**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](#_Toc190855147)

[Bài 1) Tạo ứng dụng đầu tiên 4](#_Toc190855148)

[1.1) Android Studio và Hello World 4](#_Toc190855149)

[1.2) Giao diện người dùng tương tác đầu tiên 5](#_Toc190855150)

[1.3) Trình chỉnh sửa bố cục 5](#_Toc190855151)

[1.4) Văn bản và các chế độ cuộn 5](#_Toc190855152)

[1.5) Tài nguyên có sẵn 5](#_Toc190855153)

[Bài 2) Activities 5](#_Toc190855154)

[2.1) Activity và Intent 5](#_Toc190855155)

[2.2) Vòng đời của Activity và trạng thái 5](#_Toc190855156)

[2.3) Intent ngầm định 5](#_Toc190855157)

[Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ 5](#_Toc190855158)

[3.1) Trình gỡ lỗi 5](#_Toc190855159)

[3.2) Kiểm thử đơn vị 5](#_Toc190855160)

[3.3) Thư viện hỗ trợ 5](#_Toc190855161)

[CHƯƠNG 2. Trải nghiệm người dùng 6](#_Toc190855162)

[Bài 1) Tương tác người dùng 6](#_Toc190855163)

[1.1) Hình ảnh có thể chọn 6](#_Toc190855164)

[1.2) Các điều khiển nhập liệu 6](#_Toc190855165)

[1.3) Menu và bộ chọn 6](#_Toc190855166)

[1.4) Điều hướng người dùng 6](#_Toc190855167)

[1.5) RecycleView 6](#_Toc190855168)

[Bài 2) Trải nghiệm người dùng thú vị 6](#_Toc190855169)

[2.1) Hình vẽ, định kiểu và chủ đề 6](#_Toc190855170)

[2.2) Thẻ và màu sắc 6](#_Toc190855171)

[2.3) Bố cục thích ứng 6](#_Toc190855172)

[Bài 3) Kiểm thử giao diện người dùng 6](#_Toc190855173)

[3.1) Espresso cho việc kiểm tra UI 6](#_Toc190855174)

[CHƯƠNG 3. Làm việc trong nền 6](#_Toc190855175)

[Bài 1) Các tác vụ nền 6](#_Toc190855176)

[1.1) AsyncTask 6](#_Toc190855177)

[1.2) AsyncTask và AsyncTaskLoader 6](#_Toc190855178)

[1.3) Broadcast receivers 6](#_Toc190855179)

[Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền 6](#_Toc190855180)

[2.1) Thông báo 6](#_Toc190855181)

[2.2) Trình quản lý cảnh báo 6](#_Toc190855182)

[2.3) JobScheduler 6](#_Toc190855183)

[CHƯƠNG 4. Lưu dữ liệu người dùng 7](#_Toc190855184)

[Bài 1) Tùy chọn và cài đặt 7](#_Toc190855185)

[1.1) Shared preferences 7](#_Toc190855186)

[1.2) Cài đặt ứng dụng 7](#_Toc190855187)

[Bài 2) Lưu trữ dữ liệu với Room 7](#_Toc190855188)

[2.1) Room, LiveData và ViewModel 7](#_Toc190855189)

[2.2) Room, LiveData và ViewModel 7](#_Toc190855190)

3.1) Trinfh gowx loi ……………………………………………………………………...

# LÀM QUEN

## Tạo ứng dụng đầu tiên

### Android Studio và Hello World

Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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

A screenshot of a computer

AI-generated content may be incorrect.

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**

### Giao diện người dùng tương tác đầu tiên

Introduction

Giao diện người dùng xuất hiện trong màn hình của thiết bị android bao gồm phân cấp của các đối tượng gọi là views – tất cả các element của màn hình là 1 view. Lớp view đại diện cho các khối xây dụng cơ bản cho thành phần UI (giao diện). Và các lớp cơ sở cho các lớp khác cái mà cung cấp thành phần giao diện tương tác như button, checkboxes and text. Thông thường, sử dụng các lớp con view được trình bày qua một vài bài học gồm:

* TextView để hiển thị text;
* EditText để có thể cho phép người dùng nhập và sửa text
* Button
* ScrollView and RecyclerView to display scrollable items
* ImagView để hiển thị ảnh
* ConstraintLayout and LinearLayout đẻ chứa các thuộc tính View và sắp vị trí cho chúng

Code java hiển thị và điều khiểu giao diện được chứa trong 1 lớp kế thừ từ lớp Activity. Activity thường được liên kết với 1 giao diện UI view được định nghĩa như 1 file XML. File xml này thường được đặt tên sau Activity và định nghĩa layout của các đối tượng View trên màn hình

Ví dụ, MainActivity trong Hello World app hiển thị layout được định nghĩa trong activity\_main.xml. Cái mà bao gôm cả TextView chứ chuỗi “ Hello World”

Trong các app phức tạp hơn, 1 activity có thể thực thi các hành động phản hồi tời người dùng khi ấn, vẽ đồ thị nội dung và yêu cầu dữ liệu từ database or internet. Bạn có thể học được nhiều từ lớp Activity trong bài học khác.

