TRƯỜNG ĐẠI HỌC THỦY LỢI

KHOA CÔNG NGHỆ THÔNG TIN

GIÁO TRÌNH

THỰC HÀNH PHÁT TRIỂN ỨNG DỤNG CHO THIẾT BỊ DI ĐỘNG

Hà Nội, 2.2025

MỤC LỤC

CHƯƠNG 1. Làm quen 3
Bài 1) Tạo ứng dụng đầu tiên 3
1.1) Android Studio và Hello World 3
1.2) Giao diện người dùng tương tác đầu tiên 5
1.3) Trình chỉnh sửa bố cục 9
1.4) Văn bản và các chế độ cuộn 9
1.5) Tài nguyên có sẵn 9
Bài 2) Activities 9
2.1) Activity và Intent 9
2.2) Vòng đời của Activity và trạng thái 9
2.3) Intent ngầm định 9
Bài 3) Kiểm thử, gỡ lỗi và sử dụng thư viện hỗ trợ 9
3.1) Trình gỡ lỗi 9
3.2) Kiểm thử đơn vị 9
3.3) Thư viện hỗ trợ 9
CHƯƠNG 2. Trải nghiệm người dùng 10
Bài 1) Tương tác người dùng 10
1.1) Hình ảnh có thể chọn 10
1.2) Các điều khiển nhập liệu 10
1.3) Menu và bộ chọn 10
1.4) Điều hướng người dùng 10
1.5) RecycleView 10
Bài 2) Trải nghiệm người dùng thú vị 10
2.1) Hình vẽ, định kiểu và chủ đề 10
2.2) Thẻ và màu sắc 10
2.3) Bố cục thích ứng 10
Bài 3) Kiểm thử giao diện người dùng 10

3.1) Espresso cho việc kiểm tra UI 10
CHƯƠNG 3. Làm việc trong nền..... 10
Bài 1) Các tác vụ nền..... 10
1.1) AsyncTask..... 10
1.2) AsyncTask và AsyncTaskLoader 19
1.3) Broadcast receivers 19
Bài 2) Kích hoạt, lập lịch và tối ưu hóa nhiệm vụ nền..... 19
2.1) Thông báo..... 19
2.2) Trình quản lý cảnh báo 19
2.3) JobScheduler..... 19
CHƯƠNG 4. Lưu dữ liệu người dùng 19
Bài 1) Tùy chọn và cài đặt..... 19
1.1) Shared preferences 19
1.2) Cài đặt ứng dụng 19
Bài 2) Lưu trữ dữ liệu với Room 19
2.1) Room, LiveData và ViewModel..... 19
2.2) Room, LiveData và ViewModel..... 19

3.1) Trinfh gowx loi ……...

CHƯƠNG 1. LÀM QUEN

Bài 1) Tạo ứng dụng đầu tiên

1.1) Android Studio và Hello World

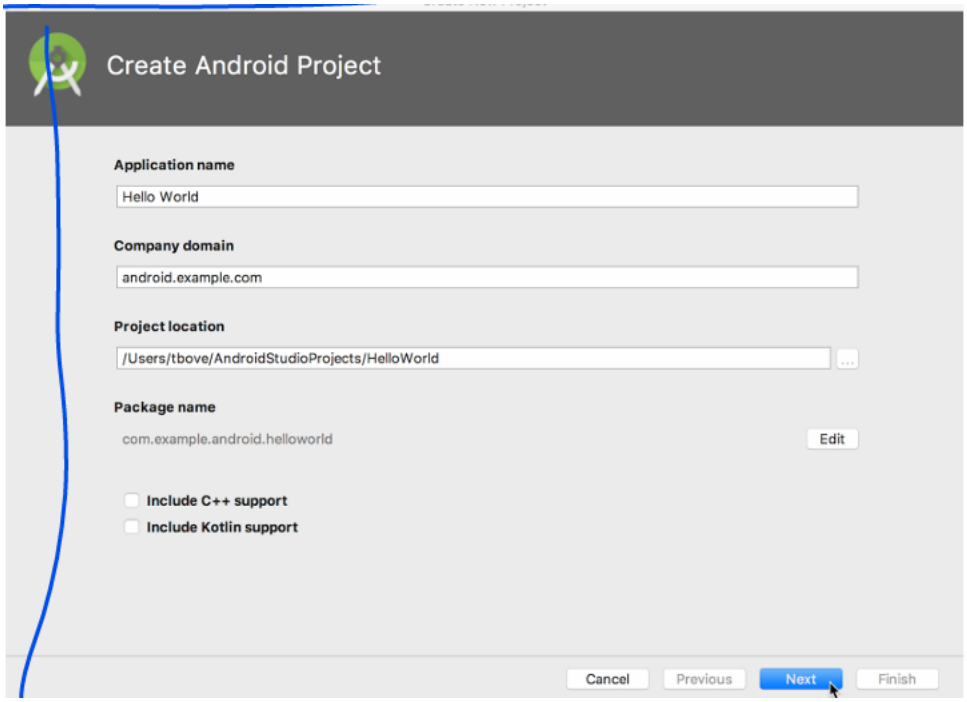
Giới thiệu

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

Những gì Bạn nên biết

Bạn nên có khả năng:

