

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.....	4
Bài 1) Tạo ứng dụng đầu tiên	4
1.1) Android Studio và Hello World	4
1.2) Giao diện người dùng tương tác đầu tiên	20
1.3) Trình chỉnh sửa bố cục.....	Error! Bookmark not defined.
1.4) Văn bản và các chế độ cuộn.....	60
1.5) Tài nguyên có sẵn	60
Bài 2) Activities	60
2.1) Activity và Intent	60
2.2) Vòng đời của Activity và trạng thái.....	60
2.3) Intent ngầm định	60
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ.....	60
3.1) Trình gỡ lỗi	60
3.2) Kiểm thử đơn vị.....	60
3.3) Thư viện hỗ trợ	60
CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG.....	61
Bài 1) Tương tác người dùng.....	61
1.1) Hình ảnh có thể chọn	61
1.2) Các điều khiển nhập liệu.....	61
1.3) Menu và bộ chọn.....	61
1.4) Điều hướng người dùng.....	61
1.5) RecyclerView	61
Bài 2) Trải nghiệm người dùng thú vị	61
2.1) Hình vẽ, định kiểu và chủ đề	61
2.2) Thẻ và màu sắc.....	61
2.3) Bố cục thích ứng.....	61
Bài 3) Kiểm thử giao diện người dùng	61

3.1) Espresso cho việc kiểm tra UI	61
CHƯƠNG 3. LÀM VIỆC TRONG NỀN.....	61
Bài 1) Các tác vụ nền	61
1.1) AsyncTask	61
1.2) AsyncTask và AsyncTaskLoader	61
1.3) Broadcast receivers	61
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền	61
2.1) Thông báo	61
2.2) Trình quản lý cảnh báo.....	61
2.3) JobScheduler	61
CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG.....	62
Bài 1) Tùy chọn và cài đặt	62
1.1) Shared preferences	62
1.2) Cài đặt ứng dụng	62
Bài 2) Lưu trữ dữ liệu với Room	62
2.1) Room, LiveData và ViewModel	62
2.2) Room, LiveData và ViewModel	62

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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

Những gì Bạn sẽ cần:

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

Những gì bạn sẽ học

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

Những gì bạn sẽ làm

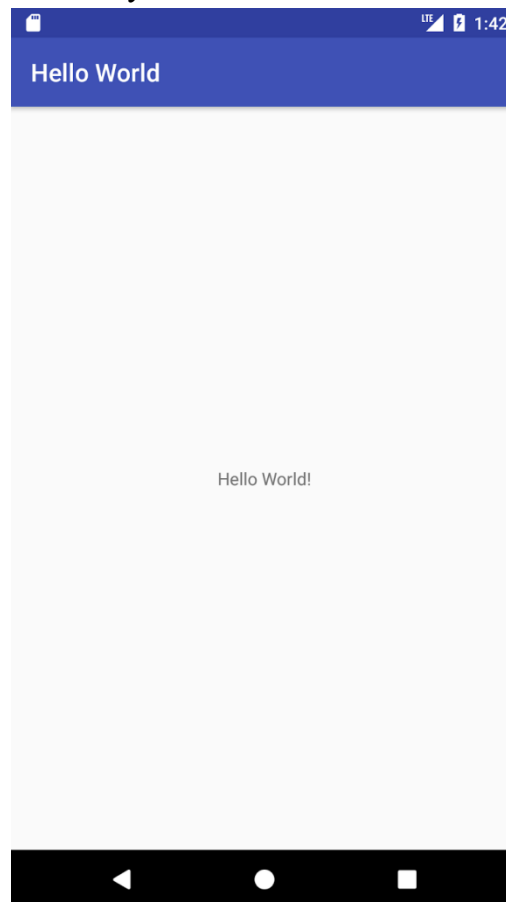
- Cài đặt môi trường phát triển **Android Studio**.
- Tạo một trình giả lập (thiết bị ảo) để chạy ứng dụng của bạn trên máy tính.
- Tạo và chạy ứng dụng **Hello World** trên các thiết bị ảo và vật lý.
- Khám phá cấu trúc dự án.

- Tạo và xem các thông điệp ghi lại từ ứng dụng của bạn.
- Khám phá tệp **AndroidManifest.xml**

Tổng quan ứng dụng

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

Ứng dụng hoàn thiện sẽ như thế này:



Nhiệm vụ 1: Cài đặt Android Studio

Android Studio cung cấp một môi trường phát triển tích hợp hoàn chỉnh (IDE) bao gồm một trình soạn thảo mã nâng cao và một tập các mẫu ứng dụng. Ngoài ra, nó chứa các công cụ cho phát triển, gỡ lỗi, kiểm thử, và hiệu suất giúp phát triển ứng dụng nhanh và hiệu quả hơn. Bạn cũng có thể thử nghiệm ứng dụng của bạn với một loạt các trình giả lập được cấu hình sẵn hoặc trên thiết bị di động của bạn, xây dựng ứng dụng sản xuất và xuất bản trên cửa hàng Google Play.

Android Studio có sẵn cho các máy tính chạy Windows hoặc Linux, và cho máy Macs chạy MacOS. OpenJDK (Java Development Kit) mới nhất được tích hợp sẵn trong Android Studio.

Để bắt đầu và chạy Android Studio, đầu tiên kiểm tra yêu cầu hệ thống để đảm bảo rằng hệ thống của bạn đáp ứng chúng. Việc cài đặt tương tự nhau trên mọi nền tảng. Bất kỳ sự khác biệt nào đã được ghi chú ở bên dưới.

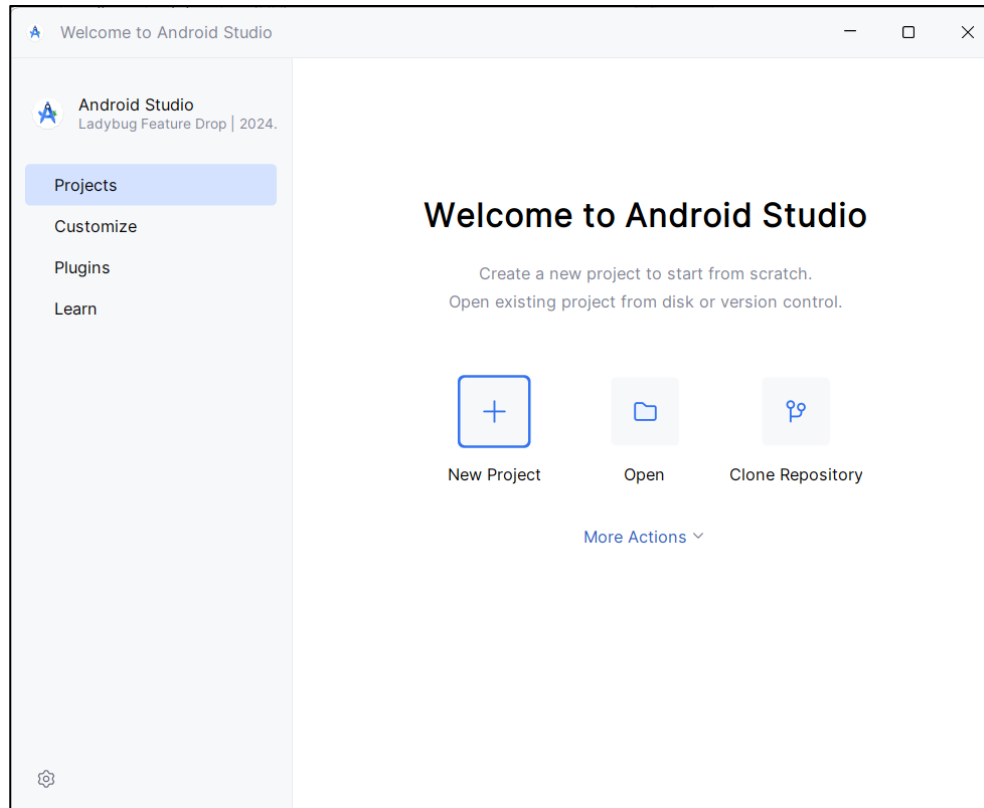
1. Truy cập trang web dành cho nhà phát triển Android và làm theo hướng dẫn để tải và cài đặt Android Studio.
2. Chấp nhận cấu hình mặc định cho tất cả các bước, và đảm bảo rằng các phần đã được chọn để cài đặt.
3. Sau khi kết thúc cài đặt, trình hướng dẫn cài đặt sẽ tải và cài đặt một số thành phần bổ sung bao gồm Android SDK. Hãy kiên nhẫn, việc này có thể mất thời gian tùy vào tốc độ internet của bạn, và một số bước có vẻ thừa thãi.
4. Khi hoàn tất tải xuống, Android Studio sẽ khởi động, và bạn sẵn sàng để tạo dự án đầu tiên của bạn.

Nhiệm vụ 2: Tạo ứng dụng Hello World

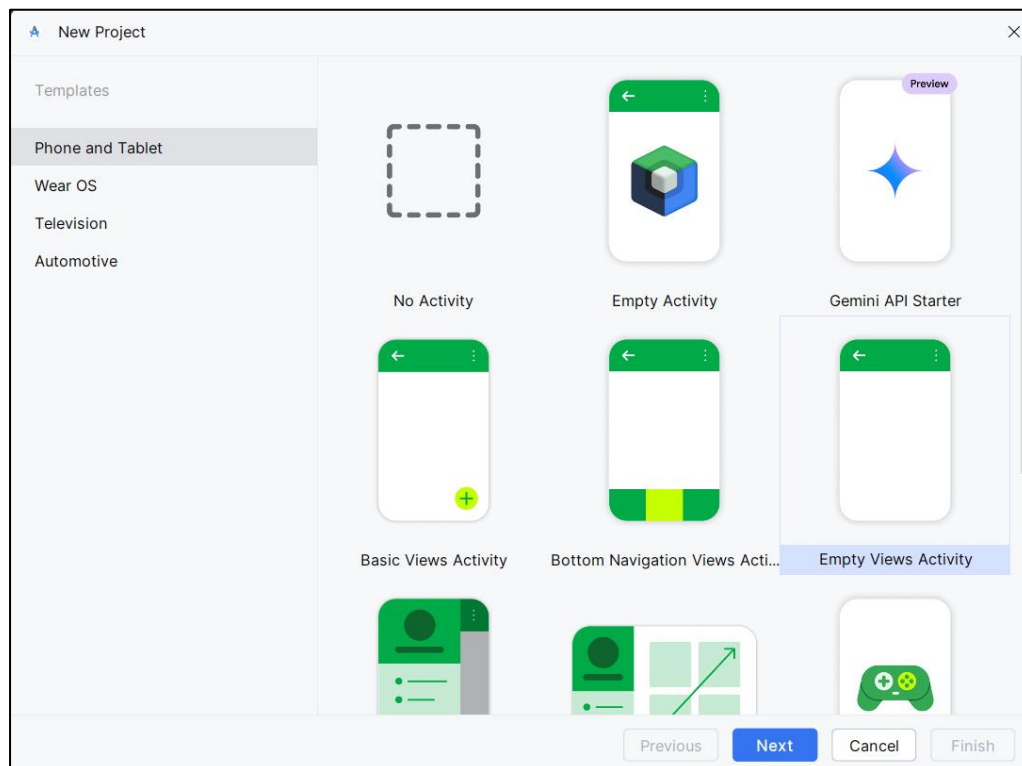
Trong nhiệm vụ này, bạn sẽ tạo một ứng dụng hiển thị “Hello World” để xác minh rằng Android Studio đã được cài đặt đúng, và để tìm hiểu cơ bản về phát triển với Android Studio.

Tạo dự án

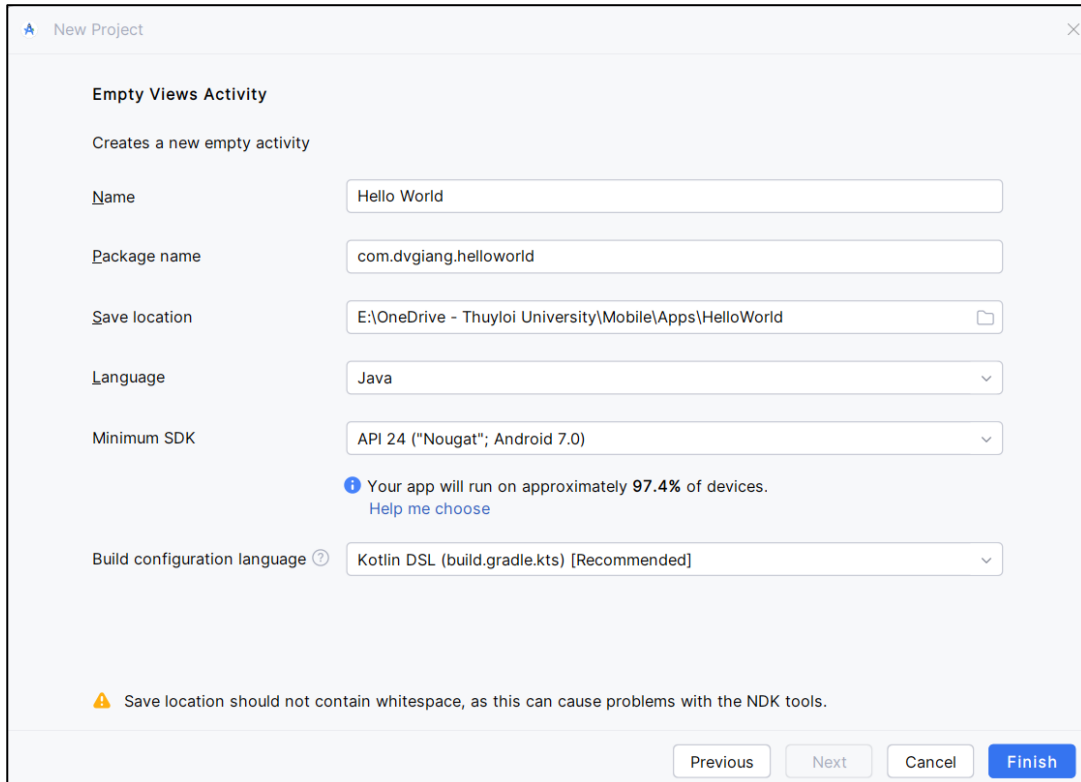
1. Mở Android Studio nếu nó chưa được mở.
2. Trong cửa sổ **Welcom to Android Studio**, chọn **New Project**.



3. Trong cửa sổ **New Project**, chọn **Phone and Tablet**, chọn **Empty Views Activity**. Tiếp theo nhấn **Next**.



4. Nhập các thông tin cho ứng dụng. Tên ứng dụng (**Name**) là **Hello World**, tên gói (**Package name**) là **com.dvgiang.helloworld**. Chọn vị trí lưu thư mục dự án (**Save location**), ngôn ngữ (**Language**) ở đây là **Java**, phiên bản hệ điều hành Android tối thiểu mà ứng dụng có thể chạy (**Minimum SDK**) ở đây là **Android 7.0**. Tiếp theo nhấn **Finish** để tạo dự án.