Trong bài thực hành này, bạn có thể học cách đẻ tạo chương trình tương tác đầu tiên - app cho phép người dùng tương tác. Và bạn tạo 1 app sử dụng Empty Activity template. Bạn cũng có thể học cách sử dụng trình chỉnh sử để thiết kế 1 layout, và cachs để chỉnh sửa layout này trong file xml. Bạn cần pháp triển những kĩ năng này. Vì vâyj bạn có thể hoàn thành các bài thực hành khác trong khóa học này

#### Task1

**Link github:** [**https://github.com/htpit1106/Android-unit1234**](https://github.com/htpit1106/Android-unit1234)

**Đọc dịch task 1: Tạo và khám phá dự án mới**

Trong luyện tập lần này, bạn sẽ thiết kế và thực thi 1 pro ject cho app HelloToast app. 1 liên kết tới nguồn code được cung cấp ở cuối

Khám phá trình chỉnh sửa layout:

Androi Studio cung cấp trình sử layout để xây dựng 1 layout của giao diện người dùng 1 cách nhanh chóng. Nó cho phép bạn kéo thả các đối tượng tới bảng thiết kế và bản phác họa, căn chỉnh vị trí của chúng trong layout. Thêm các ràng buộc và sét các thuộc tính. 1 Ràng buộc thể hiện cho 1 kết nối và 1 liên kết tới 1 view

Khám phá trình chỉnh sửa layout, và tham khảo hình ảnh dưới:

A screenshot of a computer

Description automatically generated

Thực hiện các bước sau:

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

2.Nếu không thấy bản thiết kế (blueprint), hãy nhấp vào nút Select Design Surface trên thanh công cụ và chọn Design + Blueprint.

Autoconnect tool (công cụ tự động kết nối) cũng nằm trên thanh công cụ và được bật mặc định.

3.Đảm bảo rằng công cụ này không bị tắt trong bước này.

Nhấp vào nút Zoom in để phóng to các bảng thiết kế và bản thiết kế (blueprint) để xem chi tiết.

4.Chọn TextView trong bảng Component Tree.

TextView với dòng chữ "Hello World" sẽ được tô sáng trong cả bảng thiết kế và bản thiết kế, đồng thời các ràng buộc (constraints) của phần tử này cũng sẽ hiện ra.

5.Thực hiện theo hình ảnh động bên dưới cho bước này:

Nhấp vào hình tròn ở bên phải của TextView để xóa ràng buộc ngang (horizontal constraint) đang gắn kết nó với phía bên phải của bố cục (layout).

Khi đó, TextView sẽ nhảy sang bên trái vì không còn bị ràng buộc với bên phải nữa.

Để thêm lại ràng buộc ngang, nhấp vào cùng một hình tròn đó và kéo một đường nối đến phía bên phải của bố cục.

A screen shot of a computer

Description automatically generated

Extrackting string resourse:

Thay vì các String dạng code cứng, cách tốt nhất là sử dụng string dạng nguồn, cái mà đại diện cho 1 string, giống như 1 biến lưu giá trị. Có các string trong 1 file nguồn sẽ làm quản lý các string trỡ nên dễ dàng hơn, đặc biệt nếu bạn sử dụng những string đó nhiều lần. Cũng như, nguồn string là bắt buộc đối với việc dịch và bản địa hóa ứng dụng của bạn, bởivif bạn cần tạo các file chuỗi nguồn cho mỗi language.

1.Nhấp một lần vào từ **"Toast"** (cảnh báo đầu tiên được tô sáng).

2. Nhấn **Alt-Enter** trên Windows hoặc **Option-Enter** trên macOS và chọn **Extract string resource** từ menu bật lên.

3. Nhập **button\_label\_toast** cho phần **Resource name** (tên tài nguyên).

4. Nhấp **OK**. Một tài nguyên chuỗi sẽ được tạo trong tệp **values/res/strings.xml**, và chuỗi trong mã của bạn sẽ được thay thế bằng tham chiếu đến tài nguyên:

5. Tiếp tục trích xuất các chuỗi còn lại:

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

6. Trong Project > Android, mở rộng mục values trong res, sau đó nhấp đúp vào strings.xml để xem các tài nguyên chuỗi vừa được tạo trong tệp strings.xml.

