng tùy chọn **Create the resource in directories** được chọn, sau đó chỉnh sửa tên tài nguyên cho giá trị chuỗi thành **article_title** .

Tài nguyên chuỗi được mô tả chi tiết trong [Tài nguyên chuỗi](#) .

8. Trích xuất tài nguyên kích thước cho chuỗi được mã hóa cứng "10dp" của thuộc tính android:padding trong TextView để tạo dimens.xml và thêm mục vào đó.

Đặt con trỏ vào chuỗi được mã hóa cứng, nhấn Alt-Enter (Option-Enter trên máy Mac) và chọn **Extract dimension resource**. Đảm bảo rằng tùy chọn **Create the resource in directories** được chọn, sau đó chỉnh sửa Resource name thành **padding_regular**.

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

```
32     <TextView  
33         android:id="@+id/article"  
34         android:layout_width="match_parent"  
35         android:layout_height="wrap_content"  
36         android:layout_below="@+id/article_subheading"  
37         android:lineSpacingExtra="@dimen/line_spacing"  
38         android:padding="@dimen/padding_regular"  
39         android:text="@string/article_text">  
40     </TextView>  
41 
```

```
21     <TextView  
22         android:id="@+id/article_subheading"  
23         android:layout_width="match_parent"  
24         android:layout_height="wrap_content"  
25         android:layout_below="@+id/article_heading"  
26         android:padding="@dimen/padding_regular"  
27         android:text="@string/article_subtitle"  
28         android:textAppearance="@android:style/TextAppearance.DeviceDefault"  
29     </TextView>  
30 
```

1.2 Thêm nội dung bài viết

Trong một ứng dụng thực tế truy cập vào các bài báo hoặc tạp chí, các bài viết xuất hiện có thể sẽ đến từ một nguồn trực tuyến thông qua nhà cung cấp nội dung hoặc có thể được lưu trước 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.

1. Trong thư mục **app > res > values** , mở **strings.xml** .
2. Mở bất kỳ tệp văn bản nào có nhiều văn bản hoặc mở [Tệp strings.xml của ứng dụng ScrollingText đã hoàn thiện](#) .
3. Nhập các giá trị cho chuỗi article_title và article_subtitle với tiêu đề và phụ đề đã tạo hoặc sử dụng các giá trị trong tệp strings.xml của ứng dụng ScrollingText đã hoàn thành. Làm cho các giá trị chuỗi thành văn bản một dòng không có thẻ HTML hoặc nhiều dòng.
4. Nhập hoặc sao chép và dán văn bản cho chuỗi article_text .

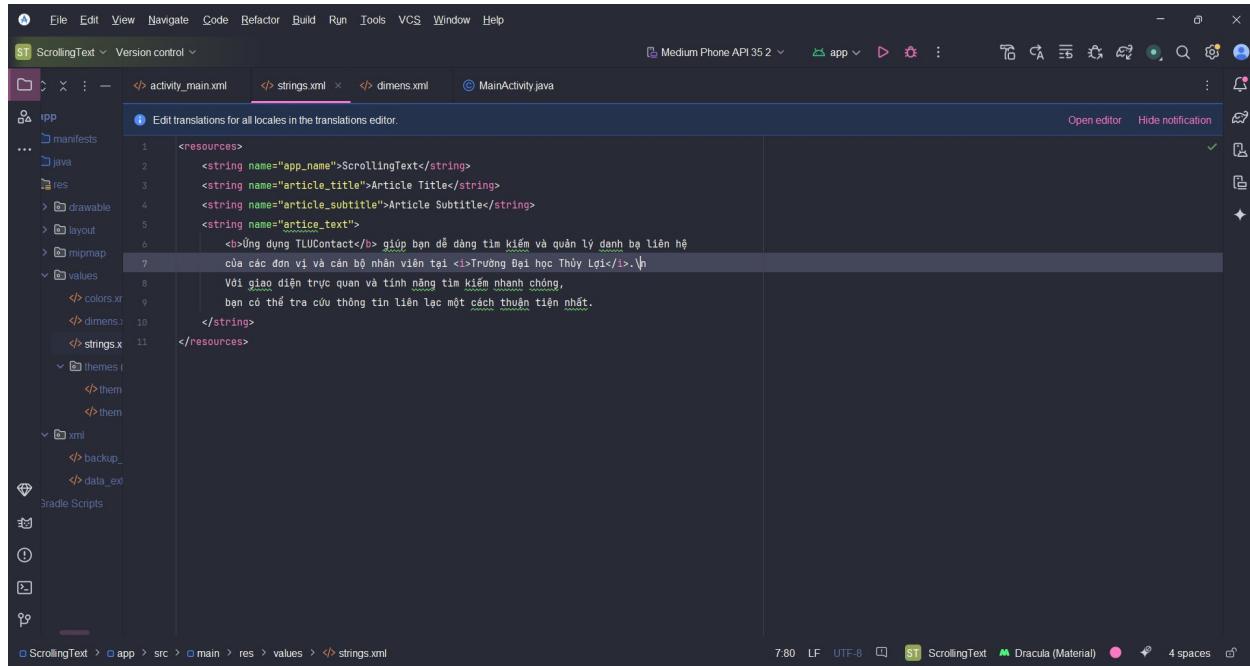
Bạn có thể sử dụng văn bản trong tệp văn bản của mình hoặc sử dụng văn bản được cung cấp cho chuỗi article_text trong tệp strings.xml của ứng dụng ScrollingText đã hoàn thành. Yêu cầu duy nhất cho tác vụ này là văn bản phải đủ dài để không vừa với màn hình.

Hãy ghi nhớ những điều sau (tham khảo hình bên dưới để biết ví dụ):

- Khi bạn nhập hoặc dán văn bản vào tệp strings.xml , các dòng văn bản không bao quanh dòng tiếp theo—chúng mở rộng ra ngoài lề phải. Đây là hành vi đúng—mỗi dòng văn bản mới bắt đầu từ lề trái đại diện cho toàn bộ một đoạn văn. Nếu bạn muốn văn bản trong strings.xml được bao quanh, bạn có thể nhấn Return để nhập các dòng kết thúc cứng hoặc định dạng văn bản đầu tiên trong trình soạn thảo văn bản với kết thúc bằng dòng cứng.

Nhập **\n** để biểu thị kết thúc một dòng và **\n khác** để biểu thị một dòng trống. Bạn cần thêm các ký tự kết thúc dòng để các đoạn văn không bị chồng lên nhau.

- Nếu bạn có dấu nháy đơn (') trong văn bản của mình, bạn phải thoát khỏi nó bằng cách đặt trước nó một dấu gạch chéo ngược (\'). Nếu bạn có dấu ngoặc kép trong văn bản của mình, bạn cũng phải thoát khỏi nó (\"). Bạn cũng phải thoát khỏi bất kỳ ký tự nào khác không phải ASCII. Xem Phần [định dạng và kiểu dáng của Chuỗi tài nguyên](#) để biết thêm chi tiết.
- Nhập thẻ HTML **** và **** xung quanh các từ cần in đậm.
- Nhập thẻ HTML **<i>** và **</i>** xung quanh các từ cần in nghiêng. Nếu bạn sử dụng dấu nháy đơn cong trong cụm từ in nghiêng, hãy thay thế chúng bằng dấu nháy đơn thẳng.
- Bạn có thể kết hợp chữ in đậm và chữ in nghiêng bằng cách kết hợp các thẻ, như trong **<i>... từ... </i>**. Các thẻ HTML khác sẽ bị bỏ qua.
- Đặt toàn bộ văn bản trong **<string name="article_text"> </string>** trong tệp strings.xml .
- Bao gồm một liên kết web để kiểm tra, chẳng hạn như **www.google.com** . (Ví dụ bên dưới sử dụng **www.rockument.com**.) Không sử dụng thẻ HTML, vì bất kỳ thẻ HTML nào ngoại trừ chữ in đậm và các thẻ in nghiêng sẽ bị bỏ qua và hiển thị dưới dạng văn bản, đây không phải là điều bạn muốn.