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



Những gì Bạn sẽ cần:

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

Những gì bạn sẽ học

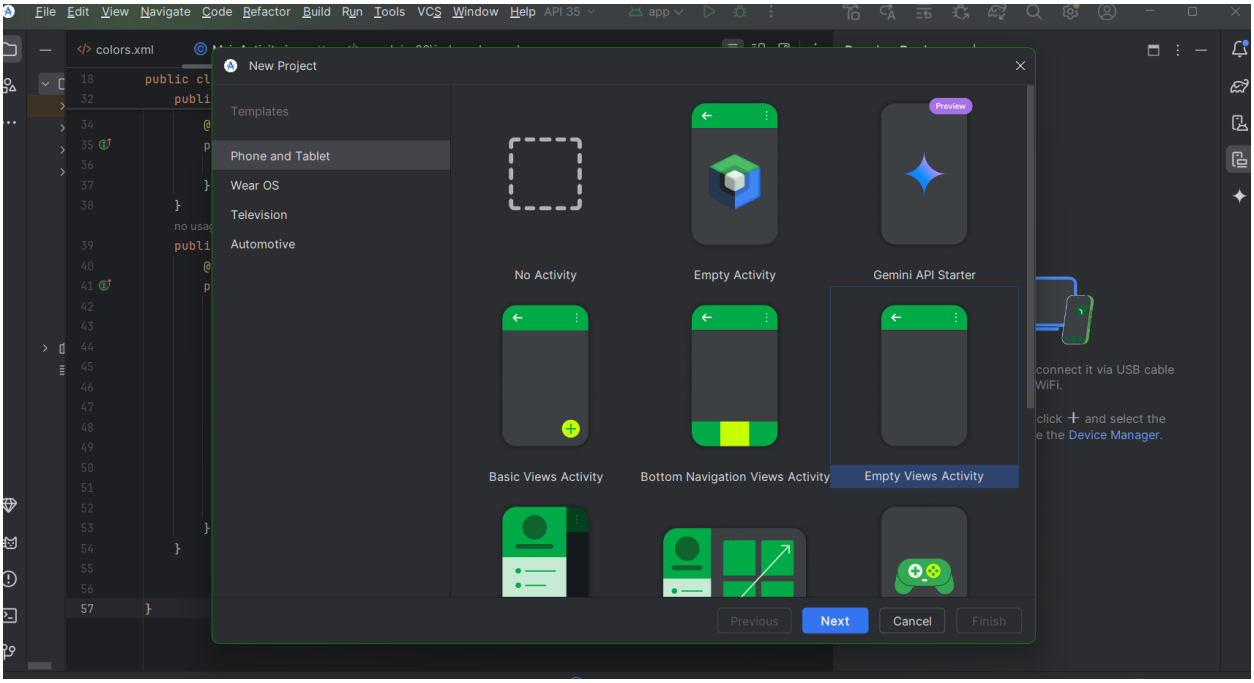
- Cách cài đặt và sử dụng IDE Android Studio.
- Cách sử dụng quy trình phát triển để xây dựng ứng dụng Android.
- Cách tạo một dự án Android từ một mẫu.