**Giao diện:**

**A yellow and blue rectangular object with a number 0

Description automatically generated**

**Nguồn code xml:**

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout 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/btnToast"

android:layout\_width="406dp"

android:layout\_height="76dp"

android:background="#673AB7"

android:text="@string/btn\_toast"

android:textSize="24sp"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

android:onClick="showToast"/>

<TextView

android:id="@+id/show\_count"

android:layout\_width="406dp"

android:layout\_height="522dp"

android:layout\_marginTop="28dp"

android:background="#FFEB3B"

android:gravity="center"

android:text="@string/button\_label\_toast"

android:textColor="#F44336"

android:textSize="160sp"

android:textStyle="bold"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintHorizontal\_bias="0.4"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toBottomOf="@+id/btnToast" />

<Button

android:id="@+id/btnCount"

android:layout\_width="412dp"

android:layout\_height="69dp"

android:background="#3F51B5"

android:onClick="countUp"

android:text="@string/btn\_count"

android:textSize="24sp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

**Ngồn code Manifests:**

package com.example.hellotoast;

import android.os.Bundle;

import android.view.View;

import android.widget.TextView;

import android.widget.Toast;

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 {

private int trackCount=0;

private TextView showCount;

@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;

});

// anh xa

showCount = (TextView) findViewById(R.id.show\_count);

}

public void showToast(View view) {

String toast\_message = "";

Toast.makeText(this, showCount.getText(), Toast.LENGTH\_SHORT).show();

}

public void countUp(View view) {

trackCount++;

showCount.setText(Integer.toString(trackCount));

}

}

#### **Task 2: Coding Challenge**

Link github: [Android-unit1234/HelloToastChallenge at master · htpit1106/Android-unit1234](https://github.com/htpit1106/Android-unit1234/tree/master/HelloToastChallenge)

Ứng dụng HelloToast hiển thị tốt khi thiết bị hoặc trình giả lập ở chế độ dọc.

Tuy nhiên, nếu bạn chuyển thiết bị hoặc trình giả lập sang chế độ ngang, nút Count có thể chồng lên TextView ở phía dưới, như minh họa trong hình bên dưới.

A blue and yellow rectangular object with a number

Description automatically generated

Thử thách: Thay đổi bố cục để ứng dụng hiển thị đẹp ở cả chế độ dọc và ngang:

1.Trên máy tính của bạn, sao chép thư mục dự án HelloToast và đổi tên thành HelloToastChallenge.

2.Mở HelloToastChallenge trong Android Studio và thực hiện refactor (tái cấu trúc).

(Xem phần Appendix: Utilities để biết hướng dẫn về cách sao chép và tái cấu trúc một dự án).

3.Thay đổi bố cục sao cho nút Toast và Count xuất hiện ở phía bên trái, như hình minh họa bên dưới. TextView xuất hiện bên cạnh chúng nhưng chỉ rộng đủ để hiển thị nội dung. (Gợi ý: Sử dụng thuộc tính wrap\_content).

4.Chạy ứng dụng ở cả hai chế độ dọc và ngang để kiểm tra giao diện.

**Phương thức onCreate()**

Phương thức onCreate() được sử dụng để inflate (khởi tạo) bố cục, nghĩa là thiết lập nội dung hiển thị trên màn hình từ tệp XML của bố cục.

Bạn cũng có thể sử dụng nó để lấy tham chiếu đến các phần tử giao diện khác trong bố cục.

**View và findViewById()**

Một View (như một chuỗi ký tự) là một tài nguyên (resource) có thể có id.

Lệnh findViewById() nhận ID của một view làm tham số và trả về đối tượng View đó.

**Sử dụng Layout Editor:**

Design tab: Để chỉnh sửa các phần tử và bố cục bằng giao diện trực quan.

Text tab: Để chỉnh sửa mã XML của bố cục.

**Palettes pane:**

Hiển thị các phần tử giao diện (UI elements) mà bạn có thể sử dụng trong bố cục của ứng dụng.

***Component Tree pane:***

Hiển thị cây phân cấp của các phần tử giao diện trong bố cục.

**Design và Blueprint panes:**

Hiển thị trực quan các phần tử giao diện trong bố cục.

**Attributes tab:**

Hiển thị bảng thuộc tính (Attributes pane) để cài đặt các thuộc tính cho phần tử giao diện.

**Các thao tác với Constraints và Handles:**

Constraint handle:

Là các hình tròn xuất hiện ở bốn phía của một phần tử giao diện.

Nhấp vào constraint handle và kéo nó đến một constraint handle khác hoặc kéo đến parent boundary (ranh giới của bố cục cha) để tạo ràng buộc (constraint).

Ràng buộc này sẽ được biểu thị bằng một đường gấp khúc (zigzag line).

**Resizing handle:**

Là các hình vuông dùng để thay đổi kích thước của phần tử giao diện.

Khi kéo, tay cầm này sẽ chuyển thành góc xiên.

**Autoconnect** **tool**:

Khi được bật, công cụ này tự động tạo hai hoặc nhiều ràng buộc cho một phần tử giao diện với bố cục cha.

Sau khi bạn kéo phần tử vào bố cục, Autoconnect sẽ tạo ràng buộc dựa trên vị trí của phần tử đó.