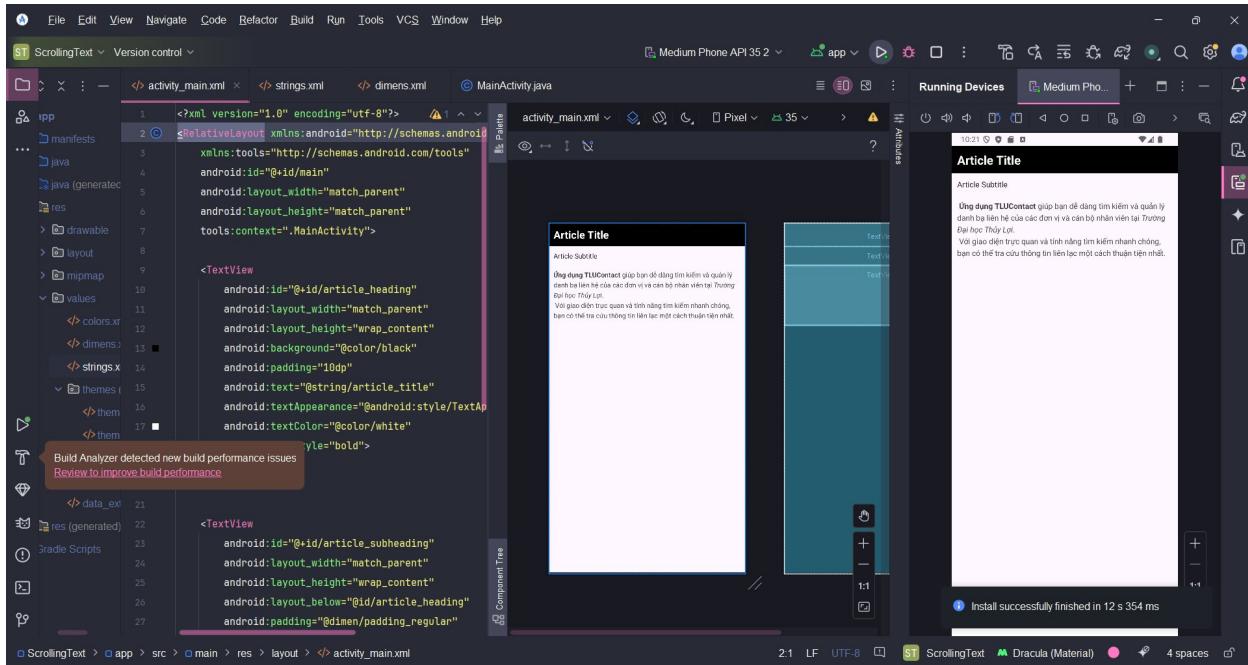
```

<resources>
    <string name="app_name">ScrollingText</string>
    <string name="article_title">Article Title</string>
    <string name="article_subtitle">Article Subtitle</string>
    <string name="article_text">
        <b>Ứng dụng TLUContact</b> giúp bạn dễ dàng tìm kiếm và quản lý danh bạ liên hệ
        của các đơn vị và cán bộ nhân viên tại <i>Trường Đại học Thủy Lợi</i>.\n
        Với giao diện trực quan và tính năng tìm kiếm nhanh chóng,
        bạn có thể tra cứu thông tin liên lạc một cách thuận tiện nhất.
    </string>
</resources>

```

1.3 Chạy ứng dụng

Chạy ứng dụng. Bài viết xuất hiện, nhưng người dùng không thể cuộn bài viết vì bạn chưa đưa vào ScrollView (bạn sẽ thực hiện trong tác vụ tiếp theo). Lưu ý rằng việc chạm vào liên kết web hiện không có tác dụng gì. Bạn cũng sẽ sửa lỗi đó trong tác vụ tiếp theo.



Nhiệm vụ 2: Thêm ScrollView và liên kết web đang hoạt động

Trong tác vụ trước, bạn đã tạo ứng dụng ScrollingText với các thành phần TextView cho tiêu đề bài viết, phụ đề và văn bản bài viết dài. Bạn cũng đã bao gồm một liên kết web, nhưng liên kết vẫn chưa hoạt động. Bạn sẽ thêm mã để kích hoạt liên kết.

Ngoài ra, TextView tự nó không thể cho phép người dùng cuộn văn bản 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 cho các liên kết web đang hoạt động

Thêm thuộc tính android:autoLink="web" vào TextView của bài viết. Mã XML cho mục này

TextView hiện trông như thế này:

2.2 Thêm ScrollView vào bộ cục

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

1. Thêm ScrollView giữa TextView article_subheading và TextView article . Khi bạn nhập <ScrollView , Android Studio sẽ tự động thêm </ScrollView> vào cuối và hiển thị các thuộc tính android:layout_width và android:layout_height với các gợi ý.
2. Chọn **wrap_content** từ các gợi ý cho cả hai thuộc tính.
3. mã kết thúc </ScrollView> sau TextView của bài viết để các thuộc tính TextView của bài viết nằm hoàn toàn bên trong ScrollView .
4. Xóa thuộc tính sau khỏi TextView của bài viết và thêm vào ScrollView :

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

Bài viết nằm bên trong phần tử ScrollView .

5. Chọn **Mã > Định dạng lại mã** để định dạng lại mã XML sao cho TextView của bài viết giờ đây xuất hiện thực lè bên trong mã <ScrollView .

6. Nhấp vào tab **Xem trước** ở bên phải trình chỉnh sửa bố cục để xem bản xem trước của bố cục.



```
16    </TextView>
17    <TextView
18        android:id="@+id/article_subheading"
19        android:layout_width="match_parent"
20        android:layout_height="wrap_content"
21        android:layout_below="@+id/article_heading"
22        android:padding="@dimen/padding_regular"
23        android:text="@string/article_subtitle"
24        android:textAppearance="@android:style/TextAppearance.DeviceDefault">
25    </TextView>
26    <ScrollView
27        android:layout_width="wrap_content"
28        android:layout_height="wrap_content"
29        android:layout_below="@+id/article_subheading">
30        <TextView
31            android:id="@+id/article"
32            android:layout_width="wrap_content"
33            android:layout_height="wrap_content"
34            android:layout_margin="16dp"
35            android:autoLink="web"
36            android:lineSpacingExtra="@dimen/line_spacing"
37            android:padding="@dimen/padding_regular"
38            android:text="@string/article_text">
39        </TextView>
40    </ScrollView>
41
```

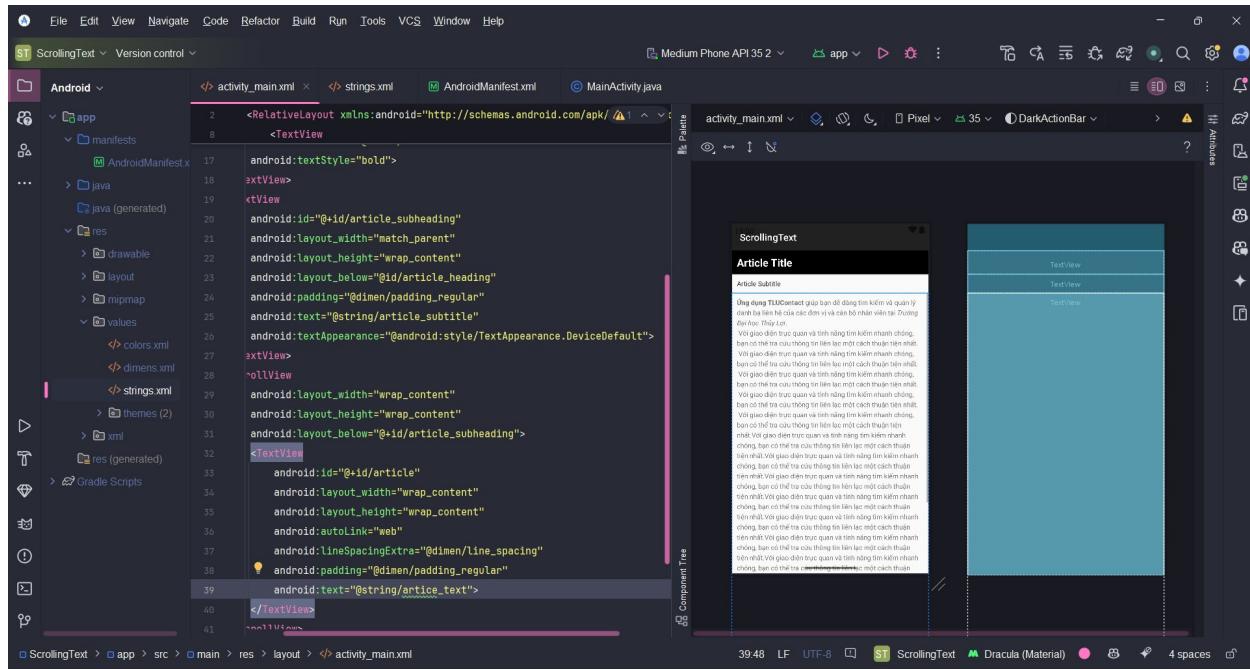
2.3 Chạy ứng dụng

1. Chạy ứng dụng trên thiết bị hoặc trình giả lập.

Vuốt lên và xuống để cuộn bài viết. Thanh cuộn sẽ xuất hiện ở lề phải khi bạn cuộn.

Nhấn vào liên kết web để đi đến trang web. Thuộc tính android:autoLink biến bất kỳ URL nào có thể nhận dạng được trong TextView (chẳng hạn như www.rockument.com) thành liên kết web.

2. Xoay thiết bị hoặc trình giả lập của bạn trong khi chạy ứng dụng. Lưu ý cách chế độ xem cuộn mở rộng để sử dụng toàn màn hình và vẫn cuộn đúng cách.
3. Chạy ứng dụng trên máy tính bảng hoặc trình giả lập máy tính bảng. Lưu ý cách chế độ xem cuộn mở rộng để sử dụng toàn màn hình và vẫn cuộn đúng cách.



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 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 *trong* ScrollView, do đó cuộn mọi thứ bên trong LinearLayout.