The screenshot shows the 'New Project' dialog in Android Studio. The dialog is titled 'New Project' and has a close button (X) in the top right corner. Below the title bar, there is a section titled 'Empty Views Activity' with the description 'Creates a new empty activity'. The dialog contains several input fields and dropdown menus:

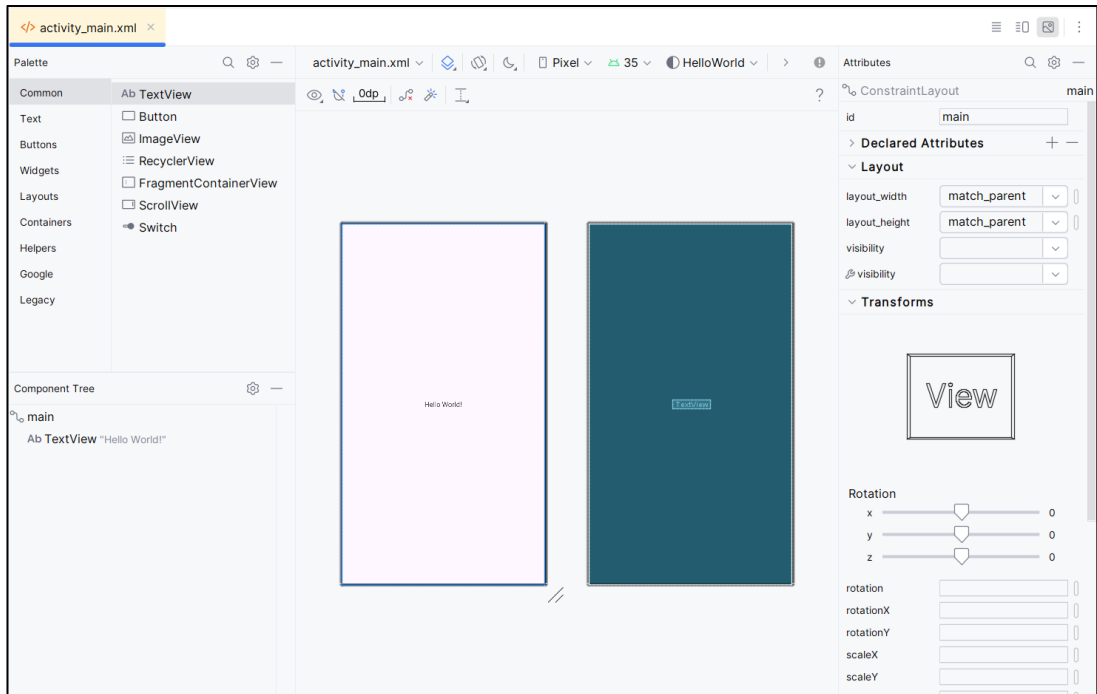
- Name:** A text field containing 'Hello World'.
- Package name:** A text field containing 'com.dvgiang.helloworld'.
- Save location:** A text field containing 'E:\OneDrive - Thuyloi University\Mobile\Apps\HelloWorld' with a folder icon on the right.
- Language:** A dropdown menu with 'Java' selected.
- Minimum SDK:** A dropdown menu with 'API 24 ("Nougat"; Android 7.0)' selected.
- Build configuration language:** A dropdown menu with 'Kotlin DSL (build.gradle.kts) [Recommended]' selected.

Below the 'Minimum SDK' dropdown, there is a blue information icon and the text: 'Your app will run on approximately 97.4% of devices. [Help me choose](#)'. At the bottom of the dialog, there is a yellow warning icon and the text: 'Save location should not contain whitespace, as this can cause problems with the NDK tools.' At the very bottom, there are four buttons: 'Previous', 'Next', 'Cancel', and 'Finish' (which is highlighted in blue).

Android Studio sẽ tạo một thư mục chứa dự án của bạn, và xây dựng dự án với Gradle.

Trình chỉnh sửa Android Studio xuất hiện, làm theo các bước sau:

1. Nhấn vào tab **activity_main.xml** để xem trình chỉnh sửa bố cục.



2. Nhấn tab **MainActivity.java** để xem trình chỉnh sửa mã.

```

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

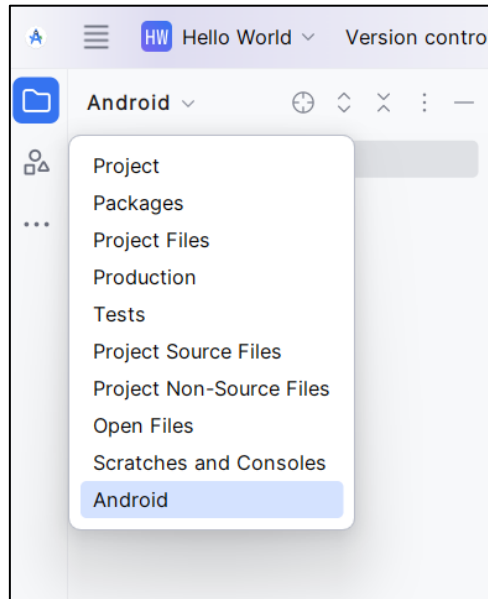
```

Khám phá dự án > Khung Android

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

1. Nếu chưa được chọn, nhấn tab **Project** trong cột tab dọc ở phía trên bên trái của cửa sổ Android Studio. Khung Project sẽ xuất hiện.

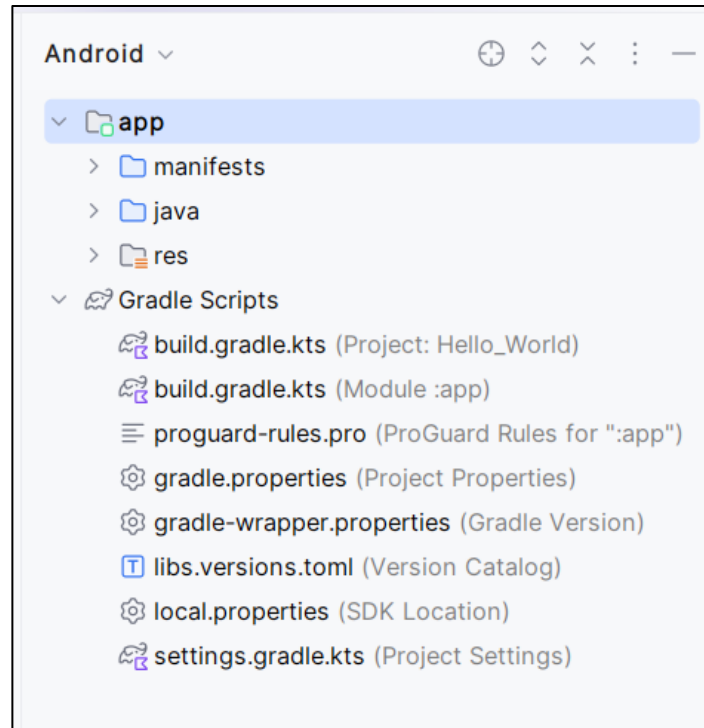
2. Để xem dự án trong hệ thống phân cấp dự án Android chuẩn, chọn Android từ menu hiện lên.



Khám phá thư mục Gradle Script

Hệ thống Gradle trong Android Studio giúp dễ dàng đưa các tệp nhị phân bên ngoài hoặc các mô-đun thư viện khác vào bản dựng của bạn như các phần phụ thuộc.

Khi bạn tạo dự án đầu tiên, **Project > Android** sẽ xuất hiện với thư mục **Gradle Scripts** như ảnh bên dưới.



Làm theo các bước sau để khám phá hệ thống Gradle:

1. Nếu thư mục **Gradle Scripts** không được mở rộng, nhấn vào hình tam giác để mở nó. Thư mục này chứa tất cả các tệp cần thiết cho hệ thống xây dựng.

2. Tìm đến tệp **build.gradle.kts(Project: Hello_World)**

Đây là nơi bạn sẽ tìm thấy các tùy chọn cấu hình chung cho tất cả các mô-đun tạo nên dự án của bạn. Mỗi dự án Android Studio đều chứa một tệp Gradle build cấp cao nhất. Hầu hết, bạn sẽ không cần thay đổi bất cứ gì trên tệp này, nhưng vẫn hữu ích khi hiểu nội dung của nó.

Mặc định, tệp xây dựng cấp cao nhất sử dụng khối **buildscript** để xác định kho lưu trữ Gradle và các phụ thuộc chung cho tất cả các mô-đun trong dự án. Khi phụ thuộc của bạn khác với thư viện cục bộ hoặc cây tập tin, Gradle tìm kiếm các tệp trong bất kỳ kho lưu trữ trực tuyến nào được chỉ định trong khối kho lưu trữ của tệp này.

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

Ngoài tệp **build.gradle.kts** mức dự án, mỗi mô-đun có một tệp **build.gradle.kts** của nó, cho phép bạn cài đặt cấu hình bản dựng cho mỗi mô-đun cụ thể (ứng dụng Hello World chỉ có một mô-đun). Cấu hình các cài đặt bản dựng này cho phép bạn cung cấp các tùy chọn đóng gói tùy chỉnh, như các loại xây dựng bổ sung. Bạn cũng có thể ghi đè các cài đặt trong tệp **AndroidManifest.xml** hoặc **build.gradle.kts** cấp cao nhất.

Tập này thường phổ biến để chỉnh sửa khi thay đổi các cấu hình mức ứng dụng, như khai báo các phụ thuộc trong phần **dependencies**. Bạn cũng có thể khai báo thư viện phụ thuộc sử dụng một trong nhiều cấu hình phụ thuộc khác nhau. Mỗi cấu hình cung cấp cho Gradle các chỉ dẫn khác nhau cách sử dụng thư viện.

4. Nhấn vào dấu tam giác để đóng **Gradle Scripts**

Khám phá thư mục app và res

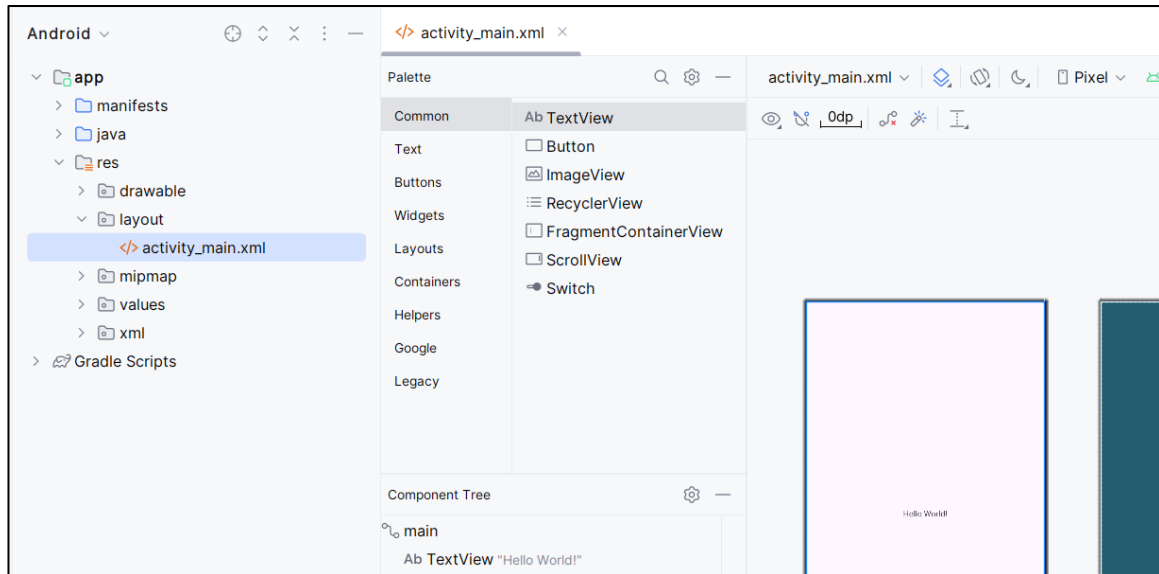
Tất cả mã và tài nguyên cho ứng dụng nằm trong thư mục **app** và **res**.

1. Mở rộng thư mục app, java, và com.dvgiang.helloworld để thấy tệp java MainActivity, nhấn đúp chuột để mở tệp trong trình chỉnh sửa mã.



Thư mục java bao gồm các tệp lớp Java trong ba thư mục con, như thể hiện trong hình trên. Thư mục **com.dvgiang.helloworld** (hoặc tên bạn đã chỉ định) chứa tất cả các tệp cho một gói ứng dụng. Hai thư mục còn lại được sử dụng để thử nghiệm. Đối với ứng dụng Hello World, chỉ có một gói chứa **MainActivity.java**. Tên của **Activity(screen)** đầu tiên mà người dùng nhìn thấy, cũng khởi tạo các tài nguyên trên toàn ứng dụng, thường được gọi là **MainActivity**.

2. Mở rộng thư mục **res**, **layout**, nhấn đúp chuột vào tệp **activity_main.xml** để mở trong trình chỉnh sửa bố cục.



Thư mục **res** chứa các tài nguyên, chẳng hạn như bố cục, chuỗi và hình ảnh. Một Activity thường được liên kết với bố cục của chế độ xem UI được định nghĩa là tệp XML. Tệp này thường được đặt tên theo Activity của nó.

Khám phá thư mục manifests

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

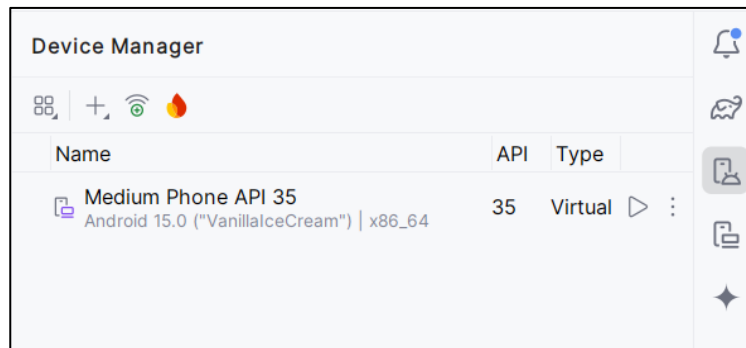
1. Mở rộng thư mục **manifests**.
2. Mở tệp **AndroidManifest.xml**.

Tệp **AndroidManifest.xml** mô tả tất cả các thành phần của ứng dụng Android của bạn. Tất cả các thành phần cho một ứng dụng, như mỗi Activity, phải được khai báo trong tệp XML này.

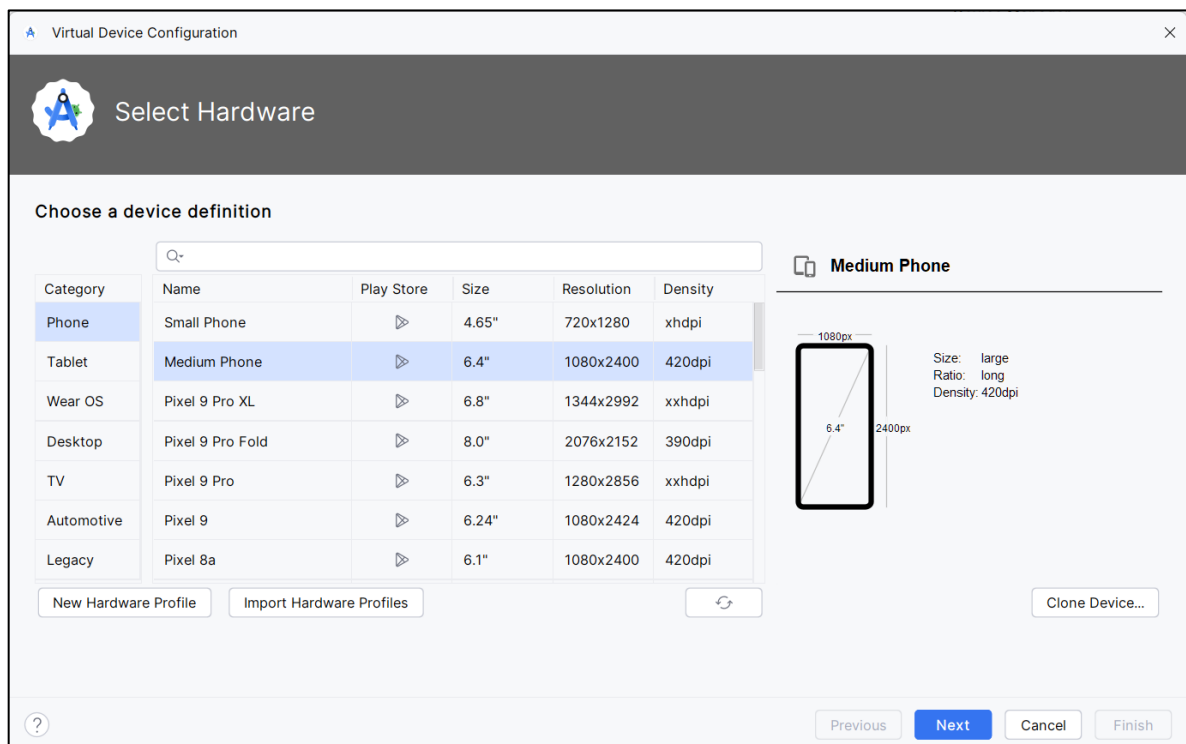
Nhiệm vụ 3: Tạo một thiết bị Android ảo (AVD)

Để chạy trình giả lập trên máy tính của bạn, bạn phải tạo một cấu hình mô tả thiết bị ảo.