**Clear Constraints:**

Để xóa tất cả ràng buộc trên một phần tử:

Chọn phần tử và di chuột qua nó để hiện nút Clear Constraints.

Nhấp vào nút này để xóa tất cả ràng buộc.

Để xóa một ràng buộc cụ thể, nhấp vào handle đã tạo ra ràng buộc đó.

**Attributes pane**

Cung cấp quyền truy cập vào tất cả các thuộc tính XML mà bạn có thể gán cho một phần tử giao diện.

Bao gồm một view inspector (bảng định cỡ hình vuông) ở trên cùng:

Các biểu tượng bên trong hình vuông đại diện cho cài đặt chiều cao và chiều rộng của phần tử giao diện

**Giao diện sau khi sửa từ task 1 ( trước đó không cân đối):**

A screenshot of a computer

Description automatically generated

**XML code:**

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout 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="horizontal"

tools:context=".MainActivity">

<LinearLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<Button

android:id="@+id/btnToast"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_margin="5dp"

android:background="#673AB7"

android:onClick="showToast"

android:text="@string/btn\_toast"

android:textSize="24sp" />

<Button

android:id="@+id/btnzero"

android:layout\_width="110dp"

android:layout\_height="wrap\_content"

android:layout\_margin="5dp"

android:background="#673AB7"

android:lineSpacingExtra="2sp"

android:onClick="EventZero"

android:text="Zero"

android:textSize="24sp" />

<Button

android:id="@+id/btnCount"

android:layout\_width="wrap\_content"

android:layout\_height="80dp"

android:layout\_margin="5dp"

android:background="#3F51B5"

android:onClick="countUp"

android:text="@string/btn\_count"

android:textSize="24sp" />

<ScrollView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical" />

</ScrollView>

</LinearLayout>

<TextView

android:id="@+id/show\_count"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_margin="5dp"

android:padding="@dimen/padding\_regular"

android:layout\_weight="6"

android:background="#FFEB3B"

android:gravity="center"

android:shadowRadius="@dimen/material\_emphasis\_disabled"

android:text="@string/button\_label\_toast"

android:textColor="#F44336"

android:textSize="160sp"

android:textStyle="bold" />

</LinearLayout>

**MainActivi.java:**

package com.example.helloconstraint;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.content.ContextCompat;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

private int trackCount=0;

private TextView showCount;

private Button btnZero;

@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;

});

// anh xa

showCount = (TextView) findViewById(R.id.show\_count);

btnZero = (Button) findViewById(R.id.btnzero);

}

public void showToast(View view) {

String toast\_message = "";

Toast.makeText(this, showCount.getText(), Toast.LENGTH\_SHORT).show();

}

public void countUp(View view) {

if(trackCount==0){

int Pink = ContextCompat.getColor(this, R.color.pink);

btnZero.setBackgroundColor(Pink);

}

trackCount++;

showCount.setText(Integer.toString(trackCount));

}

public void EventZero(View view) {

trackCount = 0;

showCount.setText(Integer.toString(trackCount));

int Gray = ContextCompat.getColor(this, R.color.Gray);

btnZero.setBackgroundColor(Gray);

}

}

Tóm tắt: Sử dụng Layout Editor để xem trước và tạo các biến thể bố cục

1.Xem trước bố cục ở chế độ ngang:

- Nhấp vào nút Orientation in Editor trên thanh công cụ phía trên và chọn Switch to Landscape để chuyển sang chế độ ngang.

- Chọn Switch to Portrait để quay lại chế độ dọc.

2.Tạo biến thể bố cục cho chế độ ngang:

- Nhấp vào Orientation in Editor và chọn Create Landscape Variation.

- Một cửa sổ chỉnh sửa mới sẽ mở ra với thẻ land/activity\_main.xml, hiển thị bố cục cho chế độ ngang.

3. Xem trước bố cục trên các thiết bị khác nhau:

- Nhấp vào nút Device in Editor trên thanh công cụ phía trên và chọn một thiết bị để xem trước mà không cần chạy ứng dụng trên thiết bị thật hoặc trình giả lập.

4. Tạo biến thể bố cục cho máy tính bảng (màn hình lớn hơn):

- Nhấp vào Orientation in Editor và chọn Create layout x-large Variation.

### Trình chỉnh sửa bố cục

**Sử dụng LinearLayout:**

LinearLayout là một ViewGroup sắp xếp tập hợp các View theo hàng ngang (horizontal) hoặc hàng dọc (vertical).

LinearLayout yêu cầu các thuộc tính sau:

* layout\_width: Độ rộng của LinearLayout.
* layout\_height: Chiều cao của LinearLayout.
* orientation: Định hướng của LinearLayout (ngang hoặc dọc).