Ví dụ, nếu bạn muốn tiêu đề phụ của bài viết cuộn cùng với bài viết, hãy thêm LinearLayout trong ScrollView và di chuyển tiêu đề phụ và bài viết vào LinearLayout. LinearLayout trở thành View con duy nhất trong ScrollView như thể hiện trong hình bên dưới và người dùng có thể cuộn toàn bộ LinearLayout: tiêu đề phụ và bài viết.

3.1 Thêm LinearLayout vào ScrollView

- Mở tệp **activity_main.xml** của dự án ứng dụng ScrollingText và chọn tab **Văn bản** để chỉnh sửa mã XML (nếu chưa chọn).

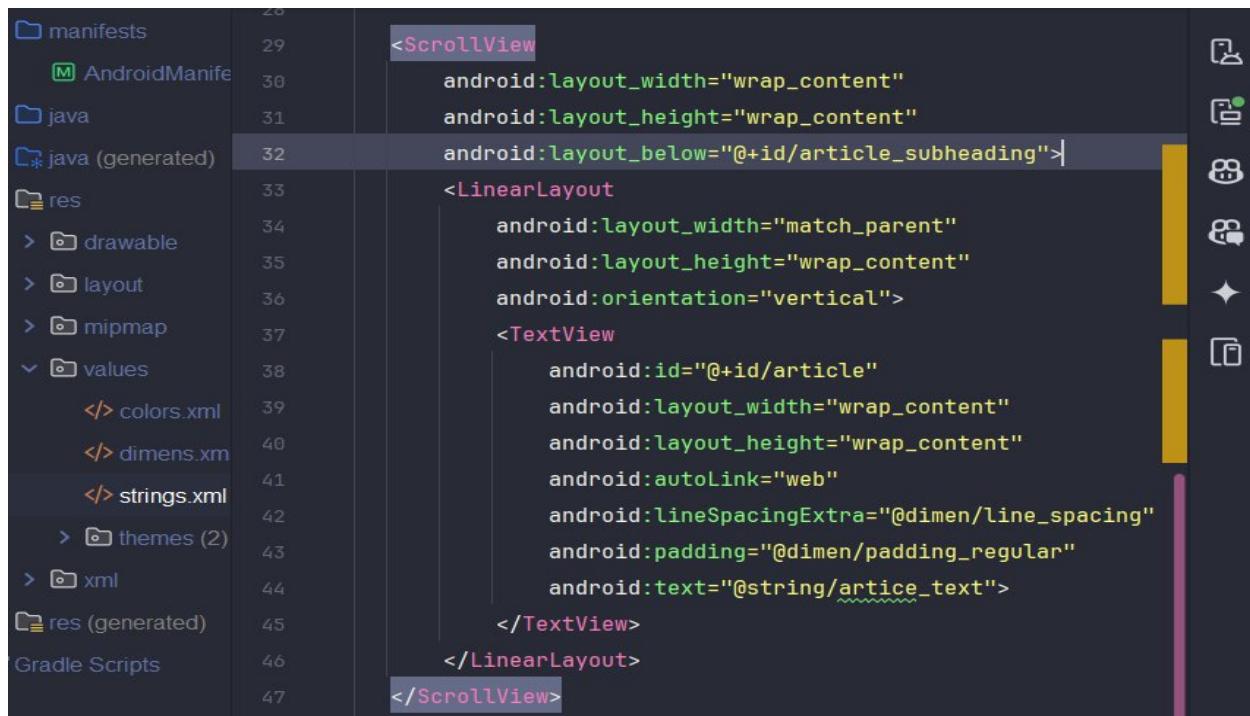
2. Thêm LinearLayout phía trên TextView của bài viết trong ScrollView. Khi bạn nhập <LinearLayout>, Android Studio tự động thêm </LinearLayout> vào cuối và trình bày các thuộc tính android:layout_width và android:layout_height với các gợi ý. Chọn **match_parent** và **wrap_content** từ các gợi ý cho chiều rộng và chiều cao của nó.

Bạn sử dụng match_parent để khớp với chiều rộng của ViewGroup cha. Bạn sử dụng wrap_content để thay đổi kích thước LinearLayout sao cho nó đủ lớn để bao quanh nội dung của nó.

3. mã kết thúc </LinearLayout> sau TextView của bài viết nhưng trước mã đóng </ScrollView> .

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

4. Thêm thuộc tính android:orientation="vertical" vào LinearLayout để đặt hướng của nó theo chiều đọc.
5. Chọn **Mã > Định dạng lại mã** để thuần lè mã đúng cách.



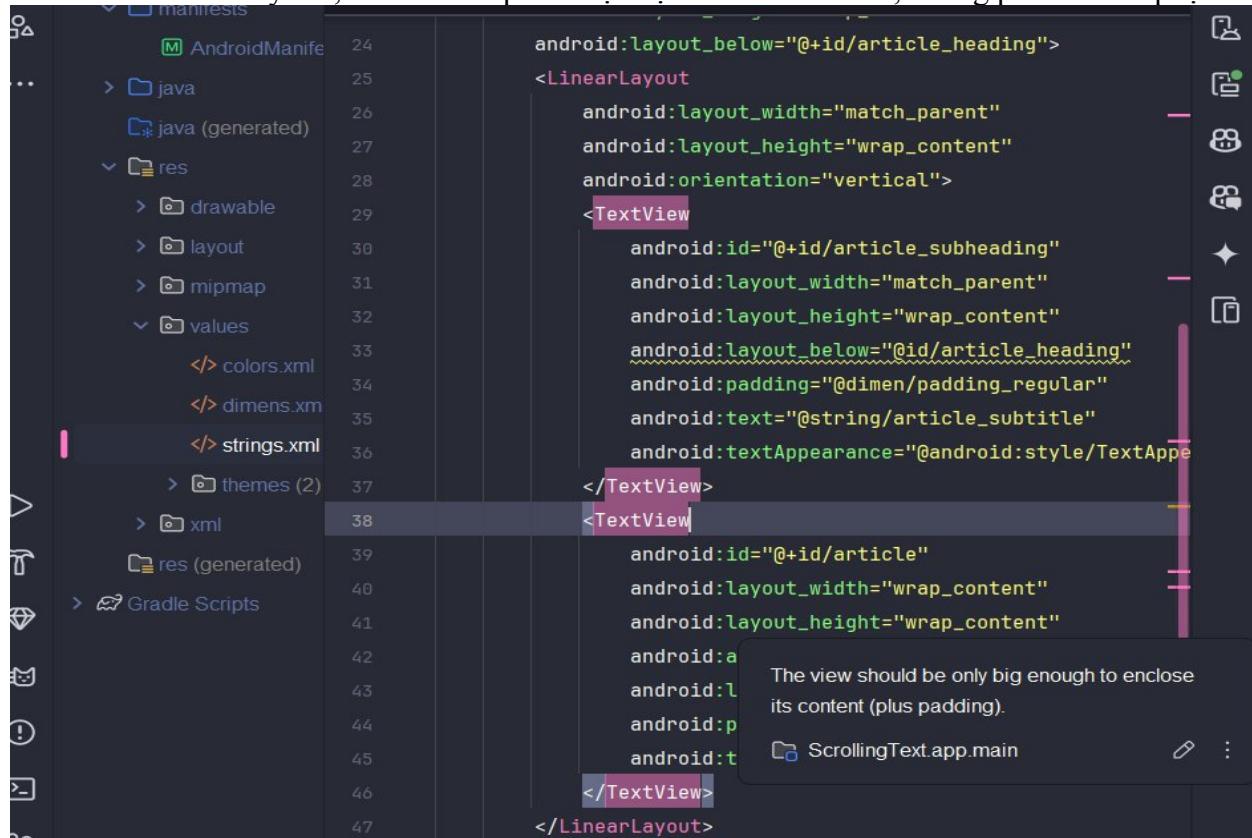
The screenshot shows the Android Studio code editor with the XML layout file for a ScrollView. The ScrollView contains a LinearLayout with a TextView inside. The TextView has several attributes set: android:id="@+id/article", android:layout_width="wrap_content", android:layout_height="wrap_content", android:autoLink="web", android:lineSpacingExtra="@dimen/line_spacing", android:padding="@dimen/padding_regular", and android:text="@string/article_text". The LinearLayout also has attributes: android:layout_width="match_parent", android:layout_height="wrap_content", and android:orientation="vertical". The code editor's sidebar shows the project structure with files like manifests, Java, res, and values. The code is numbered from 28 to 47.

```
<ScrollView>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <TextView
            android:id="@+id/article"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:autoLink="web"
            android:lineSpacingExtra="@dimen/line_spacing"
            android:padding="@dimen/padding_regular"
            android:text="@string/article_text">
    </LinearLayout>
</ScrollView>
```

3.2 Di chuyển các thành phần UI trong LinearLayout

LinearLayout hiện chỉ có một phần tử UI— textView bài viết . Bạn muốn đưa textView article_subheading vào LinearLayout để cả hai đều cuộn được .