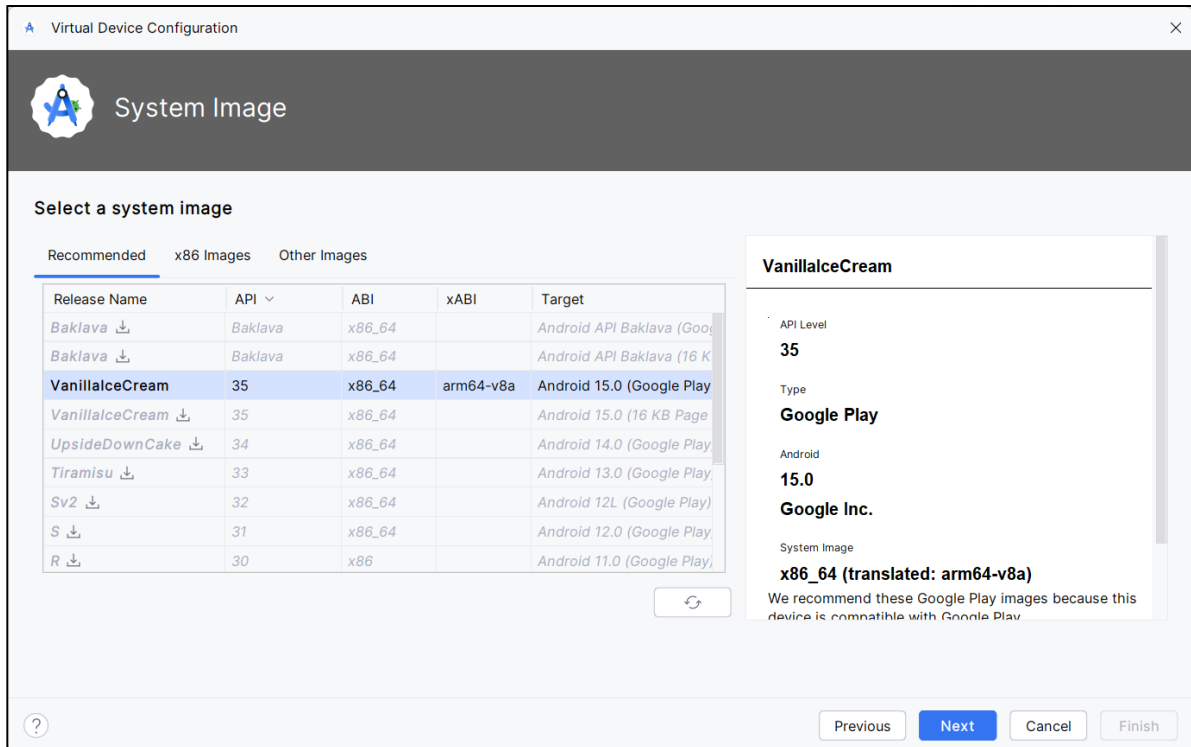
1. Trong Android Studio, chọn **Tools > Device Manager**, hoặc chọn vào biểu tượng Device Manager (📱) trên thanh công cụ. Màn hình Device Manager xuất hiện, nếu bạn đã tạo thiết bị ảo, màn hình sẽ hiển thị chúng, ngược lại, bạn thấy một danh sách trống.



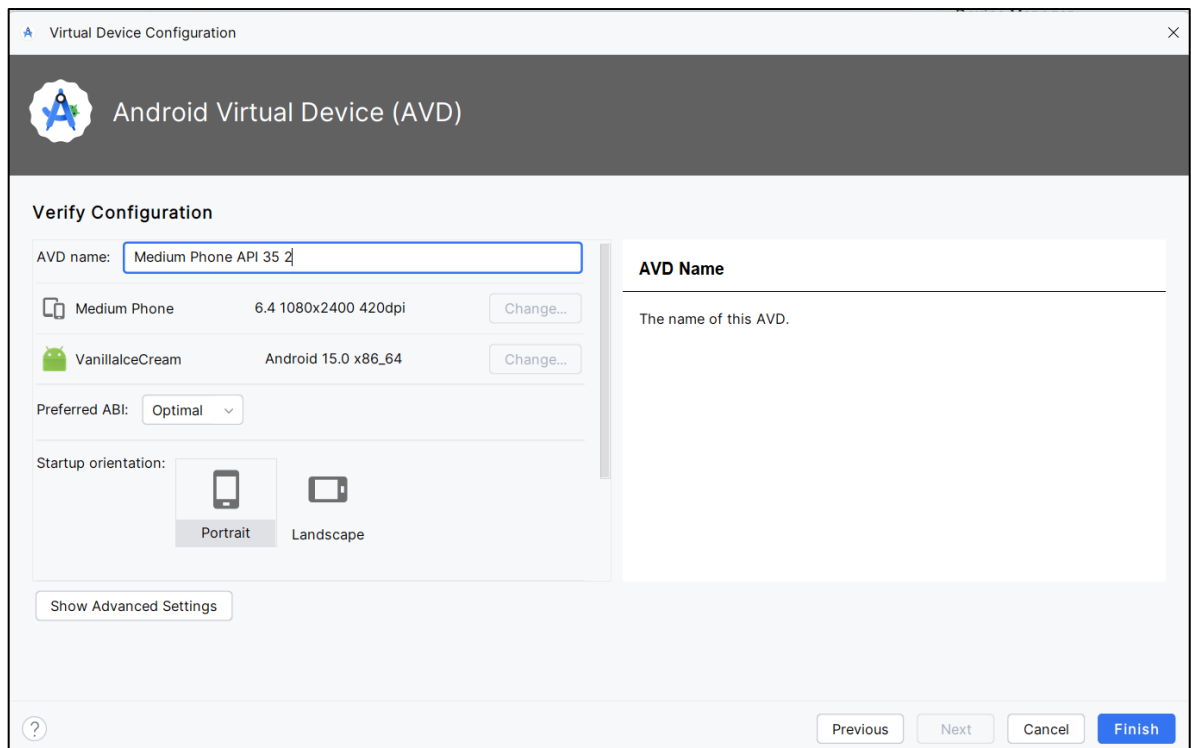
2. Nhấp vào biểu tượng dấu cộng (+) và chọn Create Virtual Device. Cửa sổ **Virtual Device Configuration** xuất hiện, tại đây bạn chọn loại thiết bị (Category) và thiết bị tương ứng. Sau đó nhấn **Next**.




3. Chọn phiên bản hệ điều hành tại cửa sổ mới, nhấn **Next**.



4. Tại màn hình mới, nhập tên thiết bị và xác nhận lại thông tin. Nhấn **Finish**.



Chạy ứng dụng trên máy ảo

Trong nhiệm vụ này, bạn sẽ chạy ứng dụng Hello World của bạn. Trong Android Studio, chọn thiết bị và nhấn biểu tượng Run (). Đợi cho ứng dụng chạy thành công, màn hình Hello World sẽ hiện ra.

Nhiệm vụ 4: Sử dụng thiết bị vật lý

Trong nhiệm vụ này, bạn sẽ chạy ứng dụng của bạn trên một thiết bị di động vật lý như điện thoại hay tablet. Bạn nên kiểm tra ứng dụng của bạn trên cả máy ảo và máy vật lý.

Những gì bạn cần:

- Một thiết bị Android như điện thoại hoặc tablet.
- Dây cáp để kết nối thiết bị Android với máy tính của bạn thông qua cổng USB.
- Nếu bạn đang sử dụng hệ thống Linux hoặc Windows, bạn có thể cần thực hiện các bước bổ sung để chạy trên thiết bị phần cứng.

Bật trình gỡ lỗi trên USB

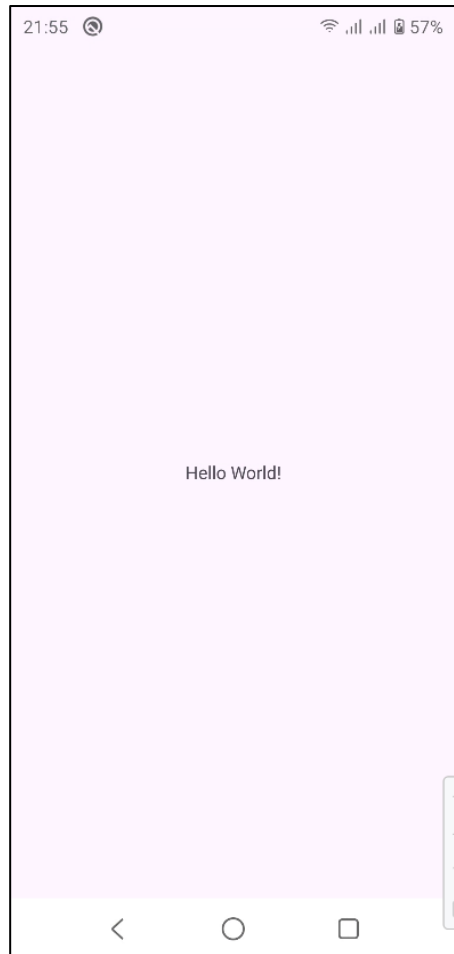
Để Android Studio giao tiếp được với thiết bị của bạn, bạn phải bật **Trình gỡ lỗi USB** trên thiết bị Android của bạn. Điều này có thể trong cài đặt **Tùy chọn nhà phát triển** của thiết bị của bạn.

Trên phiên bản Android 10.0 hoặc cao hơn, màn hình **Tùy chọn nhà phát triển** bị ẩn theo mặc định. Để hiển thị tùy chọn nhà phát triển và bật **Trình gỡ lỗi USB**:

1. Trên thiết bị của bạn, mở **Cài đặt**, tìm và nhấn vào phần **Giới thiệu thiết bị**, nhấn liên tiếp bảy lần vào **Số bản dựng**.
2. Trở về màn hình trước đó, **Tùy chọn nhà phát triển** xuất hiện. Chọn **Tùy chọn nhà phát triển**.
3. Bật chế độ **Trình gỡ lỗi USB**.

Chạy ứng dụng của bạn trên thiết bị

Bây giờ bạn có thể kết nối thiết bị của bạn và chạy ứng dụng từ Android Studio. Android Studio sẽ cài đặt và chạy ứng dụng trên thiết bị của bạn.



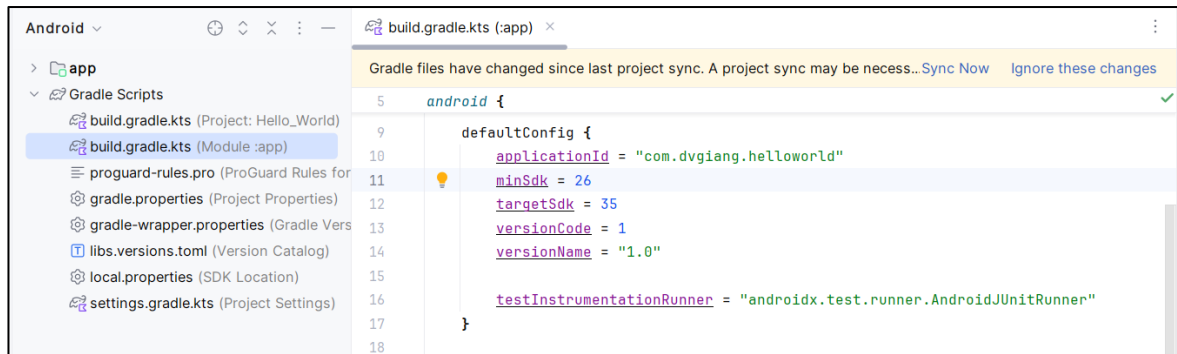
Nhiệm vụ 5: Thay đổi cấu hình Gradle của ứng dụng

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

Thay đổi phiên bản SDK tối thiểu cho ứng dụng

Thực hiện theo các bước sau:

1. Mở rộng thư mục **Gradle Scripts** nếu nó chưa mở, nhấn đúp vào tệp **build.gradle.kts (Module :app)**.
2. Trong khối **defaultConfig**, thay giá trị của **minSdk** thành 26 (hoặc phiên bản khác). Lúc này, trình sửa mã hiển thị thông báo ở trên cùng với hành động **Sync Now**.



Đồng bộ cấu hình Gradle mới

Khi bạn thực hiện thay đổi đối với các tệp cấu hình bản dựng trong một dự án, Android Studio yêu cầu bạn phải đồng bộ các tệp dự án để có thể nhập các thay đổi cấu hình bản dựng và chạy một số kiểm tra để đảm bảo cấu hình sẽ không tạo ra lỗi bản dựng.

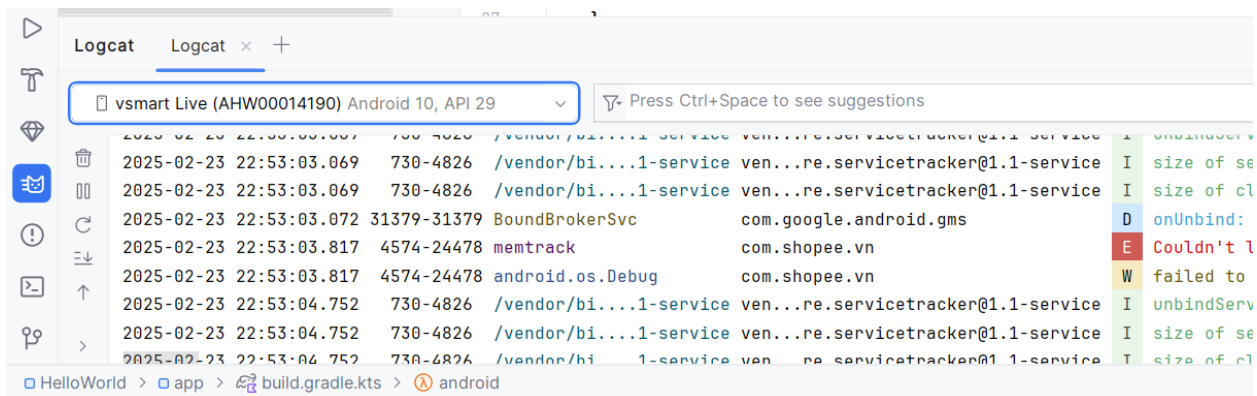
Để đồng bộ, nhấn **Sync Now** trên thanh thông báo xuất hiện khi thay đổi.

Nhiệm vụ 6: Thêm các câu lệnh log trong ứng dụng của bạn

Trong nhiệm vụ này, bạn sẽ thêm câu lệnh Log vào ứng dụng của mình, hiển thị các thông báo trong khung Logcat. Thông báo Log là một công cụ gỡ lỗi mạnh mẽ mà bạn có thể sử dụng để kiểm tra các giá trị, đường dẫn thực thi và báo cáo các ngoại lệ..