Thuộc tính kích thước:

* match\_parent cho layout\_width hoặc layout\_height:
* Mở rộng View để lấp đầy toàn bộ View cha theo chiều rộng hoặc chiều cao.
* Nếu LinearLayout là View gốc (root View), nó sẽ mở rộng kích thước bằng với màn hình (tức là kích thước của View cha).

wrap\_content cho layout\_width hoặc layout\_height:

* Thu nhỏ kích thước của View sao cho vừa đủ bao bọc nội dung bên trong.
* Nếu không có nội dung, View sẽ trở nên vô hình.

Kích thước cố định bằng dp (density-independent pixels):

* Chỉ định kích thước cố định, được điều chỉnh theo mật độ màn hình của thiết bị.

Ví dụ: 16dp nghĩa là 16 density-independent pixels.

Thuộc tính orientation:

* horizontal: Sắp xếp các phần tử từ trái sang phải.
* vertical: Sắp xếp các phần tử từ trên xuống dưới.

#### Task 3: HomeWork

Link github: [Android-unit1234/HelloConstraint at master · htpit1106/Android-unit1234](https://github.com/htpit1106/Android-unit1234/tree/master/HelloConstraint)

**Giao diện:**

Screens screenshot of a phone

Description automatically generated

**Nguồn XML:**

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout 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="horizontal"

tools:context=".MainActivity">

<LinearLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<Button

android:id="@+id/btnToast"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_margin="5dp"

android:background="#673AB7"

android:onClick="showToast"

android:text="@string/btn\_toast"

android:textSize="24sp" />

<Button

android:id="@+id/btnzero"

android:layout\_width="110dp"

android:layout\_height="wrap\_content"

android:layout\_margin="5dp"

android:background="#673AB7"

android:lineSpacingExtra="2sp"

android:onClick="EventZero"

android:text="Zero"

android:textSize="24sp" />

<Button

android:id="@+id/btnCount"

android:layout\_width="wrap\_content"

android:layout\_height="80dp"

android:layout\_margin="5dp"

android:background="#3F51B5"

android:onClick="countUp"

android:text="@string/btn\_count"

android:textSize="24sp" />

<ScrollView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="vertical" />

</ScrollView>

</LinearLayout>

<TextView

android:id="@+id/show\_count"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_margin="5dp"

android:padding="@dimen/padding\_regular"

android:layout\_weight="6"

android:background="#FFEB3B"

android:gravity="center"

android:shadowRadius="@dimen/material\_emphasis\_disabled"

android:text="@string/button\_label\_toast"

android:textColor="#F44336"

android:textSize="160sp"

android:textStyle="bold" />

</LinearLayout>

**Nguồn code MainActivity:**

package com.example.helloconstraint;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import android.widget.Toast;

import androidx.activity.EdgeToEdge;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.content.ContextCompat;

import androidx.core.graphics.Insets;

import androidx.core.view.ViewCompat;

import androidx.core.view.WindowInsetsCompat;

public class MainActivity extends AppCompatActivity {

private int trackCount=0;

private TextView showCount;

private Button btnZero;

@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;

});

// anh xa

showCount = (TextView) findViewById(R.id.show\_count);

btnZero = (Button) findViewById(R.id.btnzero);

}

public void showToast(View view) {

String toast\_message = "";

Toast.makeText(this, showCount.getText(), Toast.LENGTH\_SHORT).show();

}

public void countUp(View view) {

if(trackCount==0){

int Pink = ContextCompat.getColor(this, R.color.pink);

btnZero.setBackgroundColor(Pink);

}

trackCount++;

showCount.setText(Integer.toString(trackCount));

}

public void EventZero(View view) {

trackCount = 0;

showCount.setText(Integer.toString(trackCount));

int Gray = ContextCompat.getColor(this, R.color.Gray);

btnZero.setBackgroundColor(Gray);

}

}  
**Giao diện khi chạy:**

A computer screen shot of a blue screen

Description automatically generated

### Văn bản và các chế độ cuộn

Link github: <https://github.com/htpit1106/Android-unit1234/tree/master> (MainActivty2)

A screenshot of a cell phone

Description automatically generated

Lớp TextView là 1 lớp con của class View, cái mà hiển thị văn bản trên màn hình. Bạn có thể điều khiển văn bản xuất hiện với thuộc tính TextView của file XML layout. Trong Thực hình lần này ta thấy cách để làm việc thao tác với đối tượng TextView, bao gồm 1 trong những cái mà người dùng có thể cuộn nội dung của nó theo chiều ngang

Nếu bạn có nhiều thông tin hơn cả khả năng màn hình thiết bị của bạn có thẻ hiển thị. Bạn cần tao 1 scrollingview, để người dùng có thể cuộn theo chiều dọc bằng cách kéo xuống học lên, hoặc theo chiều ngang bằng cách kéo sang phải hay trái

Bạn sẽ thường sử dụng scrolling view cho những câu chuyện, bài báo, hay bất kễ văn bản dài nào. Bạn có thể sử dụng nó để cho phép người dùng nhập nhiều dòng văn bản, hoặc để tích hợp trong giao diện(như 1 trường văn bản và 1 button) trong 1 scrollview

**ScrollView**

* **Scr**ollView là một lớp cung cấp bố cục cho giao diện cuộn (scrolling view).
* Nó là một subclass của FrameLayout.