1. Để di chuyển TextView của tiêu đề phụ article , hãy chọn mã, chọn **Chỉnh sửa > Cắt** , nhấp vào phía trên TextView của article bên trong LinearLayout và chọn **Chỉnh sửa > Dán** .
2. Xóa thuộc tính android:layout_below="@+id/article_heading" khỏi article_subheading TextView . Vì TextView này hiện nằm trong LinearLayout , thuộc tính này sẽ xung đột với các thuộc tính LinearLayout .
3. Thay đổi thuộc tính bố cục ScrollView từ android:layout_below="@+id/article_subheading" thành android:layout_below="@+id/article_heading" . Bây giờ tiêu đề phụ là một phần của LinearLayout , ScrollView phải được đặt bên dưới tiêu đề , không phải tiêu đề phụ.



```
24        android:layout_below="@+id/article_heading">
25            <LinearLayout
26                android:layout_width="match_parent"
27                android:layout_height="wrap_content"
28                android:orientation="vertical">
29                    <TextView
30                        android:id="@+id/article_subheading"
31                        android:layout_width="match_parent"
32                        android:layout_height="wrap_content"
33                        android:layout_below="@+id/article_heading"
34                        android:padding="@dimen/padding_regular"
35                        android:text="@string/article_subtitle"
36                        android:textAppearance="@android:style/TextAppea
37                    </TextView>
38                    <TextView
39                        android:id="@+id/article"
40                        android:layout_width="wrap_content"
41                        android:layout_height="wrap_content"
42                        android:a
43                        android:l
44                        android:p
45                        android:t
46                    </TextView>
47            </LinearLayout>
```

The view should be only big enough to enclose its content (plus padding).

1.4) Văn bản và các chế độ cuộn

Nhiệm vụ 1: Thay đổi biểu tượng trình khởi chạy

Mỗi ứng dụng mới bạn tạo bằng Android Studio đều bắt đầu bằng một biểu tượng trình khởi chạy chuẩn đại diện cho ứng dụng. Biểu tượng trình khởi chạy xuất hiện trong danh sách cửa hàng Google Play. Khi người dùng tìm kiếm trên cửa hàng Google Play, biểu tượng cho ứng dụng của bạn sẽ xuất hiện trong kết quả tìm kiếm.

Khi người dùng đã cài đặt ứng dụng, biểu tượng trình khởi chạy sẽ xuất hiện trên thiết bị ở nhiều nơi khác nhau, bao gồm màn hình chính và màn hình Tìm kiếm ứng dụng. Ví dụ, ứng dụng HelloToast xuất hiện trong màn hình Tìm kiếm ứng dụng của trình giả lập với biểu tượng chuẩn cho các dự án ứng dụng mới, như được hiển thị bên dưới.

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 tác 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 [nhà phát triển android.com](https://developer.android.com/).

Tài liệu này chứa rất nhiều thông tin được Google cập nhật thường xuyên.

16. Đi đến <https://developer.android.com/design/>.

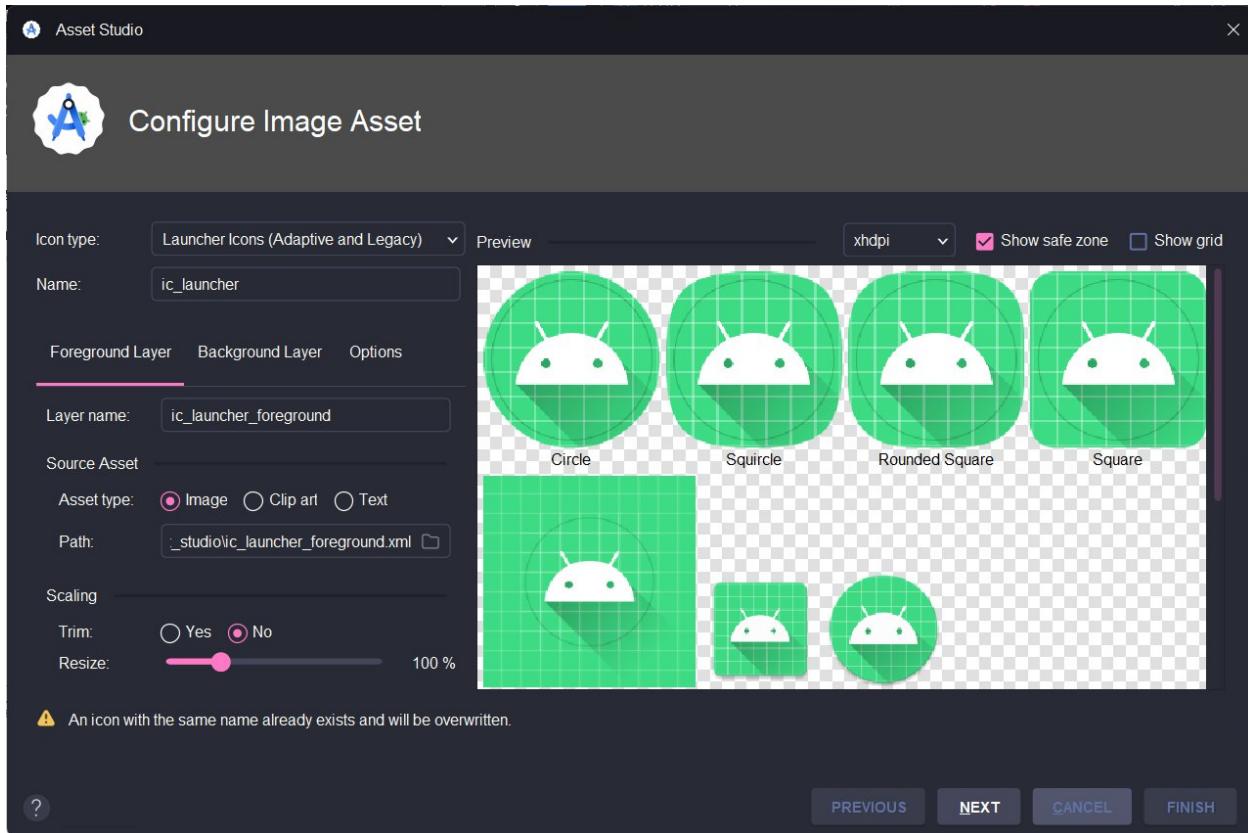
Phần này nói về Material Design, một triết lý thiết kế khái niệm phác thảo cách các ứng dụng nên trông như thế nào và hoạt động ra sao trên các thiết bị di động. Điều hướng các liên kết để tìm hiểu thêm về Material Design. Ví dụ, hãy truy cập [Phản phong cách](#) để tìm hiểu thêm về cách sử dụng màu sắc và các chủ đề khác.

17. Đi đến developer.android.com/docs/ để tìm thông tin về API, tài liệu tham khảo, hướng dẫn, hướng dẫn công cụ và mẫu mã.
18. Đi đến developer.android.com/distribute/ để tìm thông tin về việc đưa ứng dụng lên [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 [kiếm tiền](#).

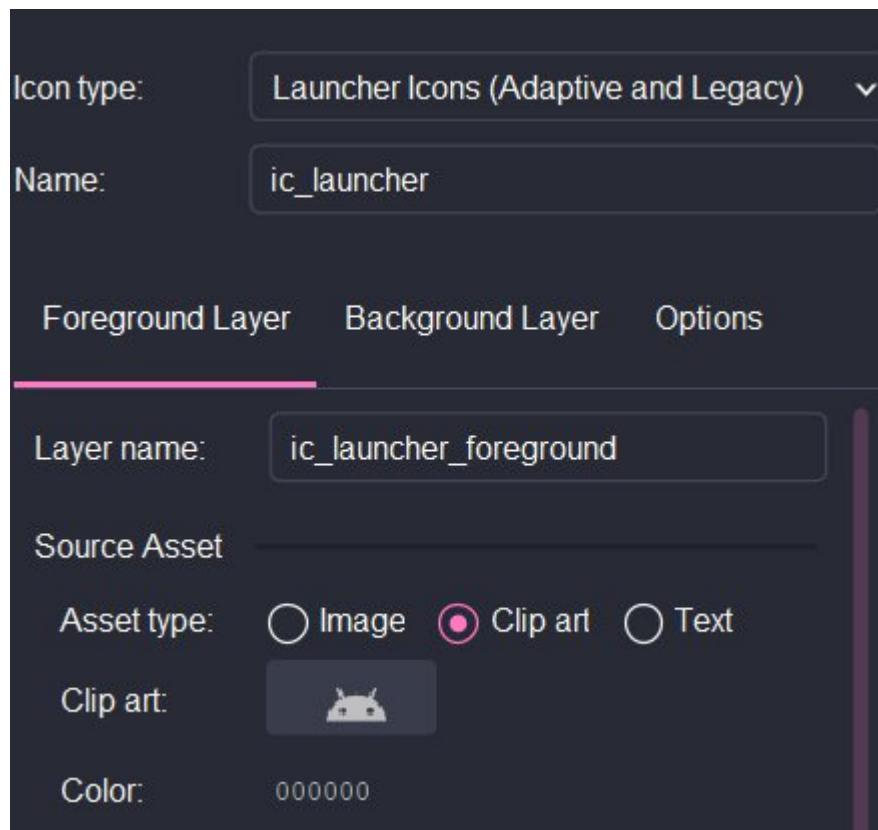
1.2 Thêm nội dung hình ảnh cho biểu tượng trình khởi chạy