Xem khung Logcat

Để xem khung Logcat, nhấp vào tab **Logcat** ở cuối cửa sổ Android Studio.



Thêm các câu lệnh log cho ứng dụng của bạn

Mọi câu lệnh log trong mã ứng dụng của bạn sẽ hiển thị trong khung Logcat. Ví dụ:

```
Log.d( tag: "MainActivity", msg: "Hello world!");
```

Trong đó:

- Log: Lớp Log để gửi thông điệp log đến khung Logcat
- d: Cài đặt log ở mức **Debug** để lọc thông điệp log hiển thị trên khung Logcat. Các mức log khác là **e** cho mức **Error**, **w** cho mức **Warn**, và **i** cho **Info**.
- “MainActivity”: Đối số đầu tiên là một thẻ có thể được sử dụng để lọc các thông điệp trong khung Logcat. Đây thường là tên của Activity mà thông điệp bắt nguồn. Tuy nhiên, bạn có thể làm bất cứ điều gì hữu ích cho bạn để gỡ lỗi.

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

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

- “Hello world!”: Đối số thứ hai là thông điệp thực tế.

Làm theo các bước sau:

1. Mở ứng dụng Hello World trên Android studio, và mở MainActivity.
2. Trong phương thức onCreate() của lớp MainActivity, thêm câu lệnh sau đây:
“Log.d(“MainActivity”, “Hello world!”);”
3. Mở Logcat nếu chưa mở.

4. Chạy ứng dụng và xem log trong Logcat.

2025-02-24 15:19:33.587 13251-13251 MainActivity

com.dvgiang.helloworld

D Hello world!

Tóm tắt

- Để cài đặt **Android Studio**, đến trang chủ và làm theo chỉ dẫn để tải và cài đặt nó.
- Khi tạo ứng dụng mới, chắc chắn rằng **API 24: Android 7.0** được đặt là phiên bản SDK tối thiểu.
- Để xem phân cấp Android của ứng dụng trong khung **Project**, hãy nhấn vào tab **Project** trong cột tab dọc, sau đó chọn Android trong menu bật lên ở trên cùng.
- Chỉnh sửa tệp `build.gradle.kts`(Module :app) khi bạn cần thêm thư viện mới vào dự án hoặc thay đổi phiên bản thư viện.
- Tất cả mã và tài nguyên cho ứng dụng đều nằm trong thư mục **app** và **res**. Thư mục **java** bao gồm các activity, test và các thành phần khác trong mã nguồn Java. Thư mục **res** chứa các tài nguyên, chẳng hạn như bố cục, chuỗi và hình ảnh.
- Chỉnh sửa tệp **AndroidManifest.xml** để thêm các thành phần chức năng và quyền hạn cho ứng dụng Android của bạn. Tất cả các thành phần cho ứng dụng, như nhiều activity, phải được định nghĩa trong tệp XML này.
- Sử dụng **Android Virtual Device (AVD) manager** để tạo máy ảo để chạy ứng dụng của bạn.
- Thêm câu lệnh Log vào ứng dụng của bạn, hiển thị thông báo trong khung Logcat như một công cụ gỡ lỗi cơ bản.
- Để chạy ứng dụng trên một thiết bị Android sử dụng Android Studio, bật Trình gỡ lỗi USB trên thiết bị. Mở **Cài đặt > Giới thiệu điện thoại** > nhấn bảy lần vào **Số bản dựng**. Quay lại màn hình trước đó (**Cài đặt**), và nhấn **Tùy chọn nhà phát triển**. Chọn **Trình gỡ lỗi USB**.

1.2) Giao diện người dùng tương tác đầu tiên

Giới thiệu

Giao diện người dùng (UI) xuất hiện trên màn hình thiết bị Android bao gồm một hệ thống phân cấp các đối tượng được gọi là **views** - mọi thành phần của màn hình đều là một **View**. Lớp **View** đại diện cho khối xây dựng cơ bản cho tất cả các thành phần UI và là lớp cơ sở

cho các lớp cung cấp các thành phần UI tương tác như nút, hộp kiểm và trường nhập văn bản. Các lớp con của View thường được sử dụng được mô tả trong nhiều bài học bao gồm:

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

Mã Java hiển thị và điều khiển UI được chứa trong một lớp Activity mở rộng. Một Activity thường có liên quan với một bố cục UI được định nghĩa như một tệp XML. Tệp XML này thường được đặt tên theo Activity của nó và định nghĩa bố cục của các thành phần View trên màn hình.

Ví dụ, mã của MainActivity trong ứng dụng Hello World hiển thị bố cục được định nghĩa trong tệp bố cục activity_main.xml, bao gồm một TextView với văn bản “Hello World”.

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

Trong phần thực hành này, bạn sẽ tìm hiểu cách tạo ứng dụng tương tác đầu tiên - một ứng dụng người dùng có thể tương tác. Bạn tạo một ứng dụng sử dụng mẫu Empty Views Activity. Bạn cũng học cách sử dụng trình chỉnh sửa bố cục để thiết kế bố cục, và cách chỉnh sửa bố cục trong XML. Bạn cần phát triển các kỹ năng mà bạn có thể hoàn thành các bài thực hành khác trong khóa học này.

Những gì bạn nên biết

Bạn nên có khả năng:

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

Những gì bạn sẽ học

- Cách tạo ứng dụng với hành vi tương tác.
- Cách sử dụng trình chỉnh sửa bố cục để thiết kế bố cục.
- Cách chỉnh sửa bố cục trong XML.
- Nhiều thuật ngữ mới.

Những gì bạn sẽ làm

- Tạo một ứng dụng và thêm hai phần tử Button và một TextView vào bố cục.
- Thao tác từng phần tử trong ConstraintLayout để giới hạn chúng vào lề và các phần tử khác.
- Thay đổi các thuộc tính phần tử UI.
- Chỉnh sửa bố cục của ứng dụng trong XML.
- Trích xuất các chuỗi được mã hóa cứng thành các tài nguyên chuỗi.
- Triển khai các phương thức xử lý nhấp để hiển thị thông báo trên màn hình khi người dùng chạm vào mỗi Button.

Tổng quan

Ứng dụng HelloToast bao gồm hai thành phần Button và một TextView. Khi người dùng chạm vào Button đầu tiên, nó sẽ hiển thị một thông báo ngắn (Toast) trên màn hình. Chạm vào Button thứ hai sẽ tăng bộ đếm "nhấp" được hiển thị trong TextView, bắt đầu từ số 0.

Nhiệm vụ 1: Tạo và khám phá dự án mới

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

Tạo dự án Android Studio

Mở Android Studio và tạo một dự án mới với mẫu **Empty Views Activity** và các tham số như sau:

Attribute	Value
-----------	-------

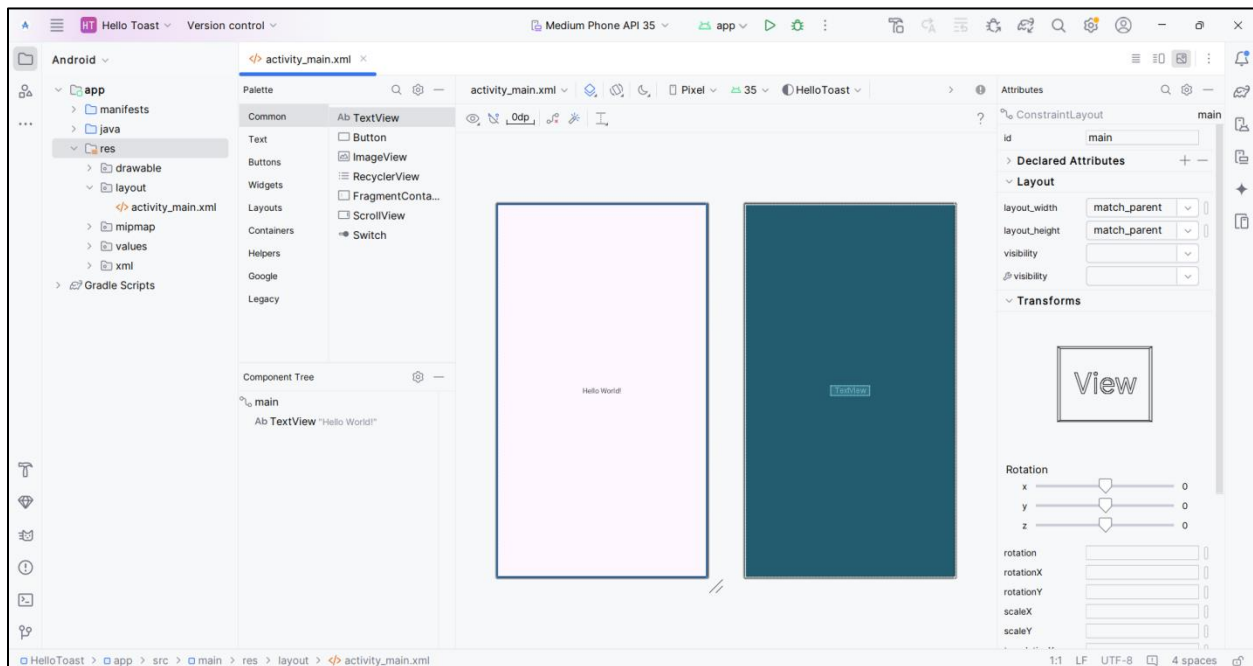
Name	Hello Toast
Package name	com.dvgiang.hellotoast (<i>Hoặc tên khác mà bạn đặt</i>)
Save location	<Nơi bạn lưu trữ dự án>
Language	Java
Minimum SDK	API 24 (“Nougat”; Android 7.0)
Build configuration language	Kotlin DSL (build.gradle.kts)

Sau đó chạy ứng dụng trên máy ảo hoặc một thiết bị Android.

Khám phá trình chỉnh sửa

Android Studio cung cấp trình chỉnh sửa bố cục để nhanh chóng xây dựng bố cục giao diện người dùng (UI) của ứng dụng. Nó cho phép bạn kéo các thành phần vào chế độ xem thiết kế trực quan và bản thiết kế, định vị chúng trong bố cục, thêm ràng buộc và đặt thuộc tính. Ràng buộc xác định vị trí của thành phần UI trong bố cục. Ràng buộc thể hiện kết nối hoặc căn chỉnh với chế độ xem khác, bố cục cha hoặc hướng dẫn vô hình.


Khám phá trình chỉnh sửa bố cục và tham khảo hình bên dưới:



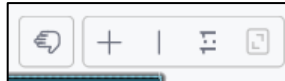
Kiểm tra các ràng buộc phần tử

Làm theo các bước sau:

1. Mở **activity_main.xml** từ khung **Project > Android** nếu chưa được mở. Nếu tab **Design** không mở, nhấp vào nó.

Nếu không có **bluesprint**, nhấn vào biểu tượng  ở trong thanh công cụ và chọn **Design + Blueprint**.

2. Nhấp vào nút phóng to để phóng to các ô thiết kế và bản thiết kế để xem cận cảnh.



3. Tham khảo hình ảnh động bên dưới để biết bước này. Nhấp vào tay cầm hình tròn ở bên phải của **TextView** để xóa ràng buộc theo chiều ngang liên kết chế độ xem với bên phải của bố cục. **TextView** nhảy sang bên trái vì nó không còn bị ràng buộc ở bên phải nữa. Để thêm lại ràng buộc theo chiều ngang, hãy nhấp vào cùng tay cầm đó và kéo một đường sang bên phải của bố cục.



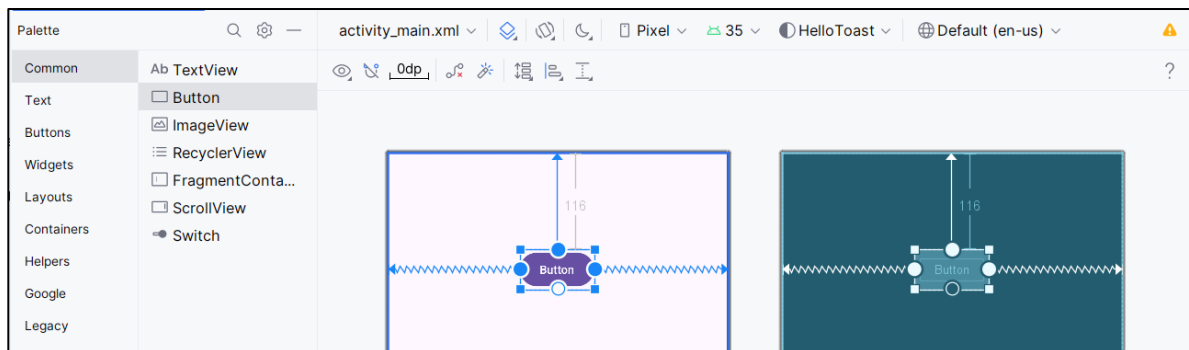
Thêm một Button vào bố cục

Làm theo các bước sau để thêm một Button:

1. Bắt đầu với một bảng trắng. Phần tử **TextView** không cần thiết, vì vậy khi nó vẫn được chọn, hãy nhấn phím **Delete** hoặc chọn **Edit > Delete**. Bây giờ bạn có một bố cục hoàn toàn trống.

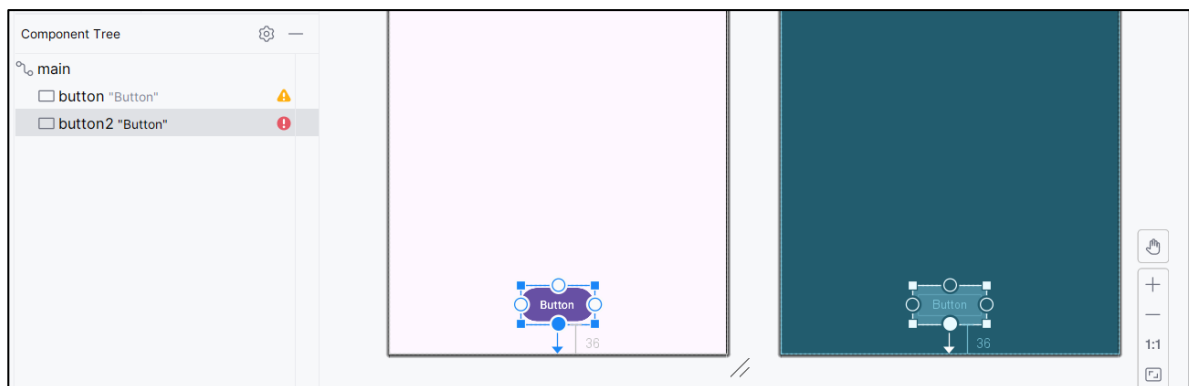
2. Kéo một **Button** từ ngăn **Palette** đến bất kỳ vị trí nào trong bố cục. Nếu bạn thả **Button** vào khu vực giữa trên cùng của bố cục, các ràng buộc có thể tự động xuất hiện.

Nếu không, bạn có thể kéo các ràng buộc lên trên cùng, bên trái và bên phải của bố cục.



Thêm Button thứ hai vào bố cục

1. Kéo một **Button** khác từ ngăn **Palette** vào giữa bố cục.
2. Kéo một ràng buộc theo chiều dọc xuống dưới cùng của bố cục.



Bạn có thể xóa ràng buộc khỏi một phần tử bằng cách chọn phần tử đó và nhấn chuột phải để hiển thị nút xóa ràng buộc (**Clear Constraints of Selection**). Nhấp vào nút này để xóa tất cả các ràng buộc trên phần tử đã chọn. Để xóa một ràng buộc duy nhất, hãy nhấp vào tay cầm cụ thể để đặt ràng buộc đó.