**Đặc điểm và Cách sử dụng:**

* ScrollView chỉ chứa một phần tử con (child view).
* Phần tử con này chứa toàn bộ nội dung cần cuộn.
* Child view này có thể là một ViewGroup (ví dụ: LinearLayout) chứa các phần tử giao diện bên trong.

Với bố cục phức tạp chứa hình ảnh hoặc nhiều thành phần, có thể xảy ra vấn đề về hiệu suất.

* Một lựa chọn tốt là sử dụng LinearLayout với orientation = vertical bên trong ScrollView.
* Điều này giúp hiển thị các mục mà người dùng có thể cuộn qua (ví dụ: các TextView xếp dọc).

**Tổng quan về ứng dụng Scrolling Text:**

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

A screenshot of a computer

Description automatically generated

Bạn sẽ tạo tất cả các tha đổi trong code xml và trong strings.xml. Bạn sẽ edit code xml cho layout trên bản Text, ưhich mà bạn sẽ show lên màn hìn. Một vài thay đổi của đối tượng giao diện và thuộc tích sẽ rất dễ dàng khi thao tác trực tiếp trên file xml.

**Giao diện**

A screenshot of a text message

Description automatically generated

**Code file Strings.xml**

<resources>

<string name="app\_name">HelloToastChallenge</string>

<string name="button\_label\_toast">0</string>

<string name="btn\_count">Count</string>

<string name="textscroll">12. Mở rộng từng cảnh báo Hardcoded text, cuộn xuống cuối thông báo của cảnh báo và bấm vào nút Fix như bên dưới:

Việc sửa từng cảnh báo hardcoded text trích xuất string resources cho văn bản. Hộp thoại Extract Resource xuất hiện, và bạn có thể nhập tên cho string resources. Nhập tên string resources theo:

Chuỗi Nhập tên theo:

Donuts are glazed and sprinkled with candy.  donuts

Ice cream sandwiches have chocolate wafers and  vanilla filling.  ice\_cream\_sandwiches

FroYo is premium selfserve frozen yogurt. froyo

Bố cục bây giờ sẽ trông giống như hình bên dưới:

1.3 Thêm mô tả văn bản

Trong bước này bạn thêm một văn bản mô tả (TextView) cho mỗi món tráng miệng. Vì bạn đã trích string resources cho trường contentDescription cho mỗi phần tử ImageView, bạn có thể sử dụng cùng các string resources này cho mỗi TextView mô tả.

1. Kéo một phần tử TextView vào bố cục.

2. Ràng buộc bên trái của phần tử này vào bên phải của ImageView donut và ràng buộc bên trên của nó vào cạnh trên của ImageView donut, cả hai với khoảng cách 24 (24dp).

3. Ràng buộc cạnh phải của phần rử này vào cạnh phải của bố cục và cũng sử dụng khoảng cách 24 (24dp). Nhập donut\_description cho trường ID trong bảng Attributes. TextView mới sẽ xuất hiện bên cạnh hình ảnh donut như hình bên dưới.

4. Trong bảng Attributes thay đổi chiều rộng trong inspector pane thành Match Constraints:

5. Trong bảng Attributes, bắt đầu nhập string resources cho trường text bằng cách đặt ký hiệu @ trước: @d. Bấm vào tên string resources (@string/donuts) khi nó xuất hiện gợi ý:

6. Lặp lại các bước trên để thêm một TextView thứ hai, được ràng buộc vào cạnh phải và bên trên của ImageView ice\_cream, và cạnh phải của bố cục. Nhập vào bảng Attributes:

Trường Attribute Nhập theo sau:

ID ice\_cream\_descriptiom

Left, right, and top margins 24

layout\_width match\_constraint

text @string/ice\_cream\_sandwiches

7. Lặp lại các bước trên để thêm một TextView thứ ba, được ràng buộc vào cạnh phải và cạnh trên của ImageView froyo, và cạnh phải của nó vào cạnh phải của bố cục. Nhập vào bảng Attributes:

Trường Attribute Nhập theo sau:

ID froyo\_descriptiom

Left, right, and top margins 24

layout\_width match\_constraint

text @string/froyo

Bố cục bây giờ sẽ trông giống như hình bên dưới:

Mã giải pháp nhiệm vụ 1

Bố cục XML cho tệp content.xml được hiển thị bên dưới

Nhiệm vụ 2: Thêm phương thức onClick cho hình ảnh

Để làm cho một View có thể bấm được để người dùng có thể chạm (hoặc bấm) vào nó, hãy thêm thuộc tính android:onClick trong bố cục XML và chỉ định phương thức xử lý sự kiện click. Ví dụ, banh có thể làm một ImageView hoạt động như một Button bằng cách thêm android:onClick vào ImageView. Trong nhiệm vụ này, bạn sẽ làm cho các hình ảnh trong bố cục của bạn có thể nhấn được

2.1 Tạo phương thức Toast

Trong nhiệm vụ này, bạn thêm từng phương thức cho thuộc tính android:onClick để gọi khi mỗi hình ảnh được nhấn vào. Trong nhiệm vụ này, các phương thức đơn giản hiển thị một thông báo Toast, nó cho biết hình ảnh đã được chạm vào. (Trong một chương khác, bạn sẽ sửa đổi các phương thức này để khởi động một Activity khác.)