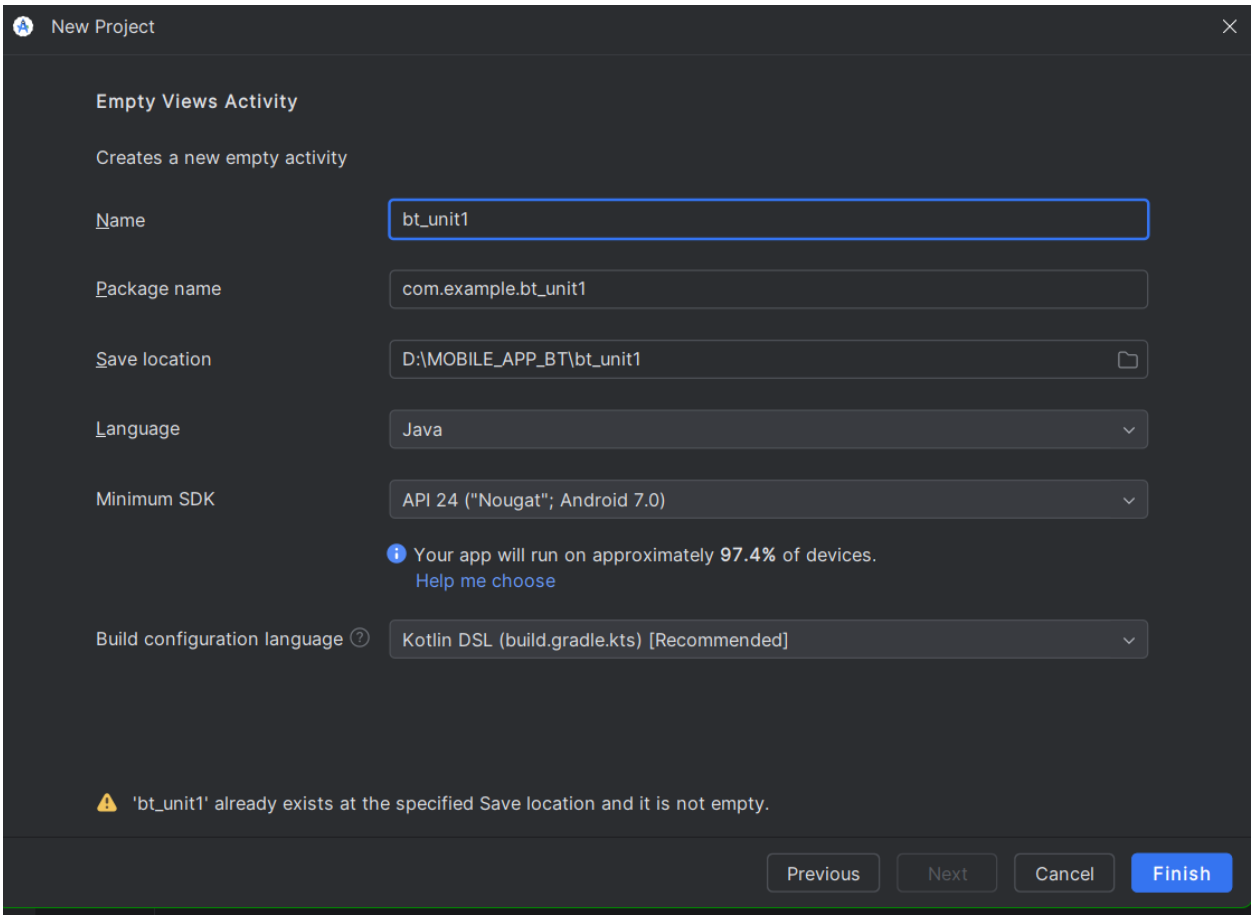
- Cách thêm thông điệp ghi lại vào ứng dụng của bạn để phục vụ mục đích gỡ lỗi.