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

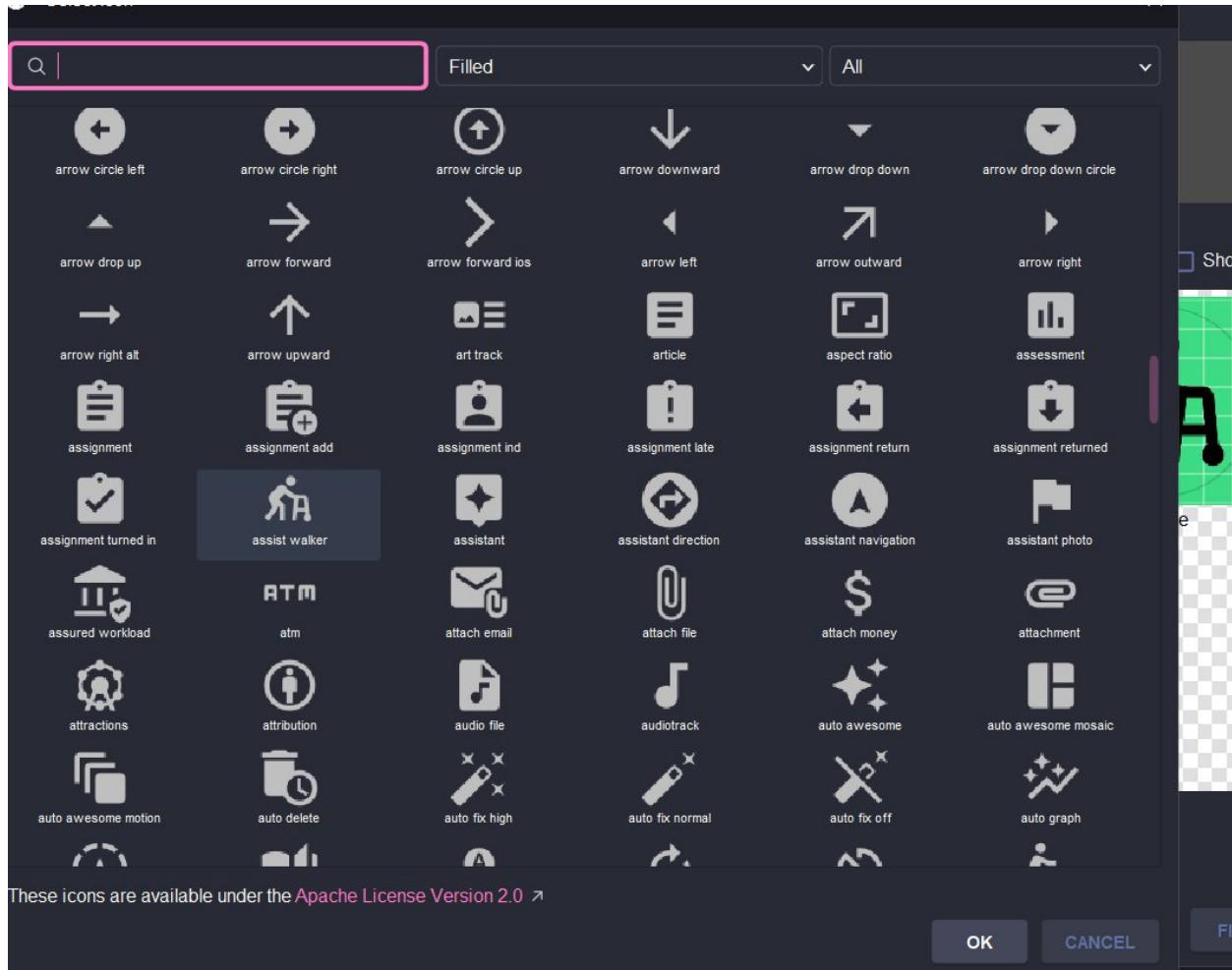
1. Mở dự án ứng dụng HelloToast từ bài học trước về cách sử dụng trình chỉnh sửa bố cục hoặc tạo một dự án ứng dụng mới.
 2. Trong ngăn **Project > Android** , nhấp chuột phải (hoặc giữ Control khi nhấp) vào thư mục **res** và chọn **New > Image Asset** . Cửa sổ Configure Image Asset sẽ xuất hiện.
-



3. Trong trường **Loại biểu tượng** , hãy chọn **Biểu tượng trình khởi chạy (Thích ứng & Cũ)** nếu mục này chưa được chọn.
4. Nhấp vào tab **Lớp nền trước** , chọn **Clip Art** cho **Loại tài sản** .

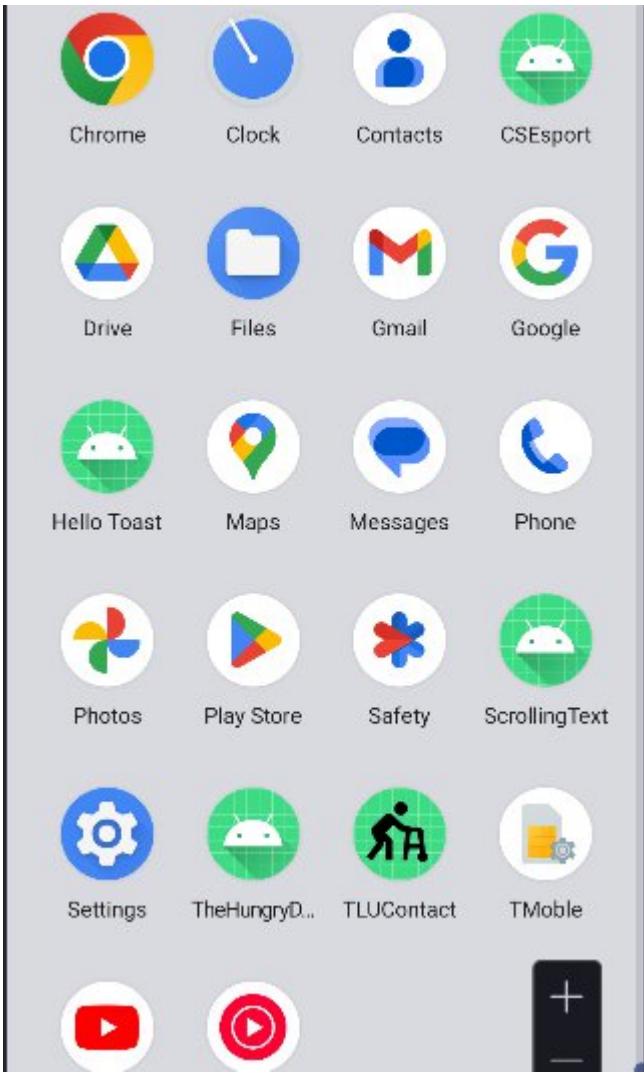


5. Nhập vào biểu tượng trong trường **Clip Art**. Các biểu tượng xuất hiện từ bộ biểu tượng thiết kế vật liệu.
6. Duyệt qua cửa sổ Chọn biểu tượng, chọn một biểu tượng thích hợp (chẳng hạn như biểu tượng tâm trạng để gợi ý tâm trạng tốt), sau đó nhập vào **OK**.



7. Nhấp vào tab **Lớp nền**, chọn **Màu làm Loại tài sản**, sau đó nhấp vào chip màu để chọn màu sử dụng làm lớp nền.
8. Nhấp vào tab **Legacy** và xem lại cài đặt mặc định. Xác nhận rằng bạn muốn tạo biểu tượng legacy, round và Google Play Store. Nhấp vào **Next** khi hoàn tất.
9. Chạy ứng dụng.

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



Nhiệm vụ 2: Sử dụng mẫu dự án

Android Studio cung cấp các mẫu cho các thiết kế ứng dụng và hoạt động phổ biến và được đề xuất. Sử dụng các mẫu tích hợp giúp tiết kiệm thời gian và giúp bạn tuân thủ các phương pháp thiết kế tốt nhất.

Mỗi mẫu kết hợp một hoạt động khung và giao diện người dùng. Bạn đã sử dụng mẫu Hoạt động trống. Mẫu Hoạt động cơ bản có nhiều tính năng hơn và kết hợp các tính năng ứng dụng được đề xuất, chẳng hạn như menu tùy chọn xuất hiện trên thanh ứng dụng.

2.1 Khám phá kiến trúc Hoạt động cơ bản

Mẫu Hoạt động cơ bản là mẫu đa năng do Android Studio cung cấp để hỗ trợ bạn bắt đầu phát triển ứng dụng.

1. Trong Android Studio, tạo một dự án mới với mẫu Hoạt động cơ bản.
2. Xây dựng và chạy ứng dụng.
3. Xác định các phần được gắn nhãn trong hình và bảng bên dưới. Tìm các phần tương đương của chúng trên màn hình thiết bị hoặc trình giả lập của bạn. Kiểm tra mã Java và tệp XML tương ứng được mô tả trong bảng.