1. Để sử dụng string resources trong code Java, bạn nên thêm chúng vào tệp strings.xml. Mở rộng res > values trong bảng điều khiển Project > Android, và mở string.xml. Thêm các string resources sau cho các chuỗi để hiển thị trong thông báo Toast:

2. Mở MainActivity, và thêm phương thức displayToast() sau vào cuối MainActivity (trước dấu ngoặc đóng):

Mặc dù bạn có thể thêm phương thức này ở bất cứ vị trí nào trong MainActivity, nhưng thực tiễn tốt nhất là đặt các phương thức bên dưới các phương thức đã được cung cấp trong MainActivity bởi mẫu.

2.2 Tạo trình xử lý sự kiện click

Mỗi hình ảnh có thể nhấp cần một trình xử lý sự kiện nhấp chuột—một phương thức cho thuộc tính android:onClick để gọi. Trình xử lý sự kiện nhấp chuột, nếu được gọi từ thuộc tính android:onClick, phải là public, trả về void, và định nghĩa một View như là tham số duy nhất của nó. Thực hiện các bước sau để thêm các trình xử lý sự kiện nhấp chuột:

1. Thêm phương thức showDonutOrder() sau vào MainActivity. Trong nhiệm vụ này, sử dụng phương thức displayToast() đã tạo trước đó để hiển thị một thông báo Toast.

Ba dòng đầu tiên là một nhận xét theo định dạng Javadoc, nó làm code dễ hiểu hơn và cũng giúp tạo tài liệu cho code của bạn. Đây là một thực tiễn tốt nhất để thêm nhận xét như vậy cho mỗi phương thức mới mà bạn tạo. Để biết thêm thông tin về cách viết nhận xét, hãy xem How to Write Doc Comments for the Javadoc Tool.

2. Thêm nhiều phương thức vào cuối MainActivity cho mỗi món tráng miệng:

3. (Tùy chọn) Chọn Code > Reformat Code để định dạng lại code mà bạn đã thêm vào MainActivity để phù hợp với các tiêu chuẩn và làm cho nó dễ đọc hơn.

2.3 Thêm thuộc tính onClick

Trong bước này bạn thêm android:onClick vào từng phần tử ImageView trong bố cục. Thuộc tính android:onClick gọi trình xử lý sự kiện nhấn cho từng phần tử.

1. Mở tệp content\_main.xml, và nhấp vào tab Text trong trình chỉnh sửa bố cục để hiển thị mã XML.

2. Thêm thuộc tính android:onClick vào ImageView donut. Khi bạn nhập, các gợi ý sẽ xuất hiện cho các trình xử lý nhấn. Chọn trình xử lý nhấn showDonutOrder. Code bây giờ trông như sau:

Dòng cuối cùng (android:onClick="showDonutOrder") gán trình xử lý nhấp chuột (showDonutOrder) cho ImageView.

3. (Tùy chọn) Chọn Code > Reformat Code để định dạng lại mã XML mà bạn đã thêm vào content\_main.xml để phù hợp với các tiêu chuẩn và làm cho nó dễ đọc hơn. Android Studio sẽ tự động di chuyển thuộc tính android:onClick lên vài dòng để kết hợp chúng với các thuộc tính khác bắt đầu bằng android: như lời nói đầu.

4. Thực hiện theo cùng một quy trình để thêm thuộc tính android:onClick vào các phần tử ImageView ice\_cream và froyo. Chọn các trình xử lý nhấn showDonutOrder và show FroyoOrder. Bạn có thể tùy chỉnh chọn Code > Reformat Code để định dạng lại code XML. Code bây giờ trông như sau:

⦁ Các điều khiển nhập liệu

⦁ Menu và bộ chọn

⦁ Điều hướng người dùng

⦁ RecycleView

Những gì Bạn nên biết

Bạn nên có khả năng:

⦁ Tạo một dự án Android Studio từ một mẫu và tạo bố cục chính.

⦁ Chạy ứng dụng trên mô hình giả lập hoặc thiết bị được kết nối.

⦁ Tạo và chỉnh sửa phần tử UI bằng trình chỉnh sửa bố cục và mã XML.

⦁ Truy cập các phần tử UI từ mã của bạn sử dụng findViewById().