Nhiệm vụ 3: Thay đổi thuộc tính phần tử UI

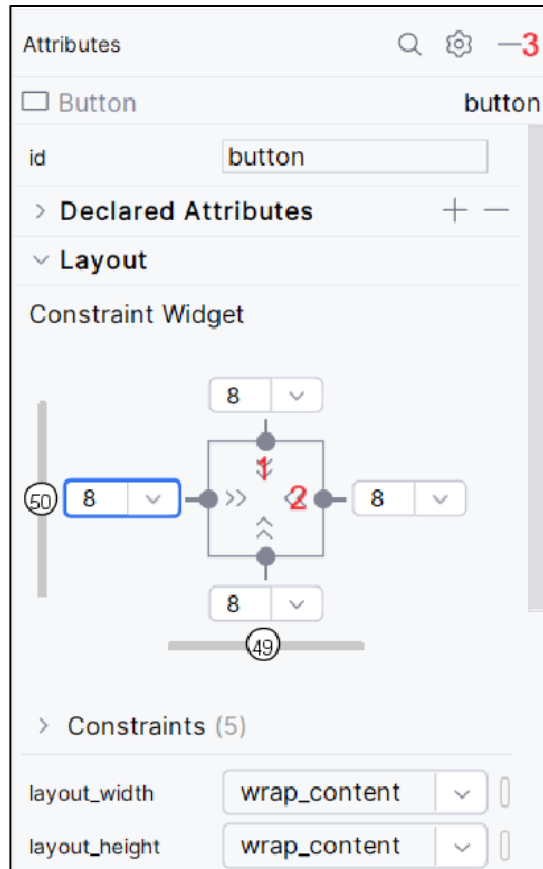
Ngăn **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Bạn có thể tìm thấy các thuộc tính chung cho tất cả các view trong tài liệu lớp View.

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

Thay đổi kích thước Button

Trình chỉnh sửa bố cục cung cấp các nút điều chỉnh kích thước ở cả bốn góc của View để bạn có thể thay đổi kích thước View một cách nhanh chóng. Bạn có thể kéo các nút điều chỉnh ở mỗi góc của View để thay đổi kích thước, nhưng làm như vậy sẽ mã hóa cứng các kích thước chiều rộng và chiều cao. Tránh mã hóa cứng các kích thước cho hầu hết các thành phần View, vì các kích thước được mã hóa cứng không thể đáp ứng với các kích thước nội dung và màn hình khác nhau.

Thay vào đó, hãy sử dụng ngăn **Attributes** ở bên phải của trình chỉnh sửa bố cục để chọn chế độ định cỡ mà không sử dụng các kích thước được mã hóa cứng. **Attributes** bao gồm một bảng định cỡ hình vuông được gọi là trình kiểm tra view ở trên cùng. Các ký hiệu bên trong hình vuông biểu thị các thiết lập chiều cao và chiều rộng như sau:



Trong hình trên:

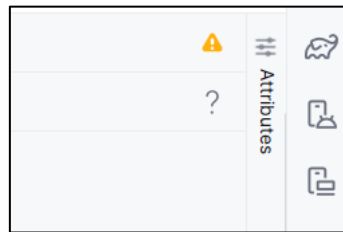
1. **Height control.** Điều khiển này chỉ định thuộc tính **layout_height** và xuất hiện trong hai phân đoạn ở phía trên và phía dưới của hình vuông. Các góc cho biết điều khiển này được đặt thành **wrap_content**, nghĩa là View sẽ mở rộng theo chiều dọc khi cần để phù hợp với nội dung của nó. "8" biểu thị lề chuẩn được đặt thành 8dp
2. **Width control.** Kiểm soát này chỉ định **layout_width** và xuất hiện trong hai phân đoạn ở bên trái và bên phải của hình vuông. Các góc cho biết kiểm soát này được đặt thành **wrap_content**, có nghĩa là View sẽ mở rộng theo chiều ngang khi cần để vừa với nội dung của nó, lên đến biên độ 8dp.
3. Nút đóng ngăn **Attributes**. Nhấn để đóng.

Làm theo các bước sau:

1. Chọn Button từ ngăn **Component Tree**.



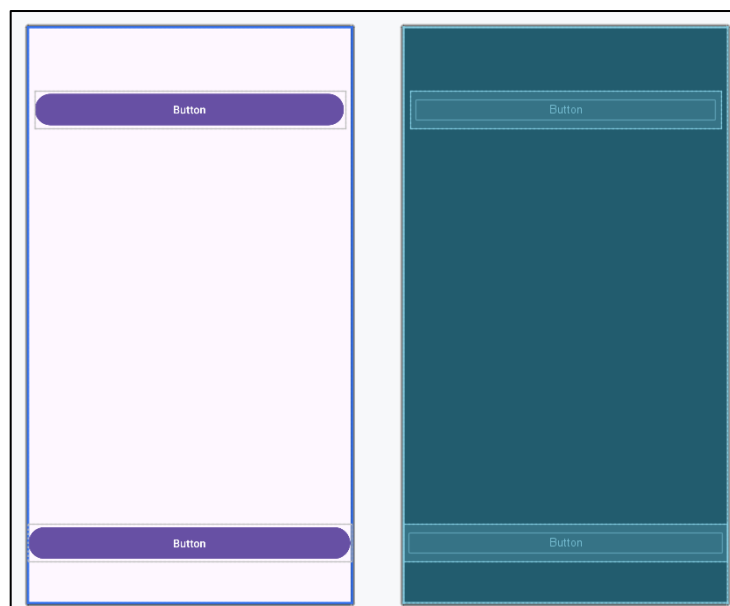
2. Nhấn tab **Attributes** ở cạnh bên phải của cửa sổ trình chỉnh sửa bố cục.



3. Nhấn vào điều khiển chiều rộng hai lần - lần đầu tiên sẽ thay đổi thành Cố định với các đường thẳng và lần thứ thứ hai sẽ thay đổi thành Phù hợp với các ràng buộc với các cuộn lò xo, như được hiển thị trong hình hoạt hình bên dưới.

Kết quả của việc thay đổi điều khiển chiều rộng, thuộc tính **layout_width** trong ngăn **Attributes** hiển thị giá trị **0dp (match_constraint)** và phần tử Button kéo dài theo chiều ngang để lấp đầy khoảng trống giữa bên trái và bên phải của bố cục.

4. Chọn Button thứ hai và làm tương tự để thay đổi **layout_width** như bước 3.



Như đã trình bày trong các bước trước, các thuộc tính **layout_width** và **layout_height** trong ngăn **Attributes** sẽ thay đổi khi bạn thay đổi các điều khiển **height** và **width** trong thanh kiểm tra. Các thuộc tính này có thể lấy một trong ba giá trị cho layout, đó là **ConstraintLayout**:

- Thiết lập **match_constraint** mở rộng phần tử View để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao - lên đến một lề, nếu có. Phần tử cha trong trường hợp này là **ConstraintLayout**. Bạn tìm hiểu thêm về **ConstraintLayout** trong nhiệm vụ tiếp theo.
- Thiết lập **wrap_content** thu nhỏ kích thước của phần tử View sao cho nó chỉ đủ lớn để bao quanh nội dung của nó. Nếu không có nội dung, phần tử View sẽ trở nên vô hình.
- Để chỉ định kích thước cố định điều chỉnh theo kích thước màn hình của thiết bị, hãy sử dụng số cố định pixel không phụ thuộc vào mật độ (đơn vị dp). Ví dụ: 16dp nghĩa là 16 pixel không phụ thuộc vào mật độ.

Thay đổi thuộc tính của Button

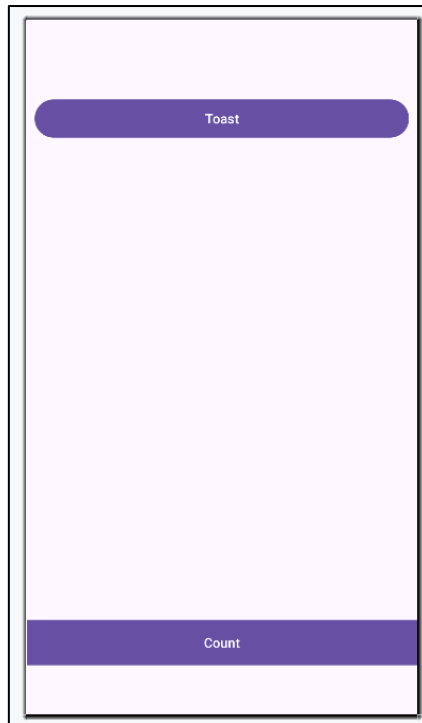
Để xác định mỗi View duy nhất trong bố cục Activity, mỗi View hoặc lớp con View (chẳng hạn như Button) cần một ID duy nhất. Và để có thể sử dụng, các phần tử Button cần có văn bản. Các phần tử View cũng có thể có nền là màu sắc hoặc hình ảnh.

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

Các bước thực hiện:

1. Sau khi chọn Button đầu tiên, hãy chỉnh sửa trường ID ở đầu ngăn Attributes thành **btn_toast** cho thuộc tính **android:id**, được sử dụng để xác định phần tử trong bố cục.
2. Đặt thuộc tính **background** thành **@color/design_default_color_primary**. (Khi bạn nhập **@c**, các lựa chọn sẽ xuất hiện để dễ dàng lựa chọn.)
3. Đặt thuộc tính **textColor** thành **@android:color/white**
4. Sửa thuộc tính **text** thành **Toast**.

5. Thực hiện các thay đổi thuộc tính tương tự cho Button thứ hai, sử dụng **btn_count** làm ID, Count cho thuộc tính text và cùng màu cho nền và văn bản như các bước trước.

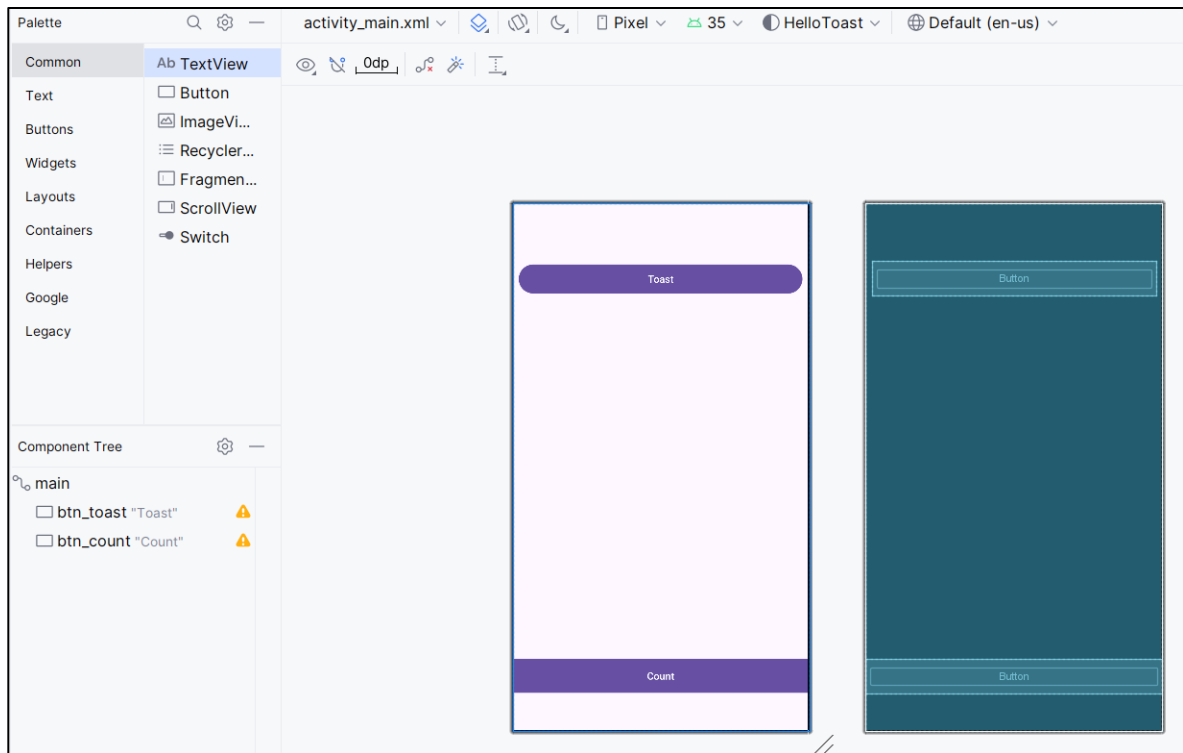


Nhiệm vụ 4: Thêm TextEdit và đặt thuộc tính của nó

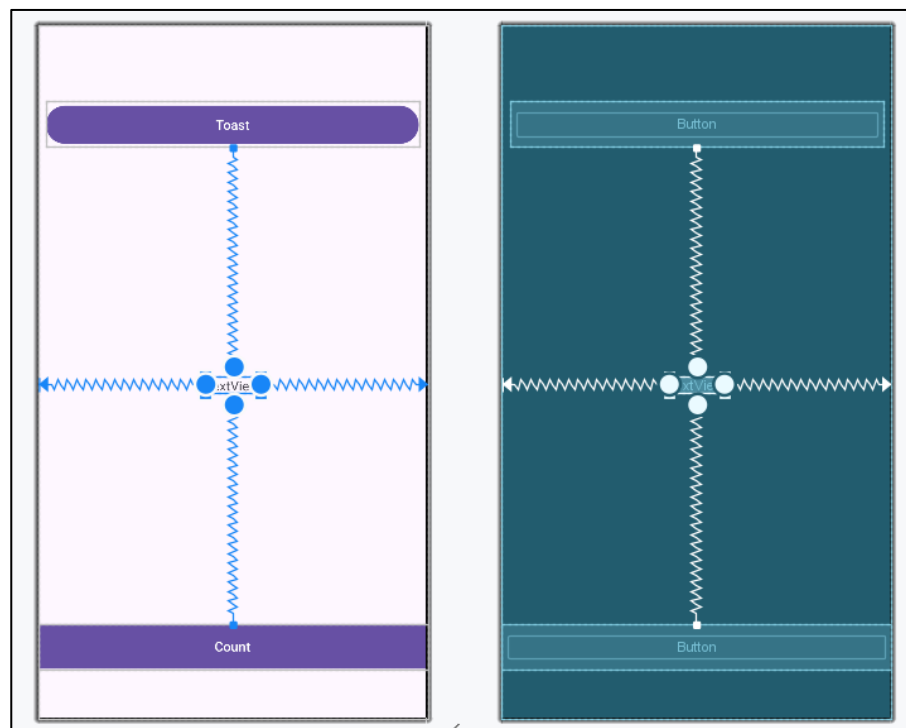
Một lợi ích của **Constraints Layout** có khả năng căn chỉnh hoặc hạn chế các thành phần so với các thành phần khác. Trong nhiệm vụ này bạn sẽ thêm một **TextView** ở giữa bố cục, và giới hạn nó theo chiều ngang so với lề và theo chiều dọc so với hai Button. Sau đó bạn thay đổi các thuộc tính của **TextView** trong ngăn Attributes.

Thêm một TextView và ràng buộc

1. Như được hiển thị trong hình bên dưới, kéo một TextView từ ngăn Palette đến phần trên của bố cục và kéo một ràng buộc từ trên cùng của TextView đến dưới cùng của Button Toast. Thao tác này ràng buộc TextView nằm bên dưới Button.



2. Như được hiển thị trong hình bên dưới, kéo một ràng buộc từ dưới cùng của TextView đến trên cùng của Button Count, và từ cạnh của TextView đến cạnh của bố cục. Điều này ràng buộc TextView nằm giữa bố cục ở giữa hai Button.