Việc quen thuộc với mã nguồn Java và các tệp XML sẽ giúp bạn mở rộng và tùy chỉnh mẫu này theo nhu cầu của riêng bạn.

#	Mô tả giao diện người dùng	Mã tham chiếu
1	Thanh trạng thái Hệ thống Android cung cấp và kiểm soát thanh trạng thái.	Không hiển thị trong mã mẫu. Bạn có thể truy cập vào mục này thông qua hoạt động của mình. Ví dụ, bạn có thể ấn thanh trạng thái , nếu cần thiết.
2	AppBarLayout > Thanh công cụ Thanh ứng dụng (còn gọi là thanh hành động) cung cấp cấu trúc trực quan, hình ảnh chuẩn hóa các yếu tố và điều hướng. Đối với ngược	Trong activity_main.xml , hãy tìm android.hỗ trợ.v7.widget.Thanh công cụ
	mẫu nhúng	khả năng tương thích. AppBarLayout trong bên trong
	mẫu nhúng chức năng như một Thanh hành động .	android.hỗ trợ.thiết kế.widget.AppBarL ngoài kia . Thay đổi thanh công cụ để thay đổi

		xuất hiện của cha mẹ nó, thanh ứng dụng. Đối với ví dụ, hãy xem Hướng dẫn thanh ứng dụng .
3	Tên ứng dụng Điều này bắt nguồn từ tên gói của bạn, nhưng có thể là bất cứ thứ gì bạn chọn.	Trong AndroidManifest.xml : android:label="@string/tên_ứng_dụng"
4	Nút tràn menu tùy chọn Các mục menu cho hoạt động cũng như toàn cầu các tùy chọn, chẳng hạn như Tìm kiếm và Cài đặt cho ứng dụng. Các mục menu ứng dụng của bạn sẽ được đưa vào menu này.	Trong MainActivity.java : onOptionsItemSelected() thực hiện điều gì xảy ra khi một mục menu là đã chọn. res > menu > menu_main.xml Tài nguyên chỉ định các mục menu cho menu tùy chọn.
5	Bố trí ViewGroup Các CoordinatorLayout ViewGroup là một bối cảnh giàu tính năng cung cấp cơ chế để các thành phần View (UI) tương tác. Ứng dụng của bạn giao diện người dùng đi vào bên trong tệp content_main.xml được bao gồm trong này Nhóm xem .	Trong activity_main.xml : Không có chế độ xem nào được chỉ định trong bối cảnh này; thay vào đó, nó bao gồm một bối cảnh khác với bao gồm hướng dẫn bố trí để bao gồm @layout/content_main nơi có các chế độ xem được chỉ định. Điều này tách biệt chế độ xem hệ thống khỏi chế độ xem dành riêng cho ứng dụng của bạn.
6	Xem văn bản Trong ví dụ, được sử dụng để hiển thị "Hello World". Thay thế phần này bằng các thành phần UI cho bạn	Trong content_main.xml : Tất cả các thành phần UI của ứng dụng của bạn được xác định trong tập tin này.

	Ứng dụng.	
7	Nút hành động nổi (FAB)	Trong activity_main.xml như một phần tử UI sử dụng biểu tượng clip-art. MainActivity.java bao gồm một stub trong onCreate() thiết lập một onClick() cho FAB.

2.2 Tùy chỉnh ứng dụng được tạo ra bởi mẫu

Thay đổi giao diện của ứng dụng được tạo ra bởi mẫu Hoạt động cơ bản. Ví dụ, bạn có thể thay đổi màu của thanh ứng dụng để phù hợp với thanh trạng thái (trên một số thiết bị là màu tối hơn) cùng màu cơ bản). Bạn cũng có thể muốn xóa nút hành động nổi nếu bạn không định sử dụng nó.

1. Thay đổi màu của thanh ứng dụng (Thanh công cụ) trong activity_main.xml bằng cách thay đổi android:background thành "?attr/colorPrimaryDark" , điều này sẽ đặt màu của thanh ứng dụng thành màu chính tối hơn phù hợp với thanh trạng thái:
2. Để xóa nút hành động nổi, hãy bắt đầu bằng cách xóa mã stub trong onCreate() để thiết lập trình lắng nghe onClick() cho nút. Mở **MainActivity** và xóa khôi mã sau:
3. Để xóa nút hành động nổi khỏi bộ cục, hãy xóa khôi mã XML sau khỏi activity_main.xml :
4. Thay đổi tên ứng dụng được hiển thị trên thanh ứng dụng bằng cách thay đổi tài nguyên chuỗi app_name trong strings.xml thành tên sau:
5. Chạy ứng dụng. Nút hành động nổi không còn xuất hiện nữa, tên đã thay đổi và màu nền của thanh ứng dụng đã thay đổi.

2.3 Khám phá cách thêm hoạt động bằng cách sử dụng mẫu

Đối với các bài thực hành cho đến nay, bạn đã sử dụng các mẫu Empty Activity và Basic Activity. Trong các bài học sau, các mẫu bạn sử dụng sẽ khác nhau, tùy thuộc vào nhiệm vụ.

Các mẫu hoạt động này cũng có sẵn bên trong dự án của bạn để bạn có thể thêm nhiều hoạt động hơn vào ứng dụng sau khi thiết lập dự án ban đầu. (Bạn sẽ tìm hiểu thêm về lớp Activity trong một chương khác.)

1. Tạo một dự án ứng dụng mới hoặc chọn một dự án hiện có.
2. Trong ngăn **Project > Android** , nhấp chuột phải vào thư mục **java** .
3. Chọn **Mới > Hoạt động > Thư viện** .
4. Thêm Hoạt động . Ví dụ: nhấp vào **Hoạt động** kéo điều hướng để thêm Hoạt động có ngăn kéo điều hướng vào ứng dụng của bạn.
5. Nhấp đúp vào tệp bối cảnh của Hoạt động để hiển thị chúng trong trình chỉnh sửa bối cảnh.

Nhiệm vụ 3: Học từ mã ví dụ

Android Studio và tài liệu Android cung cấp nhiều mẫu mã mà bạn có thể nghiên cứu, sao chép và kết hợp vào các dự án của mình.

3.1 Mẫu mã Android

Bạn có thể khám phá hàng trăm mẫu mã trực tiếp từ Android Studio.

1. Trong Android Studio, chọn **File > New > Import Sample** .
2. Duyệt qua các mẫu.
3. Chọn một mẫu và nhấp vào **Tiếp theo** .
4. Chấp nhận các cài đặt mặc định và nhấp vào **Kết thúc** .

3.2 Sử dụng Trình quản lý SDK để cài đặt tài liệu ngoại tuyến

Cài đặt Android Studio cũng cài đặt các thành phần thiết yếu của Android SDK (Software Development Kit). Tuy nhiên, các thư viện và tài liệu bổ sung có sẵn và bạn có thể cài đặt chúng bằng Trình quản lý SDK.

1. Chọn **Công cụ > Android > Trình quản lý SDK** .

2. Ở cột bên trái, nhấp vào **Android SDK** .
3. Chọn và sao chép đường dẫn đến Vị trí SDK Android ở đầu màn hình vì bạn sẽ cần đường dẫn này để
 1. Nhấp vào tab **Nền tảng SDK** . Bạn có thể cài đặt các phiên bản bổ sung của hệ thống Android từ đây.
 2. Nhấp vào tab **SDK Update Sites** . Android Studio sẽ kiểm tra các trang web được liệt kê và chọn thường xuyên để cập nhật.
 3. Nhấp vào tab **Công cụ SDK** . Bạn có thể cài đặt thêm Công cụ SDK không được cài đặt theo mặc định, cũng như phiên bản ngoại tuyến của tài liệu dành cho nhà phát triển Android.
 4. Chọn hộp kiểm "Tài liệu cho Android SDK" nếu nó chưa được cài đặt và nhấp vào **Áp dụng** .
 5. Khi quá trình cài đặt hoàn tất, hãy nhấp vào **Kết thúc** .
 6. Điều hướng đến thư mục **sdk** mà bạn đã sao chép ở trên và mở thư mục **docs** .
 7. Tìm **index.html** và mở nó.

1.5) Tài nguyên có sẵn

Bài 2) Activities

2.1) Activity và Intent

Giới thiệu

MỘT Hoạt động đại diện cho một màn hình duy nhất trong ứng dụng của bạn mà người dùng có thể thực hiện một tác vụ duy nhất, tập trung như chụp ảnh, gửi email hoặc xem bản đồ. Hoạt động thường được trình bày cho người dùng dưới dạng cửa sổ toàn màn hình.

Một ứng dụng thường bao gồm nhiều màn hình được liên kết lồng lěo với nhau. Mỗi màn hình là một hoạt động. Thông thường, một hoạt động trong ứng dụng được chỉ định là hoạt động "chính" (`MainActivity.java`), được trình bày cho người dùng khi ứng dụng được khởi chạy. Hoạt động chính sau đó có thể bắt đầu các hoạt động khác để thực hiện các hành động khác nhau.