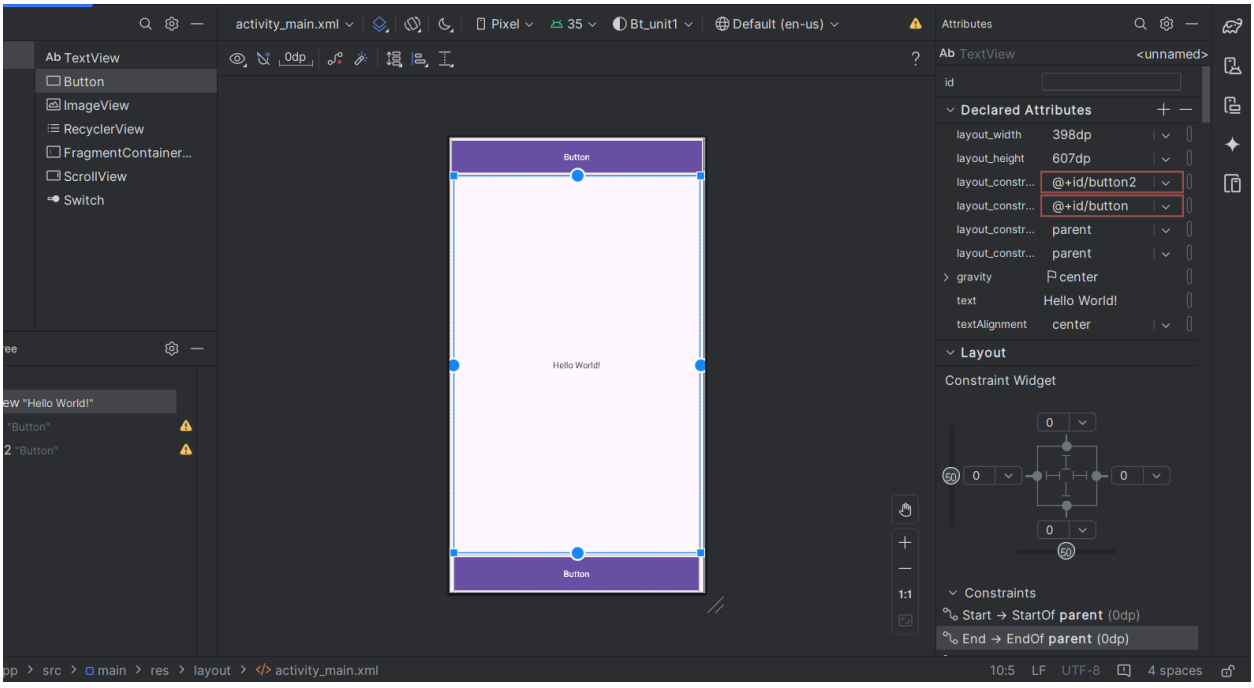
Những gì bạn sẽ làm

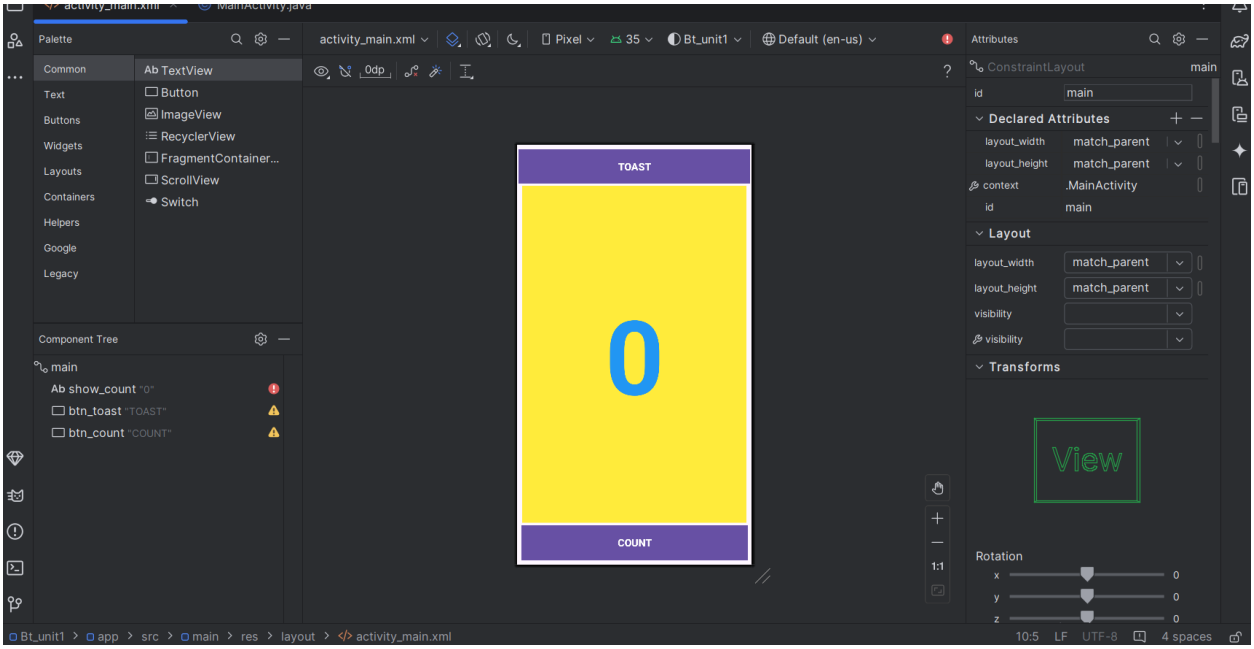
- Cài đặt môi trường phát triển **Android Studio**.
- Tạo một trình giả lập (thiết bị ảo) để chạy ứng dụng của bạn trên máy tính.
- Tạo và chạy ứng dụng **Hello World** trên các thiết bị ảo và vật lý.
- Khám phá cấu trúc dự án.
- Tạo và xem các thông điệp ghi lại từ ứng dụng của bạn.
- Khám phá tệp **AndroidManifest.xml**

1.2) Giao diện người dùng tương tác đầu tiên




New Project
Empty Views Activity
Creates a new empty activity
Name
bt_unit1
Package name
com.example.bt_unit1
Save location
D:\MOBILE_APP_BT\bt_unit1
Language
Java
Minimum SDK
API 24 ("Nougat"; Android 7.0)
Your app will run on approximately 97.4% of devices.
Help me choose
Build configuration language
Kotlin DSL (build.gradle.kts) [Recommended]
'bt_unit1' already exists at the specified Save location and it is not empty.
Previous
Next
Cancel
Finish


activity_main.xml
Pixel
35
Bt_unit1
Default (en-us)
TextView
Button
ImageView
RecyclerView
FragmentContainer...
ScrollView
Switch
Button
Hello World!
Button
Attributes
TextView
<unnamed>
id
Declared Attributes
layout_width 398dp
layout_height 607dp
layout_constr... @+id/button2
layout_constr... @+id/button
layout_constr... parent
layout_constr... parent
gravity center
text Hello World!
textAlignment center
Layout
Constraint Widget
Constraints
Start → StartOf parent (0dp)
End → EndOf parent (0dp)
app > src > main > res > layout > activity_main.xml
10:5 LF UTF-8 4 spaces