Đặt thuộc tính của TextView

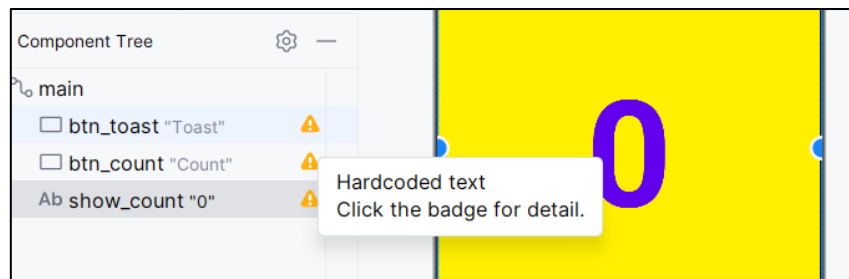
Chọn TextView, mở ngăn Attributes nếu chưa được mở. Đặt các thuộc tính của TextView như sau:

1. Đặt **ID** thành **show_count**.
 2. Đặt **text** thành **0**.
 3. Đặt **textSize** là **160sp**.
 4. Đặt **textStyle** là **bold** và **textAlignment** thành **center**.
 5. Thay đổi điều khiển chiều ngang và chiều dọc của view (**layout_width** và **layout_height**) thành **match_constraint**.
 6. Đặt **textColor** là **@color/design_default_color_primary**.
 7. Kéo xuống cuối cùng trong ngăn **Attributes** và nhấn vào **View all attributes**, kéo xuống trang thứ hai của các thuộc tính đến **background**, và nhập **#FFF000** để có màu vàng.
 8. Kéo xuống đến **gravity**, mở rộng gravity và chọn **center_vertical**.
- **textSize**: Kích thước văn bản của TextView. Đối với bài học này, kích thước được đặt thành 160sp. Sp viết tắt của scale-independent pixel, và giống như dp, là một đơn vị tỷ lệ với mật độ màn hình và giống như 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 sở thích của người dùng.
 - **textStyle** và **textAlignment**: Kiểu văn bản, được đặt thành **bold** trong bài học này và căn chỉnh văn bản, được đặt thành **center**.
 - **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 căn giữa theo chiều dọc trong ConstraintLayout cha.

Nhiệm vụ 5: Chỉnh sửa bố 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 **Component Tree**. Di con trỏ qua các dấu chấm than này để

xem các thông báo cảnh báo, như được hiển thị bên dưới. Cảnh báo tương tự xuất hiện cho cả ba thành phần: “*Hardcoded text. Click the badge for detail.*”.

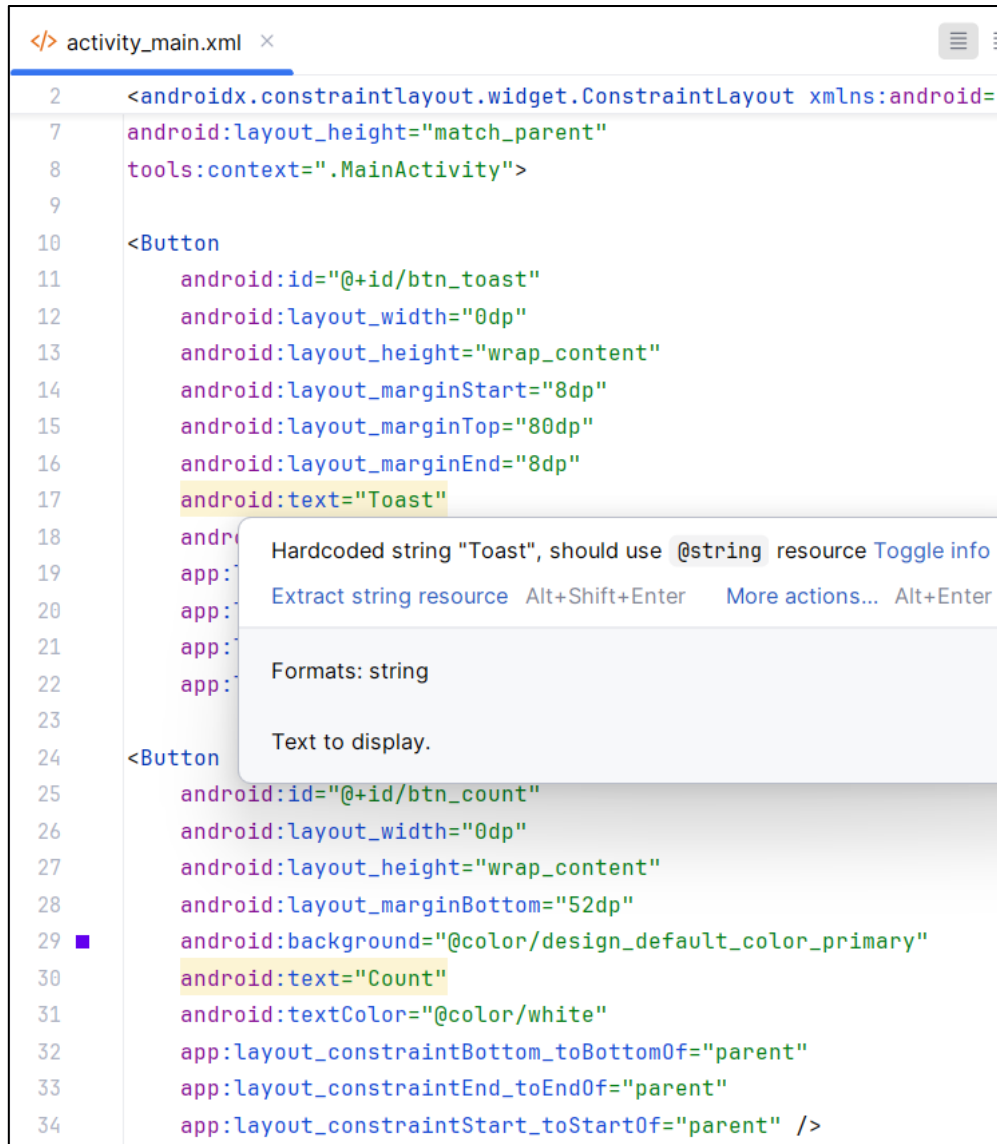


Cách dễ nhất để khắc phục sự cố bố cục là chỉnh sửa bố cục trong XML. Mặc dù trình chỉnh sửa bố cục là một công cụ mạnh mẽ, một số thay đổi dễ thực hiện trực tiếp trong mã nguồn XML hơn.

Mở mã XML của bố cục

Đối với nhiệm vụ này, mở tệp **activity_main.xml** nếu tệp này chưa mở và nhấp vào biểu tượng  ở cuối trình chỉnh sửa 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.



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/btn_toast"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="80dp"
        android:layout_marginEnd="8dp"
        android:text="Toast"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/btn_count"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginBottom="52dp"
        android:background="@color/design_default_color_primary"
        android:text="Count"
        android:textColor="@color/white"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

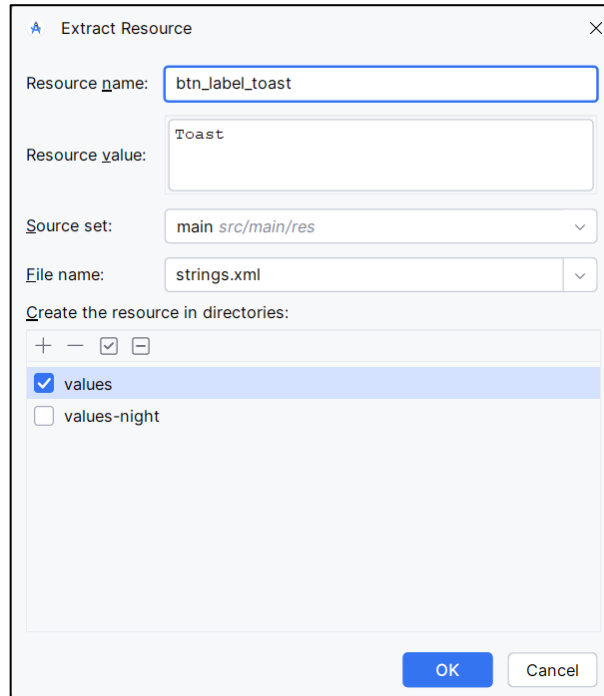
Trích xuất tài nguyên chuỗi

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

1. Nhấn một lần vào từ **“Toast”**.

2. Nhấn tổ hợp phím **Alt+Enter** trên Windows hoặc **Option+Enter** trên macOS và chọn **Extract string resource** từ menu hiện lên.

3. Nhập **btn_label_toast** cho Resource name:



4. Nhấn **OK**. Một tài nguyên chuỗi được tạo trong tệp **values/res/string.xml**, và chuỗi trong mã của bạn được thay thế bằng một tham chiếu đến tài nguyên: **@string/btn_label_toast**.

```
android:text="@string/btn_label_toast"
```

5. Trích xuất cho các chuỗi còn lại: **btn_label_count** cho “Count” và **count_init_value** cho “0”.

6. Trong ngăn **Project > Android**, mở rộng thư mục **res** trong thư mục **values**, sau đó nhấn đúp chuột vào **strings.xml** để xem các tài nguyên chuỗi của bạn trong tệp **strings.xml**.

```
1 <resources>
2     <string name="app_name">Hello Toast</string>
3     <string name="btn_label_toast">Toast</string>
4     <string name="btn_label_count">Count</string>
5     <string name="count_init_value">0</string>
6 </resources>
```

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


```
<string name="toast_message">Hello Toast!</string>
```

Nhiệm vụ 6: Thêm xử lý onClick trên button

Trong nhiệm vụ này, bạn thêm một phương thức Java cho mỗi Button trong MainActivity nhằm thực thi khi người dùng chạm vào Button.

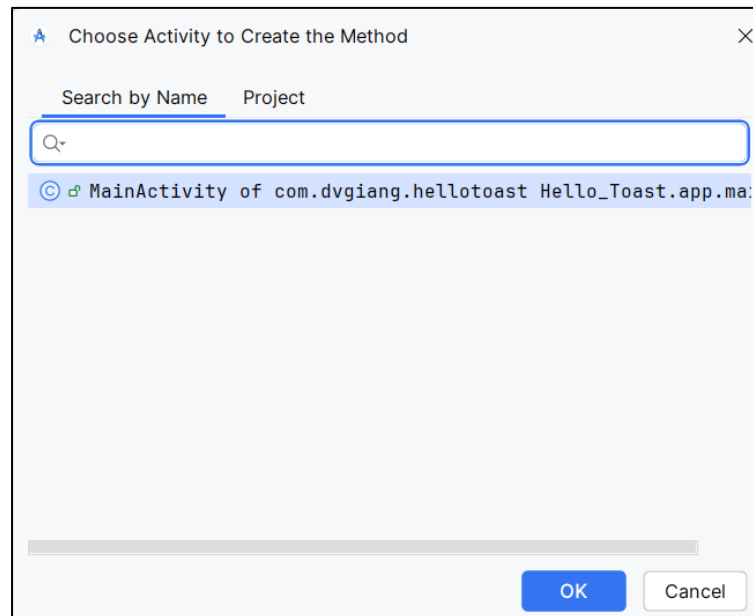
Thêm thuộc tính onClick và xử lý cho mỗi Button

Trình xử lý nhấp là phương thức được gọi khi người dùng nhấn hoặc chạm vào phần tử UI có thể nhấn. Trong Android Studio, bạn có thể chỉ định tên của phương thức trong trường onClick trong ngăn **Attributes** của tab **Design**. 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. Mở trình soạn thảo XML, tìm Button với **android:id** đặt cho **btn_toast**.
2. Thêm thuộc tính **android:onClick** vào cuối của phần tử **btn_toast** sau phần tử cuối cùng và trước chỉ báo kết thúc (**/>**).
3. Nhấp vào biểu tượng bóng đèn đỏ () xuất hiện cạnh thuộc tính. Chọn **Create onClick event handler**, chọn **MainActivity**, và nhấp **OK**.

Nếu biểu tượng bóng đèn đỏ không xuất hiện, nhấp vào tên phương thức (“showToast”). Nhấn tổ hợp phím **Alt+Enter** (**Option+Enter** trên Mac), chọn **Create ‘showToast(view)’ in MainActivity**.

Hành động này tạo ra một phương thức giữ chỗ cho phương thức showToast() trong MainActivity.



4. Lặp lại hai bước trên với btn_count: Thêm thuộc tính android:onClick vào cuối, và thêm xử lý nhấn.

```
android:onClick="increaseCount"/>
```

5. Nếu **MainActivity.java** không mở, mở rộng thư mục java trong **Project > Android**, mở rộng **com.dvgiang.hellotoast**, sau đó nhấp chuột vào **MainActivity**. Trình soạn thảo mã xuất hiện với mã trong MainActivity.

```

package com.dvgiang.hellotoast;

import android.os.Bundle;
import android.view.View;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_main);
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
        (v, insets) -> {
            Insets systemBars =
insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,
systemBars.bottom);
            return insets;
        });
    }

    public void showToast(View view) {}

    public void increaseCount(View view) {}
}

```

Chỉnh sửa trình xử lý nút Toast

Bây giờ bạn sẽ chỉnh sửa phương thức `showToast()` - trình xử lý nhấp chuột của nút Toast trong `MainActivity` - để nó hiển thị một thông báo. Toast cung cấp cách để hiển thị một thông báo đơn giản trong một cửa sổ nhỏ bật lên. 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 cho nút Toast:

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

```
public void showToast(View view) { }
```

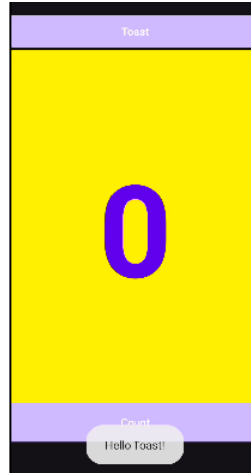
2. Tạo một thể hiện của Toast, gọi là phương thức **`makeText()`** của lớp **`Toast`**.
3. Cung cấp ngữ cảnh của Activity. Vì Toast hiển thị ở đầu Activity UI, hệ thống cần thông tin về Activity hiện tại. Khi bạn đã ở trong ngữ cảnh của Activity mà bạn cần ngữ cảnh, hãy sử dụng điều này như một phím tắt.
4. Cung cấp thông điệp để hiển thị, như tài nguyên chuỗi (**`toast_message`** mà bạn đã tạo trước đó). Tài nguyên chuỗi **`toast_message`** được xác định bởi **`R.string`**.
5. Cung cấp thời gian hiển thị. Ví dụ, **`Toast.LENGTH_SHORT`** hiển thị thông báo trong thời gian tương đối ngắn.

Thời gian hiển thị của **`Toast`** có thể là **`Toast.LENGTH_LONG`** hoặc **`Toast.LENGTH_SHORT`**. Thời gian hiển thị khoảng 3.5 giây cho Toast kiểu **`long`** và 2 giây cho kiểu **`short`**.

6. Hiển thị Toast bằng cách gọi **`show()`**. Dưới đây là phương thức `showToast()` đã hoàn thành.

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

Chạy ứng dụng và xác minh rằng thông điệp Toast đã xuất hiện khi nút Toast được nhấn.



Chỉnh sửa trình xử lý nút Count

Bây giờ bạn sẽ chỉnh sửa phương thức **increaseCount()** - trình xử lý nhấp chuột của nút **Count** trong MainActivity - để nó hiển thị số đếm hiện tại sau khi nút **Count** được nhấn. Mỗi lần nhấn sẽ tăng biến đếm lên một đơn vị.