Mỗi lần một hoạt động mới bắt đầu, hoạt động trước đó sẽ dừng lại, nhưng hệ thống vẫn giữ nguyên hoạt động trong một ngăn xếp ("ngăn xếp ngược"). Khi một hoạt động mới bắt đầu, hoạt động mới đó sẽ được đẩy vào ngăn xếp ngược và lấy sự tập trung của người dùng. Ngăn xếp ngược tuân theo logic ngăn xếp cơ bản "vào sau, ra trước". Khi người dùng hoàn tất hoạt động hiện tại và nhấn nút Quay lại, hoạt động đó sẽ được bật ra khỏi ngăn xếp và hủy, và hoạt động trước đó sẽ tiếp tục.

Một hoạt động được bắt đầu hoặc kích hoạt với một *ý định*. Một [Intent](#) là một thông điệp không đồng bộ mà bạn có thể sử dụng trong hoạt động của mình để yêu cầu một hành động từ một hoạt động khác hoặc từ một số thành phần ứng dụng khác. Bạn sử dụng Intent để bắt đầu một hoạt động từ một hoạt động khác và để truyền dữ liệu giữa các hoạt động.

Một ý định có thể *rõ ràng* hoặc *ngầm định*:

- Một *ý định rõ ràng* là ý định mà bạn biết mục tiêu của ý định đó. Nghĩa là bạn đã biết tên lớp đủ điều kiện của hoạt động cụ thể đó.
- *Ý định ngầm* là ý định mà bạn không biết tên của thành phần mục tiêu nhưng có một hành động chung cần thực hiện.

Những điều bạn nên biết

Bạn sẽ có thể:

- Tạo và chạy ứng dụng trong Android Studio.
- Sử dụng trình chỉnh sửa bố cục để tạo bố cục trong ConstraintLayout
- Chỉnh sửa mã XML bố cục.

Những gì bạn sẽ học được

- Cách tạo một Activity mới trong Android Studio.
- Cách xác định hoạt động cha và hoạt động con cho điều hướng Lên.
- Cách bắt đầu một Hoạt động với một Ý định rõ ràng .
- Cách truyền dữ liệu giữa mỗi Hoạt động với một Ý định rõ ràng .

Bạn sẽ làm gì

- Tạo một ứng dụng Android mới với một Hoạt động chính và một Hoạt động thứ hai .
- Activity chính đến Activity thứ hai bằng cách sử dụng Intent và hiển thị dữ liệu đó trong Activity thứ hai .
- Gửi một bit dữ liệu khác thứ hai trở lại Activity chính , cũng sử dụng Intent .

• **Tổng quan về ứng dụng**

Trong chương này, bạn sẽ tạo và xây dựng một ứng dụng có tên là Two Activities, không có gì ngạc nhiên khi ứng dụng này chứa hai triển khai Activity . Bạn sẽ xây dựng ứng dụng theo ba giai đoạn.

Ở giai đoạn đầu tiên, bạn tạo một ứng dụng có hoạt động chính chứa một nút, **Gửi** . Khi người dùng nhấp vào nút này, hoạt động chính của bạn sẽ sử dụng ý định để bắt đầu hoạt động thứ hai.

Ở giai đoạn thứ hai, bạn thêm chế độ xem EditText vào hoạt động chính. Người dùng nhập tin nhắn và nhấp vào **Gửi** . Hoạt động chính sử dụng ý định để bắt đầu hoạt động thứ hai và gửi tin nhắn của người dùng đến hoạt động thứ hai. Hoạt động thứ hai hiển thị tin nhắn đã nhận được.

Ở giai đoạn cuối cùng của việc tạo ứng dụng Two Activities, bạn thêm EditText và nút **Reply** vào hoạt động thứ hai. Jetzt wenn der Benutzer drückt, kann er den Text in der EditText eingeben und auf den Button **Reply** klicken. Der Benutzer kann dann eine Nachricht in die EditText eingeben und auf den Button **Reply** klicken. Die Antwort wird dann imEditText angezeigt. Bei diesem Zeitpunkt kann der Benutzer die Antwort aus dem EditText kopieren und sie in die EditText des zweiten Aktivitäts übertragen. Dies ist ein einfaches Beispiel für die Verwendung von Intent im Android-System.

Nhiệm vụ 1: Tạo dự án TwoActivities

Trong tác vụ này, bạn thiết lập dự án ban đầu với một Hoạt động chính , xác định bố cục và xác định phương thức khung cho sự kiện onClick

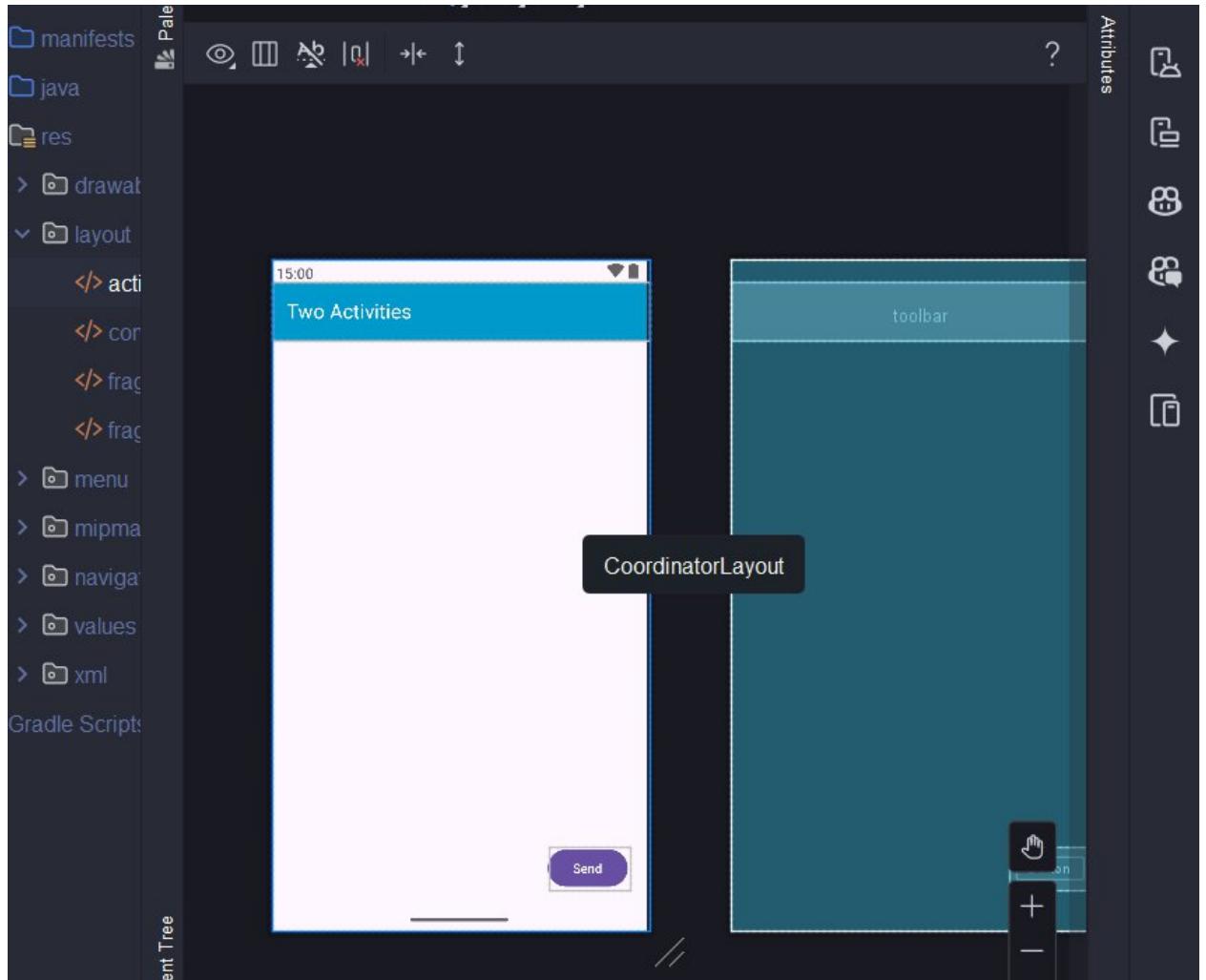
1.1 Tạo dự án TwoActivities

1. Khởi động Android Studio và tạo một dự án Android Studio mới.
Đặt tên cho ứng dụng của bạn là **Two Activities** và chọn cùng cài đặt **Điện thoại và Máy tính bảng** mà bạn đã sử dụng trong các bài thực hành trước. Thư mục dự án được tự động đặt tên là TwoActivities và tên ứng dụng xuất hiện trên thanh ứng dụng sẽ là "Two Activities".
2. Chọn **Hoạt động trống** cho mẫu Hoạt động . Nhấp vào **Tiếp theo**.
3. Chấp nhận tên Hoạt động mặc định (MainActivity). Đảm bảo rằng tùy chọn **Tạo tệp Bố cục** và **Tương thích ngược (AppCompat)** được chọn.