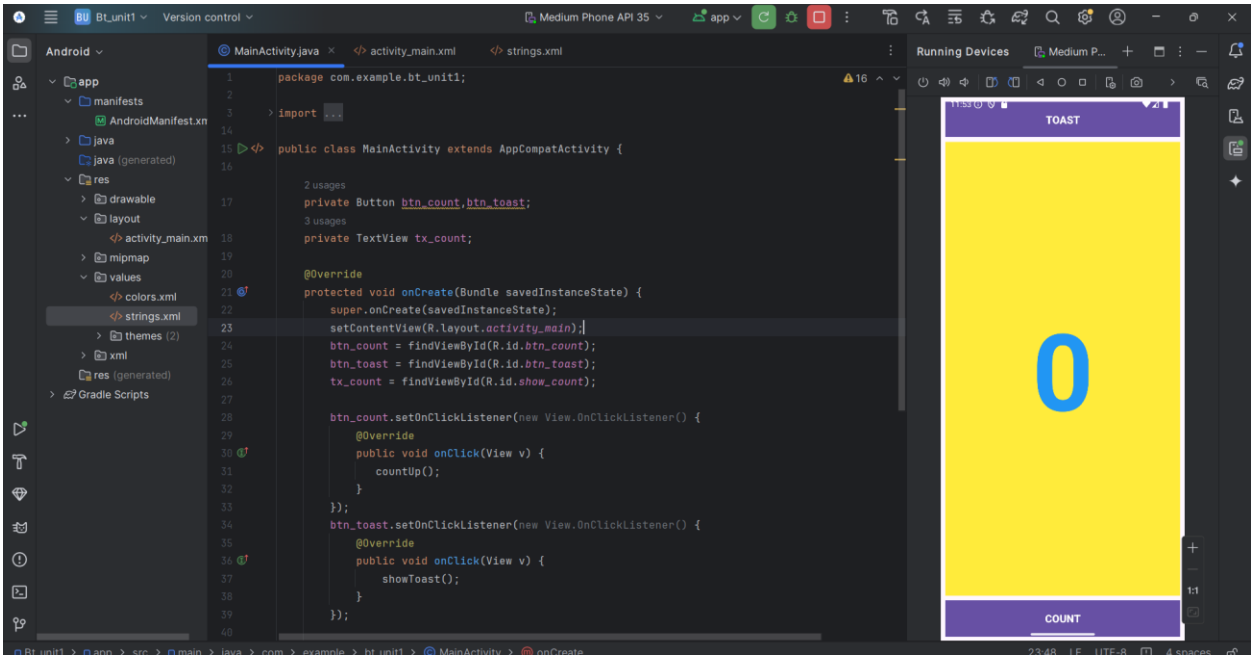


activity_main.xml | MainActivity.java

```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"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/show_count"
        android:layout_width="393dp"
        android:layout_height="587dp"
        android:background="#FFEB3B"
        android:gravity="center"
        android:text="0"
        android:textAlignment="center"
        android:textColor="#2196F3"
        android:textSize="180dp"
        android:textStyle="bold"
        app:layout_constraintBottom_toTopOf="@+id/btn_count"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btn_toast" />
```

```
    <Button
        android:id="@+id/btn_toast"
        android:layout_width="406dp"
        android:layout_height="59dp"
        android:layout_margin="8dp"
        android:background="#990C0C"
        android:text="TOAST"
        android:textSize="18dp"
        android:textStyle="bold"
        app:circularflow_radiusInDP="0"
        app:cornerRadius="0dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
```

```
40
41      <Button
42          android:id="@+id/btn_count"
43          android:layout_width="396dp"
44          android:layout_height="61dp"
45          android:layout_margin="8dp"
46          android:background="#B61243"
47          android:backgroundTint="#E9D6D6"
48          android:text="COUNT"
49          android:textSize="18dp"
50          android:textStyle="bold"
51          app:cornerRadius="0dp"
52          app:layout_constraintBottom_toBottomOf="parent"
53          app:layout_constraintEnd_toEndOf="parent"
54          app:layout_constraintHorizontal_bias="0.466"
55          app:layout_constraintStart_toStartOf="parent" />
56
57  </androidx.constraintlayout.widget.ConstraintLayout>
```



1.3) Trình chỉnh sửa bố cục

1.4) Văn bản và các chế độ cuộn

1.5) Tài nguyên có sẵn

Bài 2) Activities

2.1) Activity và Intent

2.2) Vòng đời của Activity và trạng thái

2.3) Intent ngầm định

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

3.1) Trình gỡ lỗi

3.2) Kiểm thử đơn vị

3.3) Thư viện hỗ trợ

CHƯƠNG 2. TRẢI NGHIỆM NGƯỜI DÙNG

Bài 1) Tương tác người dùng

1.1) Hình ảnh có thể chọn

1.2) Các điều khiển nhập liệu

1.3) Menu và bộ chọn

1.4) Điều hướng người dùng

1.5) RecycleView

Bài 2) Trải nghiệm người dùng thú vị

2.1) Hình vẽ, định kiểu và chủ đề

2.2) Thẻ và màu sắc

2.3) Bố cục thích ứng

Bài 3) Kiểm thử giao diện người dùng

3.1) Espresso cho việc kiểm tra UI

CHƯƠNG 3. LÀM VIỆC TRONG NỀN

Bài 1) Các tác vụ nền

1.1) AsyncTask

Giới thiệu

Luồng là một đường dẫn thực thi độc lập trong một chương trình đang chạy. Khi một chương trình Android được khởi chạy, hệ thống sẽ tạo một luồng chính, còn được gọi là luồng UI. Luồng UI này là cách ứng dụng của bạn tương tác với các thành phần từ bộ công cụ Android UI.