Trong mã xử lý phải:

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

Làm theo các bước sau để sửa trình xử lý nhấn của nút **Count**:

1. Xác định phương thức **increaseCount()** mới được tạo.
2. Theo dõi số đếm, bạn cần một biến thành viên **private**. Mỗi lần nhấn của nút **Count** tăng giá trị của biến này. Nhập thông tin sau sẽ hiện cảnh đánh dấu đỏ và hiện bóng đèn cảnh báo đỏ:

```
public void increaseCount(View view) {  
    count ++;  
}
```


3. Nhấp vào biểu tượng bóng đèn đỏ và chọn **Create field 'count' in MainActivity** từ menu hiện lên. Việc này tạo một biến thành viên private ở trên cùng của MainActivity, và Android Studio giả định bạn muốn tạo nó là một biến kiểu **int**.

```
public class MainActivity extends AppCompatActivity {  
    1 usage  
    private int count;  
}
```

4. Thay đổi câu lệnh biến private để khởi tạo giá trị của biến thành 0.

```
public class MainActivity extends AppCompatActivity {  
    1 usage  
    private int count = 0;  
}
```

5. Cùng với biến ở trên, bạn cũng cần một biến private để tham chiếu đến TextView **show_count**, cái mà bạn sẽ thêm vào trình xử lý nhấn. Nó có tên là **showCount**.

```
public class MainActivity extends AppCompatActivity {  
    1 usage  
    private int count = 0;  
    no usages  
    private TextView showCount;  
}
```

6. Bây giờ bạn có **showCount**, bạn có thể tham chiếu đến TextView bằng ID mà bạn đặt trong tệp bố cục. Để chỉ lấy tham chiếu một lần, hãy chỉ định cụ thể trong phương thức **onCreate()**. Như đã học trong bài học khác, **onCreate()** được dùng để thổi phồng bố cục, nghĩa là đặt nội dung của màn hình thành bố cục. Bạn cũng có thể sử dụng nó để tham chiếu đến các phần tử UI khác, như TextView.

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);  
    setContentView(R.layout.activity_main);  
}
```

7. Thêm lệnh **findViewById** vào cuối phương thức.

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    EdgeToEdge.enable( $this$enableEdgeToEdge: this);  
    setContentView(R.layout.activity_main);  
    showCount = findViewById(R.id.show_count);  
}
```

Một View, như một chuỗi, một tài nguyên có thể có một ID. Lệnh gọi findViewById lấy ID của một View làm tham số và trả về View đó. Vì phương thức trả về một View, bạn phải ép kiểu kết quả thành loại View mà bạn muốn, trong trường hợp này là TextView.

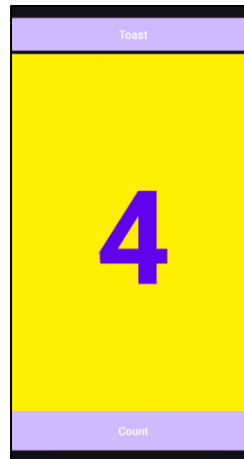
8. Bây giờ bạn đã gán TextView cho showCount, bạn có thể sử dụng biến để đặt text trong TextView thành giá trị của biến count. Thêm như sau vào phương thức increaseCount():

```
if (showCount != null) {  
    showCount.setText(Integer.toString(count));  
}
```

Toàn bộ phương thức sẽ như thế này:

```
public void increaseCount(View view) {  
    count++;  
    if (showCount != null) {  
        showCount.setText(Integer.toString(count));  
    }  
}
```

9. Chạy ứng dụng để xác minh số đếm tăng lên khi bạn nhấn nút **Count**.




Tóm tắt

View, ViewGroup, và bố cục:

- Tất cả các thành phần UI đều là lớp con của lớp View và do đó thừa hưởng nhiều thuộc tính của lớp View.
- Các phần tử View có thể được nhóm bên trong một ViewGroup, hoạt động như một container. Mỗi quan hệ là **parent-child**, trong đó parent là một ViewGroup, và child là một View hoặc một ViewGroup khác.
- Phương thức **onCreate()** được sử dụng để làm phòng bố cục, nghĩa là đặt chế độ xem nội dung của màn hình thành bố cục XML. Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các thành phần UI khác trong bố cục.
- View, giống như một chuỗi, là một tài nguyên có thể có một id. Lệnh gọi **findViewById** lấy ID của view làm tham số và trả về View.

Sử dụng trình chỉnh sửa bố cục:

- Nhấp vào tab Design để thao tác các thành phần và bố cục, và tab Text để chỉnh sửa mã XML cho bố cục.
- Trong tab **Design**, ngăn **Palettes** hiển thị các thành phần UI mà bạn có thể sử dụng trong bố cục ứng dụng của mình, và ngăn **Component tree** hiển thị phân cấp chế độ xem của các thành phần UI.

- Các ngăn design và bluesprint của trình chỉnh sửa bố cục hiển thị các thành phần UI trong bố cục.
- Tab **Attributes** hiển thị ngăn **Attributes** để thiết lập thuộc tính cho một phần tử UI.
- **Constraint handle**: Nhấp vào **constraint handle**, được hiển thị dưới dạng hình tròn ở mỗi bên của phần tử, sau đó kéo đến **constraint handle** khác hoặc đến ranh giới cha để tạo ràng buộc. Ràng buộc được biểu thị bằng đường ngoằn ngoèo.
- **Resizing handle**: Bạn có thể kéo **resizing handle** hình vuông để thay đổi kích thước phần tử. Trong khi kéo, handle sẽ thay đổi thành góc nghiêng.
- Bạn có thể xóa các ràng buộc khỏi một phần tử bằng cách chọn phần tử đó nhấp vào biểu tượng  để xóa tất cả các ràng buộc trên phần tử đã chọn.
- Ngăn **Attributes** cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử UI. Ngăn này cũng bao gồm một bảng điều khiển kích thước hình vuông được gọi là trình kiểm tra view ở trên cùng. Các ký hiệu bên trong hình vuông biểu thị các thiết lập chiều cao và chiều rộng.

Cài đặt chiều rộng và chiều cao bố cục:

Các thuộc tính **layout_width** và **layout_height** thay đổi khi bạn thay đổi kích thước chiều cao và chiều rộng controls trong view inspector. Các thuộc tính này có thể lấy một trong ba giá trị cho một ConstraintLayout:

- Thiết lập **match_constraint** mở rộng view để lấp đầy phần tử cha theo chiều rộng hoặc chiều cao - lên đến lề, nếu có.
- Thiết lập **wrap_content** thu nhỏ kích thước view để view chỉ đủ lớn để bao quanh nội dung của nó. Nếu không có nội dung, chế độ xem sẽ trở nên vô hình.
- Sử dụng số **dp** (density-independent pixels) cố định để chỉ định kích thước cố định, được điều chỉnh cho kích thước màn hình của thiết bị.

Trích xuất tài nguyên chuỗi:

Thay vì mã hóa cứng các chuỗi, cách tốt nhất là sử dụng tài nguyên chuỗi, đại diện cho chuỗi. Thực hiện theo các bước sau:

1. Nhấp một lần vào chuỗi được mã hóa cứng để trích xuất, nhấn **Alt-Enter (Option-Enter)** trên máy Mac) và chọn **Create string value resource** từ menu bật lên.

2. Đặt **Resource value**.

3. Nhấn **OK**. Điều này tạo ra một tài nguyên chuỗi trong tệp **values/res/string.xml** và chuỗi trong mã của bạn được thay thế bằng tham chiếu đến tài nguyên đó: **@string/button_label_toast**

Xử lý nhấn:

- Trình xử lý nhấn là một phương thức được gọi khi người dùng nhấn hoặc chạm vào một thành phần UI.
- Chỉ định trình xử lý nhấp cho một thành phần UI như nút bằng cách nhập tên của thành phần đó vào trường **onClick** trong ngăn **Attributes** của tab **Design** hoặc trong trình soạn thảo XML bằng cách thêm thuộc tính **android:onClick** vào một thành phần UI như Button.
- Tạo trình xử lý nhấp trong **Activity** chính bằng cách sử dụng tham số View. Ví dụ: `public void showToast(View view) {...}`.

Hiển thị thông điệp Toast:

Toast cung cấp 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. Để tạo một phiên bản Toast, hãy làm theo các bước sau:

1. Gọi phương thức **makeText()** trên lớp **Toast**.
2. Cung cấp ngữ cảnh của Activity và thông điệp để hiển thị (như một tài nguyên chuỗi)
3. Cung cấp thời lượng hiển thị, ví dụ **Toast.LENGTH_SHORT** cho chu kỳ ngắn. Thời lượng có thể là **Toast.LENGTH_LONG** hoặc **Toast.LENGTH_SHORT**.
4. Hiển thị Toast bằng cách gọi **show()**.

1.3) Trình chỉnh sửa bố cục

Giới thiệu

Như bạn đã học trong **1.2: Giao diện người dùng tương tác đầu tiên**, 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 một ViewGroup, là một View đặc biệt có thể chứa các đối tượng View khác (gọi là children hoặc child views). Bài thực hành này sẽ cho thấy nhiều tính năng hơn của ConstraintLayout và trình chỉnh sửa bố cục.

Bài thực hành này cũng giới thiệu hai lớp con ViewGroup khác:

- **LinearLayout**: Một nhóm sắp xếp các phần tử View con bên trong theo chiều ngang hoặc chiều dọc.
- **RelativeLayout**: Một nhóm các phần tử View con trong đó mỗi phần tử View được định vị và căn chỉnh tương đối với phần tử View khác trong ViewGroup. Vị trí của các phần tử View con được mô tả liên quan đến nhau hoặc đến ViewGroup cha.

Những gì bạn nên biết:

Bạn nên biết:

- Tạo ứng dụng Hello World với Android Studio.
- Chạy ứng dụng trên máy ảo hoặc máy thật.
- Tạo một bố cục đơn giản cho ứng dụng với ConstraintsLayout.
- Trích xuất và sử dụng tài nguyên chuỗi.

Những gì bạn sẽ học:

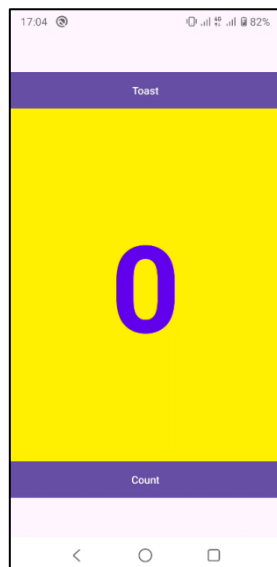
- Cách tạo biến thể bố cục theo hướng ngang.
- Cách tạo biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Cách sử dụng ràng buộc đường cơ sở để căn chỉnh các thành phần UI với văn bản.
- Cách sử dụng các nút đóng gói và căn chỉnh để căn chỉnh các thành phần trong bố cục.
- Cách định vị chế độ xem trong LinearLayout.
- Cách định vị chế độ xem trong RelativeLayout.

Những gì bạn sẽ làm:

- Tạo một biến thể bố cục cho hướng hiển thị ngang.
- Tạo một biến thể bố cục cho máy tính bảng và màn hình lớn hơn.
- Sửa đổi bố cục để thêm ràng buộc vào các thành phần UI.
- Sử dụng ràng buộc cơ sở ConstraintLayout để căn chỉnh các thành phần với văn bản.
- Sử dụng ConstraintLayout và các nút căn chỉnh để căn chỉnh các thành phần.
- Thay đổi bố cục để sử dụng LinearLayout.
- Định vị các thành phần trong LinearLayout.
- Thay đổi bố cục để sử dụng RelativeLayout.
- Sắp xếp lại các chế độ xem trong bố cục chính để tương đối với nhau.

Tổng quan

Ứng dụng Hello Toast trong bài học trước sử dụng Constraints sắp xếp các thành phần UI trên bố cục Activity.



Để thực hành nhiều hơn với ConstraintLayout, bạn sẽ tạo một biến thể của bố cục này theo hướng ngang.

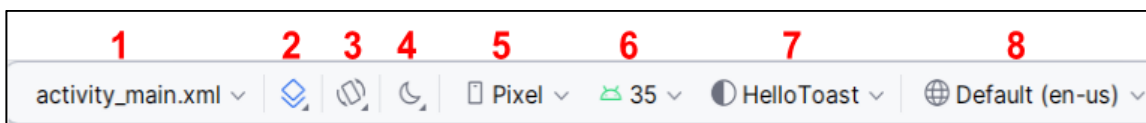
Bạn cũng sẽ học cách sử dụng các ràng buộc cơ sở và một số tính năng căn chỉnh của `ConstraintLayout` bằng cách tạo một biến thể bố cục khác cho màn hình máy tính bảng.

Bạn cũng tìm hiểu về các lớp con `ViewGroup` khác như `LinearLayout` và `RelativeLayout`, và thay đổi bố cục ứng dụng `Hello Toast` để sử dụng chúng.

Nhiệm vụ 1: Tạo biến thể bố cục

Trong bài học trước, thử thách mã hóa yêu cầu thay đổi bố cục của ứng dụng `Hello Toast` để nó có thể vừa vận theo hướng ngang hoặc dọc. Trong nhiệm vụ này, bạn sẽ học cách dễ dàng hơn để tạo các biến thể bố cục của mình theo hướng ngang và hướng dọc cho điện thoại và cho màn hình lớn hơn như máy tính bảng.

Trong nhiệm vụ này, bạn sẽ sử dụng một vài nút ở các thanh công cụ của trình chỉnh sửa bố cục.

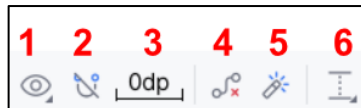


Trong ảnh trên gồm:

1. Tên tệp bố cục đang thao tác
2. **Select Design Surface**: Chọn **Design** để hiển thị bản xem trước màu của bố cục hoặc **Blueprint** để chỉ hiển thị các phác thảo cho từng thành phần UI. Để xem cả hai gần cạnh nhau, chọn **Design + Blueprint**.
3. **Orientation for Preview**: 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.
4. **System UI Mode**: Chế độ hiển thị của hệ thống. Chọn **Not Night** để xem giao diện sáng, **Night** để xem giao diện tối.
5. **Device for Preview**: Chọn loại thiết bị.
6. **API Version for Preview**: Chọn phiên bản Android sử dụng để hiển thị bản xem trước.
7. **Theme for Preview**: Chọn một chủ đề để áp dụng cho bản xem trước.

8. **Locale for Preview:** Chọn ngôn ngữ và bản địa hóa để 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 **Preview Right to Left** để xem bố cục như thể đã chọn ngôn ngữ.

Thanh công cụ thứ hai cho phép bạn cấu hình giao diện của các thành phần UI trong ConstraintLayout:



Trong hình trên:

1. **View Options:** Chọn **Show All Constraints** và **Show Margins** để hiển thị chúng trong bản xem trước hoặc dừng hiển thị chúng.
2. **Autoconnect:** Bật hoặc tắt **Autoconnect**. Khi được bật, bạn có thể kéo bất kỳ phần tử nào (chẳng hạn như Button) đến bất kỳ phần nào của bố cục để tạo ràng buộc đối với bố cục cha.
3. **Default Margins:** Đặt độ rộng lề mặc định cho bố cục.
4. **Clear All Constraints:** Xóa tất cả ràng buộc.
5. **Infer Constraints:** Tạo ràng buộc bằng suy luận.
6. **Guidelines:** Thêm đường hướng dẫn theo chiều dọc hoặc chiều ngang.

Thanh công cụ thứ ba cho phép bạn phóng to và thu nhỏ bản xem trước:

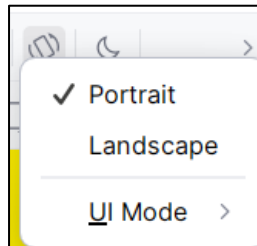


1. **Pan screen:** Chế độ con trỏ trong trình chỉnh sửa bố cục
2. **Zoom In:** Phóng to bố cục
3. **Zoom Out:** Thu nhỏ bố cục

Xem trước bố cục theo hướng ngang

Để xem trước bố cục ứng dụng Hello Toast theo hướng ngang, làm theo các bước sau:

1. Mở ứng dụng Hello Toast từ bài học trước.
2. Mở tệp **activity_main.xml**. Nhấp vào tab **Design** nếu chưa được chọn.
3. Nhấp vào nút **Orientation for Preview** trên thanh công cụ.
4. Chọn **Landscape** từ menu thả xuống. Bố cục xuất hiện theo hướng ngang. Để quay lại hướng dọc, chọn **Portrait**.