Nhấp vào **Kết thúc**

1.2 Xác định bố cục cho Hoạt động chính

1. Mở **res > layout > activity_main.xml** trong ngăn **Project > Android** .
Trình chỉnh sửa bố cục sẽ xuất hiện.
2. Nhấp vào tab **Thiết kế** nếu tab này chưa được chọn và xóa **TextView** (**TextView** có nội dung "Hello World") trong ngăn **Cây thành phần** .
3. Khi bật **Tự động kết nối** (cài đặt mặc định), hãy kéo Nút từ ngăn **Bảng màu** đến góc dưới bên phải của bố cục. **Tự động kết nối** tạo ra các ràng buộc cho Nút .
4. Trong ngăn **Thuộc tính** , đặt **ID** thành **button_main** , **layout_width** và **layout_height** thành **wrap_content** và nhập **Send** cho trường Text. Bây giờ, bố cục sẽ trông như thế này:

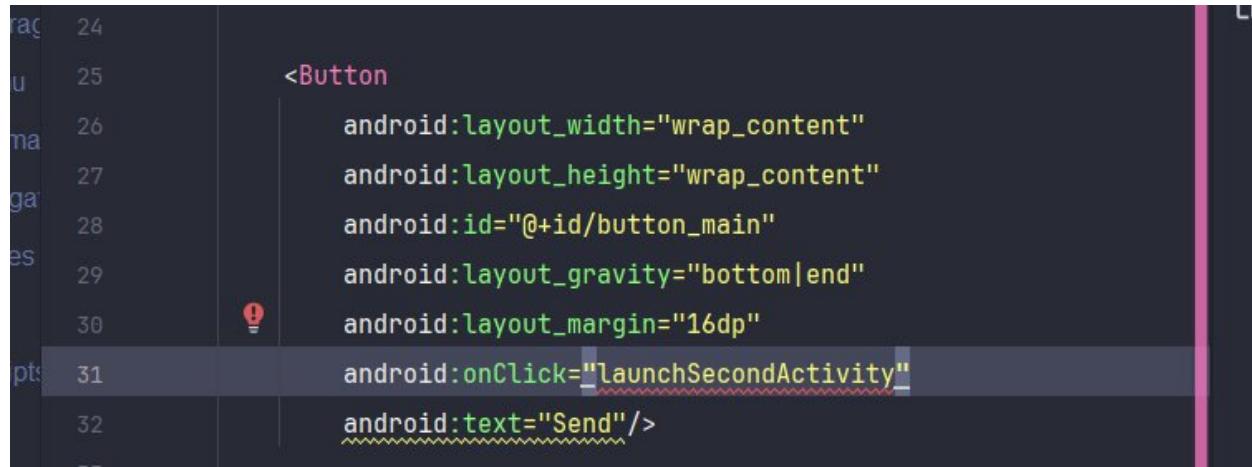


5. Nhập vào tab **Văn bản** để chỉnh sửa mã XML. Thêm thuộc tính sau vào Nút :

```
    android:onClick="launchHoạt động thứ hai"
```

Giá trị thuộc tính được gạch chân màu đỏ vì phương thức `launchSecondActivity()` vẫn chưa được tạo. Bỏ qua lỗi này ngay bây giờ; bạn sửa nó trong tác vụ tiếp theo.

6. Trích xuất tài nguyên chuỗi, như đã mô tả trong bài thực hành trước, cho "Gửi" và sử dụng tên `button_main` cho tài nguyên đó.



```
rac 24
u 25      <Button
ma 26          android:layout_width="wrap_content"
ga 27          android:layout_height="wrap_content"
es 28          android:id="@+id/button_main"
30          android:layout_gravity="bottom|end"
pts 31          android:layout_margin="16dp"
32          android:onClick="launchSecondActivity"
    
```

1.3 Xác định hành động của Nút

Trong tác vụ này, bạn triển khai phương thức launchSecondActivity() mà bạn đã tham chiếu trong bối cảnh cho thuộc tính android:onClick .

1. Nhập vào "**launchSecondActivity**" trong mã XML activity_main.xml .
2. Nhấn Alt+Enter (Option+Enter trên máy Mac) và chọn **Tạo 'launchSecondActivity(View)' trong 'MainActivity'**.

Tệp MainActivity mở ra và Android Studio tạo phương thức khung cho trình xử lý launchSecondActivity () .

3. Bên trong launchSecondActivity() , thêm câu lệnh Log có nội dung "Nút đã nhấp!"

Log.d(LOG_TAG, "Đã nhấp vào nút!");

LOG_TAG sẽ hiển thị màu đỏ. Bạn thêm định nghĩa cho biến đó ở bước sau.

4. Ở đầu lớp MainActivity , thêm hằng số cho biến LOG_TAG :
private static final String LOG_TAG = MainActivity.class.getSimpleName
5. Chạy ứng dụng của bạn. Khi bạn nhấp vào nút **Gửi** , bạn sẽ thấy thông báo "Nút đã nhấp!" trong ngăn **Logcat** . Nếu có quá nhiều điều ra trong màn hình, hãy nhập **MainActivity** vào tìm kiếm hộp và ngăn **Logcat** sẽ chỉ hiển thị các dòng khớp với thẻ đó.

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

protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Log.d(LOG_TAG, "Button clicked!");
}
```

Nhiệm vụ 2: Tạo và khởi chạy Hoạt động thứ hai

Mỗi hoạt động mới mà bạn thêm vào dự án của mình đều có bộ cục và tệp Java riêng, tách biệt với hoạt động chính. Chúng cũng có các phần tử <activity> riêng trong tệp AndroidManifest.xml . Giống như hoạt động chính, các triển khai hoạt động mới mà bạn tạo trong Android Studio cũng mở rộng từ lớp AppCompatActivity .

Mỗi hoạt động trong ứng dụng của bạn chỉ được kết nối lỏng lẻo với các hoạt động khác. Tuy nhiên, bạn có thể định nghĩa một hoạt động là cha của một hoạt động khác trong tệp AndroidManifest.xml . Mỗi quan hệ cha-con này cho phép Android thêm các gợi ý điều hướng như mũi tên hướng sang trái trên thanh tiêu đề cho mỗi hoạt động.

Một hoạt động giao tiếp với các hoạt động khác (trong cùng một ứng dụng và trên các ứng dụng khác nhau) bằng một ý định. Một ý định có thể là *rõ ràng* hoặc *ngầm định* :

- Một ý định rõ ràng là ý định mà bạn biết mục tiêu của ý định đó; nghĩa là bạn đã biết tên lớp đủ điều kiện của hoạt động cụ thể đó.
- Ý định ngầm là ý định mà bạn không biết tên của thành phần mục tiêu nhưng có hành động chung để thực hiện.

Trong tác vụ này, bạn thêm một hoạt động thứ hai vào ứng dụng của chúng tôi, với bộ cục riêng của nó. Bạn sửa đổi tệp AndroidManifest.xml để xác định hoạt động chính là hoạt động cha của hoạt động thứ hai. Sau đó, bạn sửa đổi phương thức launchSecondActivity() trong MainActivity để bao gồm một mục đích khởi chạy hoạt động thứ hai khi bạn nhấp vào nút.

2.1 Tạo Hoạt động thứ hai

1. Nhấp vào thư mục **Ứng dụng** cho dự án của bạn và chọn **Tệp > Mới > Hoạt động > Hoạt động trống**.
2. Hoạt động mới **SecondActivity**. Đảm bảo rằng **Generate Layout File** và **Backwards Compatibility (AppCompat)** được chọn. Tên layout được điền là `activity_second`. Không *chọn* tùy chọn **Hoạt động của trình khởi chạy**.
3. Nhấp vào **Hoàn tất**. Android Studio thêm cả bộ cục Activity mới (`activity_second.xml`) và tệp Java mới (`SecondActivity.java`) vào dự án của bạn cho Activity mới. Nó cũng cập nhật tệp `AndroidManifest.xml` để bao gồm Activity mới.

2.2 Sửa đổi tệp `AndroidManifest.xml`

1. Mở **manifest > AndroidManifest.xml**.
2. Tìm phần tử `<activity>` mà Android Studio đã tạo cho Activity thứ hai.
`<hoạt động android:name=".SecondActivity"></hoạt động>`
3. Thay thế toàn bộ phần tử `<activity>` bằng phần tử sau:

2.2) Vòng đời của Activity và trạng thái

2.3) Intent ngầm định

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

3.1) Trình gỡ lỗi

3.2) Kiểm thử đơn vị

3.3) Thư viện hỗ trợ

CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG

Bài 1) Tương tác người dùng

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

Bài 2) Trải nghiệm người dùng thú vị

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

Bài 3) Kiểm thử giao diện người dùng

- 3.1) Espresso cho việc kiểm tra UI**

CHƯƠNG 3. LÀM VIỆC TRONG NỀN

Bài 1) Các tác vụ nền

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

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

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

CHƯƠNG 4. LUU DỮ LIỆU NGƯỜI DÙNG

Bài 1) Tùy chọn và cài đặt

1.1) Shared preferences

1.2) Cài đặt ứng dụng

Bài 2) Lưu trữ dữ liệu với Room

2.1) Room, LiveData và ViewModel

2.2) Room, LiveData và ViewModel