Đôi khi, một ứng dụng cần thực hiện các tác vụ tốn nhiều tài nguyên như tải xuống tệp, thực hiện truy vấn cơ sở dữ liệu, phát phương tiện hoặc tính toán phân tích phức tạp. Loại công việc tốn nhiều tài nguyên này có thể chặn luồng UI khiến ứng dụng không phản hồi đầu vào của người dùng hoặc

không vẽ trên màn hình. Người dùng có thể cảm thấy bực bội và gỡ cài đặt ứng dụng của bạn.

Để duy trì trải nghiệm người dùng (UX) chạy trơn tru, khuôn khổ Android cung cấp một lớp trợ giúp có tên là AsyncTask, xử lý công việc ngoài luồng UI. Sử dụng AsyncTask để chuyển xử lý chuyên sâu sang một luồng riêng biệt có nghĩa là luồng UI có thể phản hồi.

Vì luồng riêng biệt không được đồng bộ hóa với luồng gọi, nên nó được gọi là luồng không đồng bộ. AsyncTask cũng chứa lệnh gọi lại cho phép bạn hiển thị kết quả tính toán trở lại trong luồng UI.

Trong phần thực hành này, bạn sẽ học cách thêm tác vụ nền vào ứng dụng Android bằng AsyncTask.

Những điều bạn nên biết:

Bạn có thể :

- Tạo một Activity.
- Thêm một TextView vào bố cục của Activity.
- Chương trình lấy id của TextView và thiết lập nội dung của nó.
- Sử sụng chế độ xem Button và chức nang onClick của chúng.

Bạn sẽ học được gì

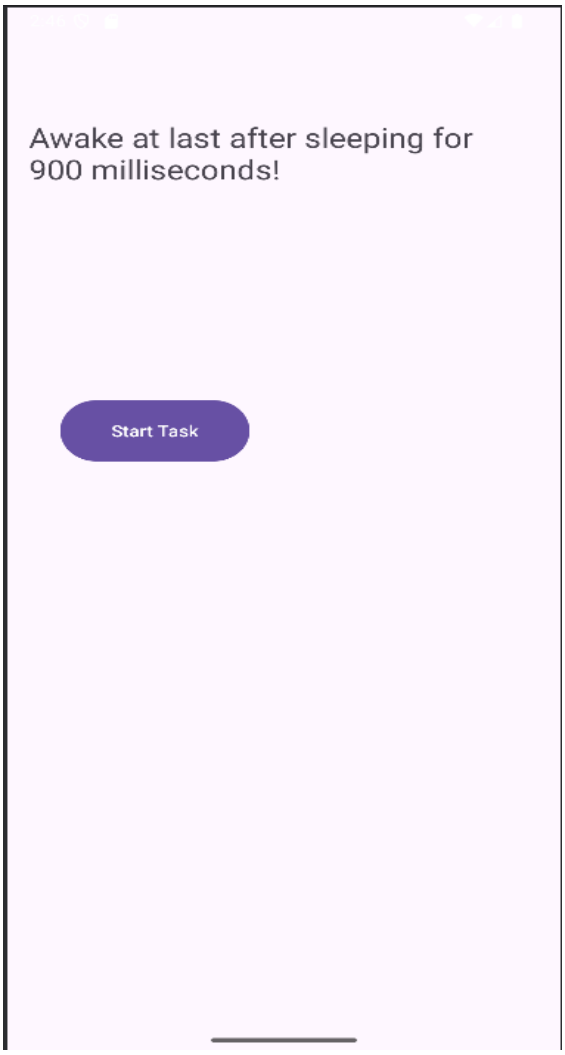
- Cách thêm AsyncTask vào ứng dụng của bạn để chạy tác vụ ở chế độ nền của ứng dụng.
- Những hạn chế của việc sử dụng AsyncTask cho các tác vụ nền.

Bạn sẽ làm những gì

- Tạo một ứng dụng đơn giản để thực hiện một tác vụ nền sử dụng AsyncTask.
- Chạy ứng dụng và xem điều gì xảy ra khi bạn xoay thiết bị.
- Thực hiện các trangj thái thể hiện Activity để duy trì trạng thái của thông báo TextView

Tổng quan ứng dụng

Bạn sẽ xây dựng một ứng dụng có một TextView và một Button. Khi người dùng click vào Button, ứng dụng ngủ trong một khoảng thời gian ngẫu nhiên, và sau đó hiển thị một thong báo trong TextView khi nó hoạt động trở lại. Ứng dụng khi hòa thành sẽ trông như thế này :



Nhiệm vụ 1 : Cài đặt dự án SimpleAsyncTask

Giao diện người dùng SimpleAsyncTask chứa một Button khởi chạy AsyncTask, và một TextView hiển thị trạng thái của ứng dụng.

1.1 Tạo dự án và bố cục

Bước 1 : Tạo một dự án gọi **SimpleAsyncTask** sử dụng mẫu **Emty Views Activity**. Chấp nhận mặc định cho tất cả các lựa chọn khác.