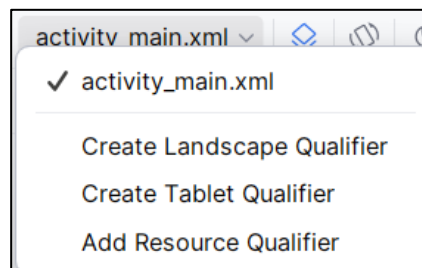


Tạo biến thể bố cục theo hướng ngang

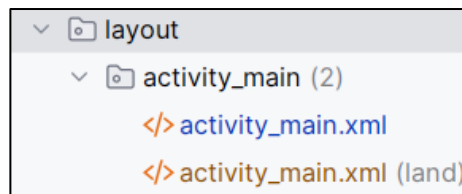
Sự khác biệt trực quan giữa hướng dọc và hướng ngang cho bố cục này là chữ số (0) trong phần tử TextView quá thấp so với hướng ngang - quá gần nút **Count**. Tùy thuộc vào thiết bị hoặc trình giả lập bạn sử dụng, phần tử TextView có thể xuất hiện quá lớn hoặc không được căn giữa vì kích thước văn bản được cố định ở mức 160sp.

Để khắc phục điều này đối với hướng ngang trong khi vẫn giữ nguyên hướng dọc, bạn có thể tạo biến thể của bố cục ứng dụng Hello Toast khác với hướng ngang. Thực hiện theo các bước sau:

1. Nhấp vào biểu tượng activity_main.xml trên thanh công cụ.
2. Chọn **Create Landscape Qualifier** từ menu thả xuống.



3. Trong **Project > Android**, bên trong thư mục **res > layout**, bạn sẽ thấy Android Studio tự động tạo biến thể cho bạn, được gọi là **activity_main.xml (land)**.



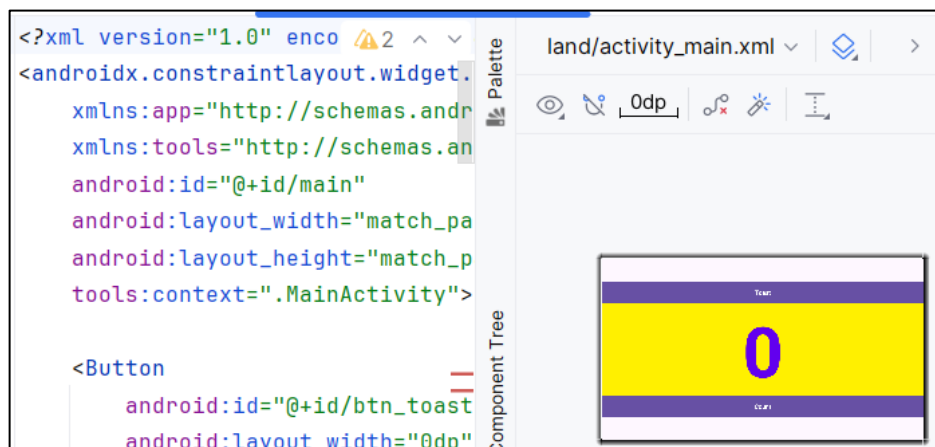
Xem trước bố cục cho các thiết bị khác nhau

Bạn có thể xem trước bố cục cho các thiết bị khác nhau mà không phải chạy ứng dụng trên thiết bị hoặc trình giả lập. Làm theo các bước sau:


1. Mở tệp **land/activity_main.xml** trong trình chỉnh sửa bố cục.
2. Nhấp vào **Devices for Preview** trên thanh công cụ.
3. Chọn một thiết bị khác trong menu thả xuống. Những khác biệt này là do kích thước văn bản cố định cho TextView.

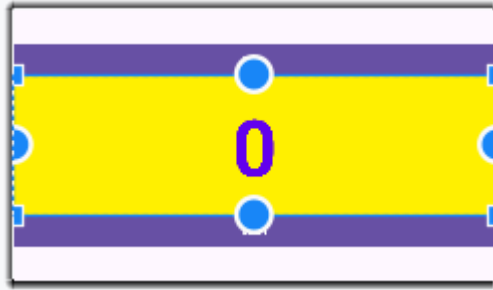
Thay đổi bố cục theo hướng ngang

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



Để thay đổi bố cục, làm theo các bước sau:

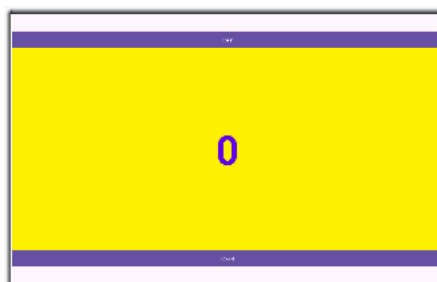
1. Mở tệp **land/activity_main.xml** trong trình chỉnh sửa bố cục.
2. Nhấp vào biểu tượng  trên thanh công cụ.
3. Tìm đến thành phần **TextView** trong mã XML.
4. Thay đổi thuộc tính **android:textSize="160sp"** thành **android:textSize="120sp"**.
Bố cục xem trước hiện kết quả.



5. Chọn các thiết bị khác nhau trong menu thả xuống của **Devices for Preview** để xem bố cục trông như thế nào trên các thiết bị khác nhau theo hướng ngang.
6. Chạy ứng dụng trên trình giả lập hoặc thiết bị và chuyển hướng từ dọc sang ngang để xem cả hai bố cục.

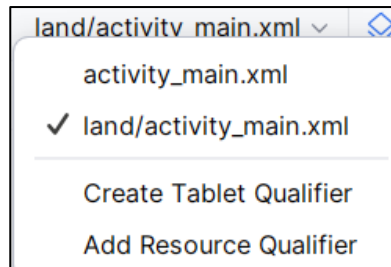
Tạo biến thể bố cục cho tablets

Như bạn đã học trước đó, bạn có thể xem trước bố cục cho các thiết bị khác nhau bằng cách nhấp vào nút **Devices for Preview** 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 bố 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 điều này cho máy tính bảng trong khi vẫn giữ nguyên hướng ngang và dọc của kích thước điện thoại, bạn có thể tạo biến thể của bố cục hoàn toàn khác cho máy tính bảng. Thực hiện theo các bước sau:

1. Nhấp vào tab **Design** nếu chưa được mở để hiển thị ngăn **Design + Blueprint**
2. Nhấp vào **land/activity_main.xml** trên thanh công cụ và chọn **Create Tablet Qualifier** từ menu thả xuống.



Một cửa sổ mới mở ra với tab **sw600dp/activity_main.xml** hiển thị bố 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 10, để xem trước. Bạn có thể thay đổi bố cục này, dành riêng cho máy tính bảng, mà không cần thay đổi các bố cục khác.

Thay đổi biến thể bố cục cho tablets

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

1. Tắt công cụ **Autoconnect** 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 bố cục bằng cách nhấp vào biểu tượng **Clear All Constraints** trên thanh công cụ.

Khi các ràng buộc được gỡ bỏ, bạn có thể di chuyển và thay đổi kích thước các thành phần trên bố cục một cách tự do

3. Trình chỉnh sửa bố cục cung cấp các nút điều khiển thay đổi kích thước ở cả bốn góc của một phần tử để thay đổi kích thước của phần tử đó. Trong **Component Tree**,

chọn TextView có tên **show_count**. Để TextView không cản trở bạn có thể tự do kéo các phần tử Button, hãy kéo một góc của phần tử đó để thay đổi kích thước.

Thay đổi kích thước của một phần tử sẽ mã hóa cứng các kích thước chiều rộng và chiều cao. Tránh mã hóa cứng các kích thước cho hầu hết các phần tử, vì bạn không thể dự đoán các kích thước được mã hóa cứng sẽ như thế nào trên các màn hình có kích thước và mật độ khác nhau. Bạn đang thực hiện việc này ngay bây giờ chỉ để di chuyển phần tử ra khỏi đường đi và bạn sẽ thay đổi các kích thước trong một bước khác.

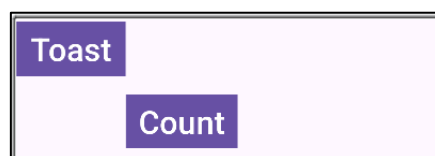
4. Chọn button **btn_toast** trong **Component Tree**, nhấp vào tab **Attributes** để mở ngăn **Attributes** và thay đổi **textSize** thành **60sp** và thay đổi **layout_width** thành **wrap_content**.


5. Chọn button **btn_count** trong **Component Tree**, thay đổi **textSize** thành **60sp** và thay đổi **layout_width** thành **wrap_content**, và kéo button phía trên TextView đến một khoảng trống trong bố cục.

Sử dụng ràng buộc cơ bản

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.


1. Giới hạn nút **btn_toast** ở phía trên và bên trái của bố cục, kéo nút **btn_count** đến khoảng trống gần nút **btn_toast** và giới hạn nút **btn_count** ở phía bên trái của nút **btn_toast**.



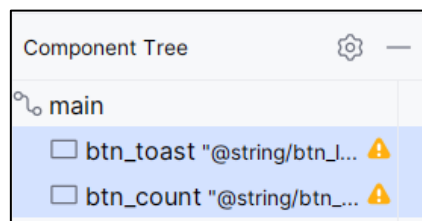
2. Sử dụng ràng buộc đường cơ sở, bạn có thể ràng buộc **btn_count** sao cho đường cơ sở văn bản của nó khớp với đường cơ sở văn bản của **btn_toast**. Chọn phần tử **btn_count**, sau đó nhấp chuột phải để hiển thị menu, nhấp vào biểu tượng  Show Baseline để hiển thị base line.


3. Nhấp và kéo đường ràng buộc đường cơ sở đến đường cơ sở của phần tử **btn_toast**.

Mở rộng các nút theo chiều ngang

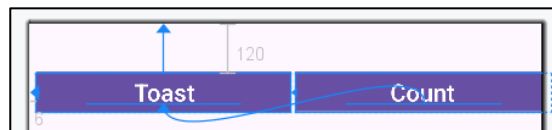
Nút  **Organize** cung cấp các tùy chọn để đóng gói hoặc mở rộng các thành phần UI đã chọn. Bạn có thể sử dụng nút này để sắp xếp đều các thành phần Button theo chiều ngang trên toàn bộ bố cục.

1. Chọn nút **btn_count** trong **Component Tree** và nhấn **Shift+nhấn chuột** vào **btn_toast** để cả hai được chọn.



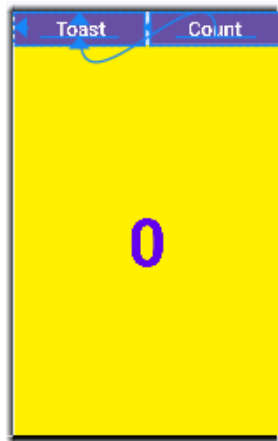
2. Nhấn chuột phải và chọn  **Organize** trong menu hiện lên, và chọn **Expand Horizontally**.

Các phần tử sẽ lấp đầy bố cục như hiển thị bên dưới:



3. Để hoàn thiện bố cục, hãy ràng buộc TextView **show_count** vào cuối **btn_toast** và vào các cạnh và cuối của bố cục.

4. Các bước cuối cùng là thay đổi **layout_width** và **layout_height** của TextView **show_count** thành **Match Constraints** và **textSize** thành **200sp**.



5. Thử trên bố cục ngang và dọc khác nhau.

6. Chạy ứng dụng trên các trình giả lập (hoặc thiết bị) 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.

Nhiệm vụ 2: Thay đổi bố cục thành **LinearLayout**

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 yêu cầu các thuộc tính này:

- `layout_width`
- `layout_height`
- `orientation`

Thuộc tính **`layout_width`** và **`layout_height`** có thể lấy 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, chế độ xem này 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 chỉ 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: 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 vào mật độ.

Thuộc tính **orientation** có thể là:

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

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

Thay đổi ViewGroup gốc thành LinearLayout

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

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="....."
```

3. Thay đổi mã XML thành **LinearLayout**:

```
<LinearLayout xmlns:android="....."
```

4. Đảm bảo thẻ đóng ở cuối mã đã thay đổi thành **</LinearLayout>**. Nếu thẻ chưa tự động thay đổi, hãy thay đổi thủ công.
5. Dưới dòng tag **<LinearLayout**, thêm thuộc tính sau **android:layout_height**:

```
android:orientation="vertical"
```

Sau khi thực hiện những thay đổi này, một số thuộc tính XML cho các phần tử khác được gạch chân màu đỏ vì chúng được sử dụng với ConstraintLayout và không liên quan đến LinearLayout.

Thay đổi các thuộc tính của phần tử trong LinearLayout

Làm theo các bước sau để thay đổi các thuộc tính cho các thành phần UI làm việc trong LinearLayout:

1. Mở ứng dụng **Hello Toast** từ nhiệm vụ trước.
2. Mở tệp **activity_main.xml** (nếu chưa mở), và nhấn vào tap Text.
3. Tìm đến nút **btn_toast**, thay đổi các thuộc tính sau:

Original	Change to
android:layout_width="0dp"	android:layout_width="match_parent"

4. Xóa các thuộc tính sau khỏi phần tử **button_toast**:

```
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
```

5. Tìm đến nút **btn_count**, thay đổi các thuộc tính sau:

Original	Change to
android:layout_width="0dp"	android:layout_width="match_parent"

6. Xóa các thuộc tính sau khỏi phần tử **button_count**:

```
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
```

7. Tìm đến TextView **show_count**, và thay đổi các thuộc tính sau:

Original	Change to
android:layout_width="0dp"	android:layout_width="match_parent"
android:layout_height="0dp"	android:layout_height="wrap_content"

8.

Thay đổi thuộc tính phần tử UI

3. Kéo một **Button** khác từ ngăn **Palette** vào giữa bố cục.

1.4) Văn bản và các chế độ cuộn

1.5) Tài nguyên có sẵn

Bài 2) Activities

2.1) Activity và Intent

2.2) Vòng đời của Activity và trạng thái

2.3) Intent ngầm định

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

3.1) Trình gỡ lỗi

3.2) Kiểm thử đơn vị

3.3) Thư viện hỗ trợ

CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG

Bài 1) Tương tác người dùng

1.1) Hình ảnh có thể chọn

1.2) Các điều khiển nhập liệu

1.3) Menu và bộ chọn

1.4) Điều hướng người dùng

1.5) RecyclerView

Bài 2) Trải nghiệm người dùng thú vị

2.1) Hình vẽ, định kiểu và chủ đề

2.2) Thẻ và màu sắc

2.3) Bố cục thích ứng

Bài 3) Kiểm thử giao diện người dùng

3.1) Espresso cho việc kiểm tra UI

CHƯƠNG 3. LÀM VIỆC TRONG NỀN

Bài 1) Các tác vụ nền

1.1) AsyncTask

1.2) AsyncTask và AsyncTaskLoader

1.3) Broadcast receivers

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

2.1) Thông báo

2.2) Trình quản lý cảnh báo

2.3) JobScheduler

CHƯƠNG 4. LƯU DỮ LIỆU NGƯỜI DÙNG

Bài 1) Tùy chọn và cài đặt

1.1) Shared preferences

1.2) Cài đặt ứng dụng

Bài 2) Lưu trữ dữ liệu với Room

2.1) Room, LiveData và ViewModel

2.2) Room, LiveData và ViewModel