⦁ Xử lý thao tác bấm vào Button.

⦁ Hiển thị thông báo Toast.

⦁ Thêm những hình ảnh vào thư mục drawable của dự án.

Những gì Bạn sẽ cần:

⦁ Cách sử dụng hình ảnh như một phần tử tương tác để thực hiện hành động.

⦁ Cách thiết lập các thuộc tính cho phần tử ImageView trong trình chỉnh sửa bố cục.

⦁ Cách thêm phương thức onClick() để hiển thị thông báo Toast.

Những gì bạn sẽ làm

⦁ Tạo một dự án Android Studio mới cho một ứng dụng giả lập đặt món tráng miệng sử dụng hình ảnh làm phần tử tương tác.

⦁ Thiết lập xử lý onClick() cho các hình ảnh để hiển thị ra các thông báo Toast khác nhau.

⦁ Thay đổi nút hành động nổi do mẫu cung cấp để hiển thị biểu tượng khác nhau và khởi chạy một Activity khác.

Tổng quan ứng dụng

Trong bài thực hành này, bạn tạo và xây dựng một ứng dụng mới bắt đầu bằng mẫu Basic Activity, mô phỏng một ứng dụng đặt món tráng miệng. Người dùng có thể nhấn vào hình ảnh để thực hiện một hành động—trong trường hợp này là hiển thị một thông báo Toast—như minh họa trong hình bên dưới. Người dùng cũng có thể nhấn vào nút giỏ hàng để tiến tới Activity tiếp theo.

Nhiệm vụ 1: Thêm hình ảnh vào bố cục

Bạn có thể làm một chế độ xem có thể nhấn được, như một nút, bằng cách thêm thuộc tính android:onClick trong bố cục XML. Ví dụ, bạn có thể làm cho ảnh hoạt động giống như một nút bằng cách thêm android:onClick vào ImageView.

Trong nhiệm vụ này bạn tạo một nguyên mẫu của ứng dụng đặt món tráng miệng từ một quán cà phê. Sau khi bắt đầu một dự án mới dựa trên mẫu Basic Activity, bạn sửa đổi TextView “Hello world” thành nội dung phù hợp và thêm hình ảnh mà người dùng có thể nhấn vào.

1.1 Bắt đầu dự án mới

1. Bắt đầu một dự án Android Studio mới với tên ứng dụng là Droid Cafe.

2. Chọn mẫu Basic Activity, và chấp nhận tên Activity mặc định (MainActivity). Đảm bảo các tùy chọn Generate Layout file và Backwards Compatibility (AppCompat) được chọn.

3. Bấm Finish.

Dự án mở ra với 2 bố cục trong trong thư mục res > layout: activity\_main.xml dành cho thanh ứng dụng và nút hành động nổi (Bạn không thay đổi trong nhiệm vụ này), và content\_main.xml dành cho mọi thứ khác trong giao diện</string>

</resources>

**File Layout .xml**

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout 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=".MainActivity2">

<TextView

android:id="@+id/textView2"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:background="#03A9F4"

android:text="Báo Mới ngày 25/2"

android:textColor="#FAF3F3"

android:textSize="24sp"

android:textStyle="bold" />

<ScrollView

android:layout\_width="391dp"

android:layout\_height="849dp">

<TextView

android:id="@+id/textView"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@string/textscroll" />

</ScrollView>

</LinearLayout>

**File MainActivity2**

package com.example.hellotoastchallenge;

import android.os.Bundle;

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 MainActivity2 extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

EdgeToEdge.enable(this);

setContentView(R.layout.activity\_main2);

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;

});

}

}

### Tài nguyên có sẵn

## Activities

### Activity và Intent

### Vòng đời của Activity và trạng thái

### Intent ngầm định

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

### Trình gỡ lỗi

### Kiểm thử đơn vị

### Thư viện hỗ trợ

# TRẢI NGHIỆM NGƯỜI DÙNG

## Tương tác người dùng

### Hình ảnh có thể chọn

### Các điều khiển nhập liệu

### Menu và bộ chọn

### Điều hướng người dùng

### RecycleView

## Trải nghiệm người dùng thú vị

### Hình vẽ, định kiểu và chủ đề

### Thẻ và màu sắc

### Bố cục thích ứng

## Kiểm thử giao diện người dùng

### Espresso cho việc kiểm tra UI

# LÀM VIỆC TRONG NỀN

## Các tác vụ nền

### AsyncTask

### AsyncTask và AsyncTaskLoader

### Broadcast receivers

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

### Thông báo

### Trình quản lý cảnh báo

### JobScheduler

# LƯU DỮ LIỆU NGƯỜI DÙNG

## Tùy chọn và cài đặt

### Shared preferences

### Cài đặt ứng dụng

## Lưu trữ dữ liệu với Room

### Room, LiveData và ViewModel

### Room, LiveData và ViewModel