Bước 2 : Mở file bố cục **activity_main.xml .** Nhấn vào tab **Text.**

Bước 3 : Thêm thuộc tính **layout_margin** vào **ConstraintLayout** cấp cao nhất:

```
android:layout_margin="16dp"
```

Bước 4 : Thêm hoặc sửa đổi các thuộc tính sau của TextView "Hello World!" để có các giá trị này. Trích xuất chuỗi thành một tài nguyên

```
    android:id="@+id/textView1"
    android:text="I am ready to start work!"
    android:textSize="24sp"
```

Bước 5 : Xóa thuộc tính **app : layout_constraintRight_toRightOf** và

app : layout_constraintTop_toTopOf .

Bước 6 : Thêm một phần tử **Button** ngay bên dưới **TextView** và cung cấp cho nó các thuộc tính này. Trích xuất văn bản **Button** thành một tài nguyên chuỗi.

```
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Start Task"
    android:layout_marginTop="24dp"
    android:onClick="startTask"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView1"
```

Bước 7: Thuộc tính **onClick** cho **Button** sẽ được làm nổi bật bằng màu đỏ, bởi vì phương thực **startTask()** chưa được thực hiện trong **MainActivity**. Đưa con trỏ tới dòng nổi bật đó, nhấn **Alt + Enter** (**Option + Enter** với Mac) và chọn **Creat startTask(View) in 'MainActivity'** để tạo phương thức trong **MainActivity.**

Mẫu code cho **activity_main.xml:**

```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:layout_margin="16dp"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="I am ready to start work!"
        android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.2" />
```

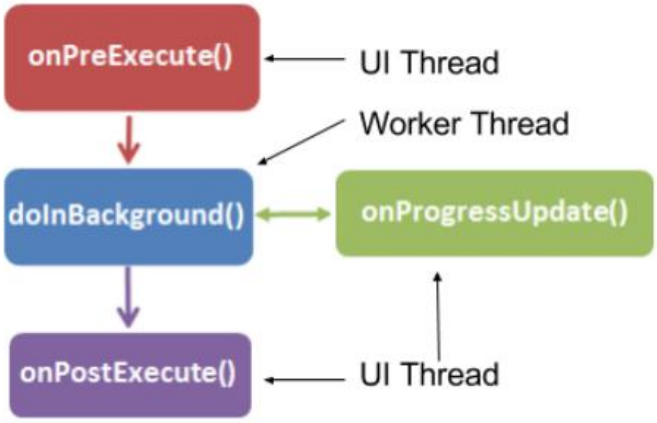
```xml
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start Task"
        android:layout_marginTop="24dp"
        android:onClick="startTask"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView1"

        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent" />


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

Nhiệm vụ 2: Tạo lớp con AsyncTask

AsyncTask là một lớp trừu tượng, nghĩa là bạn phải phân lớp nó để sử dụng. Trong ví dụ này, AsyncTask thực hiện một tác vụ nền rất đơn giản: nó ngủ trong một khoảng thời gian ngẫu nhiên. Trong một ứng dụng thực tế, tác vụ nền có thể thực hiện đủ loại công việc, từ truy vấn cơ sở dữ liệu, kết nối internet đến tính toán nước đi tiếp theo để đánh bại nhà vô địch cờ vây hiện tại.

Lớp con AsyncTask có các phương thức sau để thực hiện công việc ngoài luồng chính:

- **onPreExecute():** Phương thức này chạy trên luồng UI và được sử dụng để thiết lập tác vụ của bạn (như hiển thị thanh tiến trình).
- **doInBackground():** Đây là nơi bạn triển khai mã để thực hiện công việc cần thực hiện trên luồng riêng biệt.
- **onProgressUpdate():** Được gọi trên luồng UI và được sử dụng để cập nhật tiến trình trong UI (chẳng hạn như điền vào thanh tiến trình)
- **onPostExecute():** Một lần nữa trên luồng UI, phương thức này được sử dụng để cập nhật kết quả vào UI sau khi **AsyncTask** tải xong.



Ghi chú : Luồng của nền hoặc luồng công việc là bất cứ luồng nào không là luồng chính hoặc luồng UI

Khi bạn tạo một lớp con **AsyncTask**, bạn có lẽ cần cấp cho nó thông tin về công việc mà nó sẽ thực hiện. Liệu có nên báo cáo tiến trình của nó không và báo cáo kết quả theo hình thức nào.

Khi bạn tạo một lớp con **AsyncTask**, bạn có thể cấu hình nó sử dụng các tham số sau:

- **Params**: Kiểu dữ liệu của các tham số được gửi đến tác vụ khi thực hiện phương thức ghi đè **doInBackground()**
- **Progress:** Kiểu dữ liệu của các đơn vị tiến trình được công bố bằng phương thức ghi đè **onProgressUpdated()**
- **Result:** Kiểu dữ liệu của kết quả được cung cấp bởi phương thức ghi đè **onPostExecute()**

Ví dụ, một lớp con **AsyncTask** có tên là **MyAsyncTask** với khai báo sau có thể sử dụng các tham số sau:

- Chuỗi là tham số trong **doInBackground**() , dùng để truy vấn
- Một số nguyên cho **onProgressUpdate**(), để biểu thị phần trăm công việc hoàn thành.
- Một **Bitmap** cho kết quả trong **onPostExecute**(), chỉ ra kết quả truy vấn.

```
public class MyAsyncTask
        extends AsyncTask <String,Integer, Bitmap>{
```

Trong tác vụ này, bạn sẽ sử dụng lớp con AsyncTask để xác định công việc sẽ chạy trong một luồng khác với luồng UI.

2.1 Lớp con AsyncTask

Trong ứng dụng này, lớp con AsyncTask mà bạn tạo không yêu cầu tham số truy vấn hoặc công bố tiến trình của nó. Bạn sẽ chỉ sử dụng các phương thức **doInBackground**() và **onPostExecute**().

1. Tạo một lớp Java mới có tên là **SimpleAsyncTask** mở rộng **AsyncTask** và sử dụng ba tham số kiểu chung.

   Sử dụng **Void** cho các tham số, vì **AsyncTask** này không yêu cầu bất kỳ đầu vào nào. Sử dụng **Void** cho loại tiến trình, vì tiến trình không được công bố. Sử dụng **String** làm loại kết quả, vì bạn sẽ cập nhật **TextView** bằng một chuỗi khi **AsyncTask** đã hoàn tất thực thi.

```
public class SimpleAsyncTask extends AsyncTask<Void,Void,String>{
```

Lưu ý: Phần khai báo lớp sẽ được gạch chân màu đỏ vì phương thức doInBackground() cần thiết vẫn chưa được triển khai.

2. Ở trên cùng của lớp, định nghĩa một biến thành viên mTextView của loại WeakReference<TextView>:

```
    private WeakReference<TextView> mTextView;
```

3. Triển khai một hàm tạo cho AsyncTask lấy TextView làm tham số và tạo một tham chiếu weak mới cho TextView đó

```
    public void SimpleAsyncTask(TextView tv) {
        mTextView = new WeakReference<>(tv);
    }
```

AsyncTask cần cập nhật **TextView** trong **Activity** sau khi hoàn tất việc ngủ (trong phương thức **onPostExecute**()). Do đó, hàm tạo cho lớp sẽ cần tham chiếu đến **TextView** để được cập nhật.

Tham chiếu weak (lớp **WeakReference**) dùng để làm gì? Nếu bạn truyền **TextView** vào hàm tạo **AsyncTask** rồi lưu trữ trong biến thành viên, tham chiếu đó đến **TextView** có nghĩa là **Activity** không bao giờ có thể được thu gom rác và do đó sẽ rò rỉ bộ nhớ, ngay cả khi **Activity** bị hủy và được tạo lại như trong thay đổi cấu hình thiết bị. Điều này được gọi là tạo ngữ cảnh rò rỉ và **Android Studio** sẽ cảnh báo bạn nếu bạn thử làm vậy.

Tham chiếu weak ngăn chặn rò rỉ bộ nhớ bằng cách cho phép thu gom rác đối tượng được tham chiếu đó nếu cần.

2.2 Thực thi **doInBackground()**

Phương thức **doInBackground**() được yêu cầu cho cho lớp con **AsyncTask**

1 . Đặt con trỏ của bạn trên lớp đc tô sáng, nhấn Alt + Enter (Option + Enter trên máy Mac) và chọn Implement methods. Chọn **doInBackground()** và nhấp OK. Mẫu phương thức sau được thêm vào lớp của bạn

```
@Override
protected String doInBackground(Void... voids) {
    return "";
}
```

2 Thêm code để tạo một số int ngẫu nhiên từ 0 - 10. Đây là số mili giây mà tác vụ tạm dừng, do vậy hãy nhân số đó với 200 để kéo dài thời gian.

```
1 usage
private Random r = new Random();
1 usage
int n = r.nextInt( bound: 11);
no usages
int s = n * 200;
```

3 Thêm một khối **try/catch** và đặt luồng vào trạng thái ngủ

```
try {
    Thread.sleep(s);
} catch (InterruptedException e) {
    e.printStackTrace();
}
```

4 Thay thế câu lệnh **return** hiện tại để trả về chuỗi "*"Awake at last after sleeping for xx milliseconds*" trong đó *xx* là số mili giây ngủ của ứng dụng.

```
return "Awake at last after sleeping for " + s + " milliseconds!";
```

*Phương thức **doInBackground**() hoàn chỉnh sẽ trông thế này:*

```
@Override
protected String doInBackground(Void... voids) {
    Random r = new Random();
    int n = r.nextInt( bound: 11);
    int s = n * 200;

    try {
        Thread.sleep(s);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }

    return "Awake at last after sleeping for " + s + " milliseconds!";
}
```

2.3 Thực hiện onPostExecute()

Khi phương thức ***doInBackground()*** *hoàn thành, giá trị trả về sẽ được tự động chuyển đến hàm gọi lại* ***onPostExecute()***

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