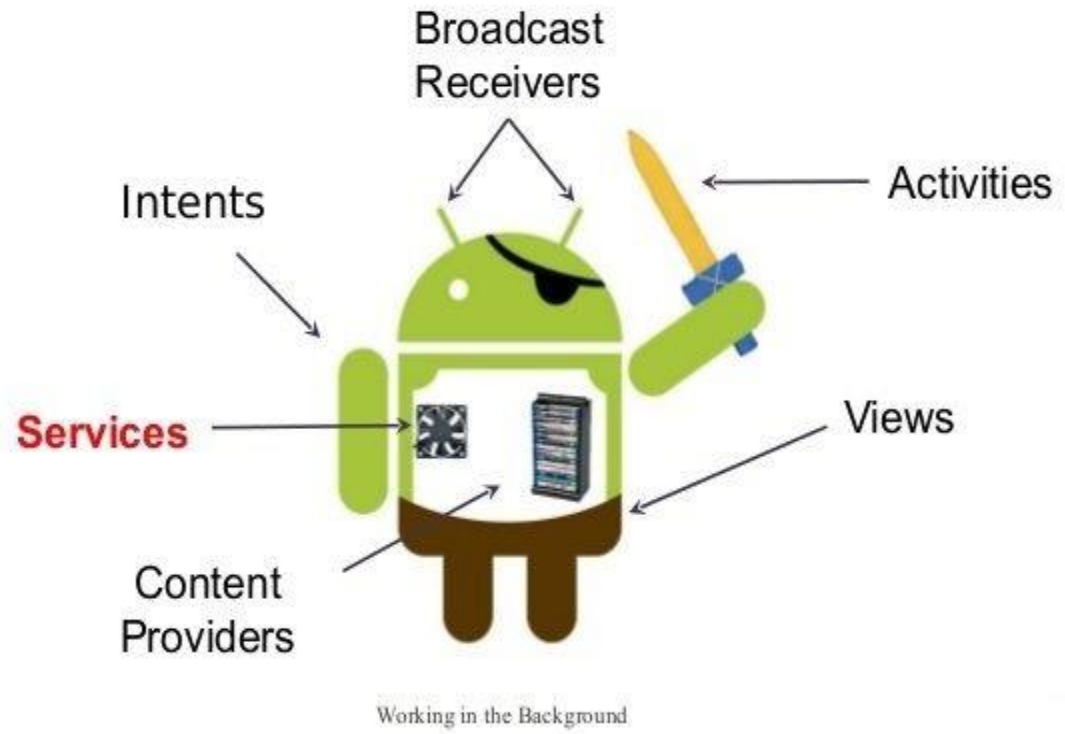
# LẬP TRÌNH CHO CÁC THIẾT BỊ DI ĐỘNG

Bài 2: CÁC THÀNH PHẦN CƠ BẢN TRONG MỘT ỨNG DỤNG ANDROID

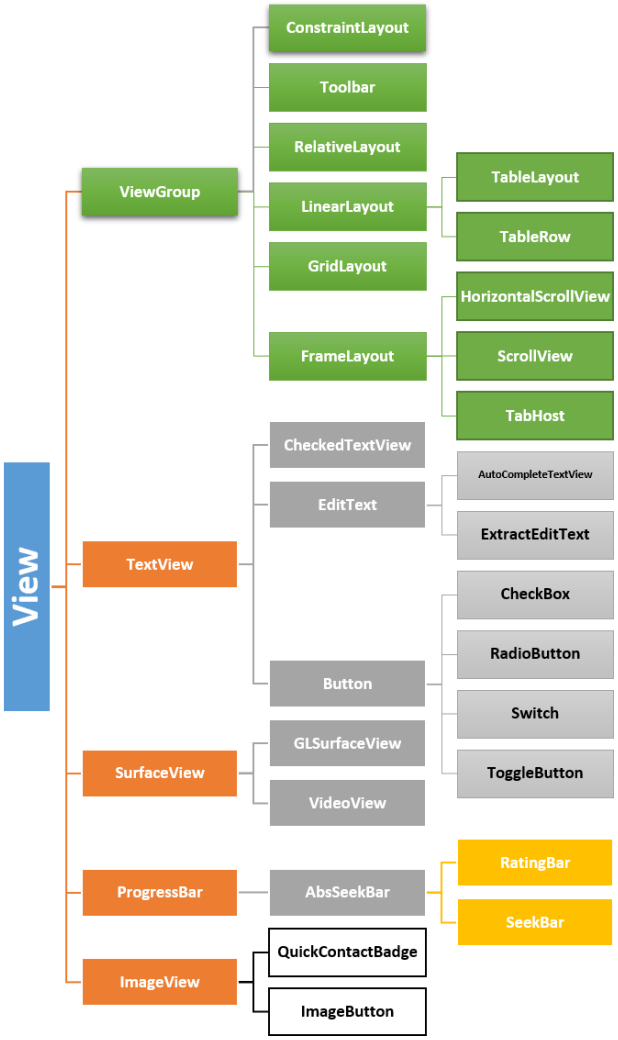
# Giới thiệu chung về các component trong Android



# Views

- Các thành phần giao diện xây dựng từ lớp cơ sở View (android.view.View) của Android, các thành phần này cung cấp sẵn khá đa dạng như Button, TextView, CheckBox ... tất cả chúng ta gọi nó là View. Sơ đồ các View được mộ tả theo sơ đồ như hình dưới.
- View biểu diễn một hình chữ nhật, trong đó nó hiện thị thông tin nào đó cho người dùng, và người dùng có thể tương tác với View. Nhưng loại View cơ bản cần tìm hiểu trước tiên đó là: TextView, ImageView, Button, ImageButton, EditText.

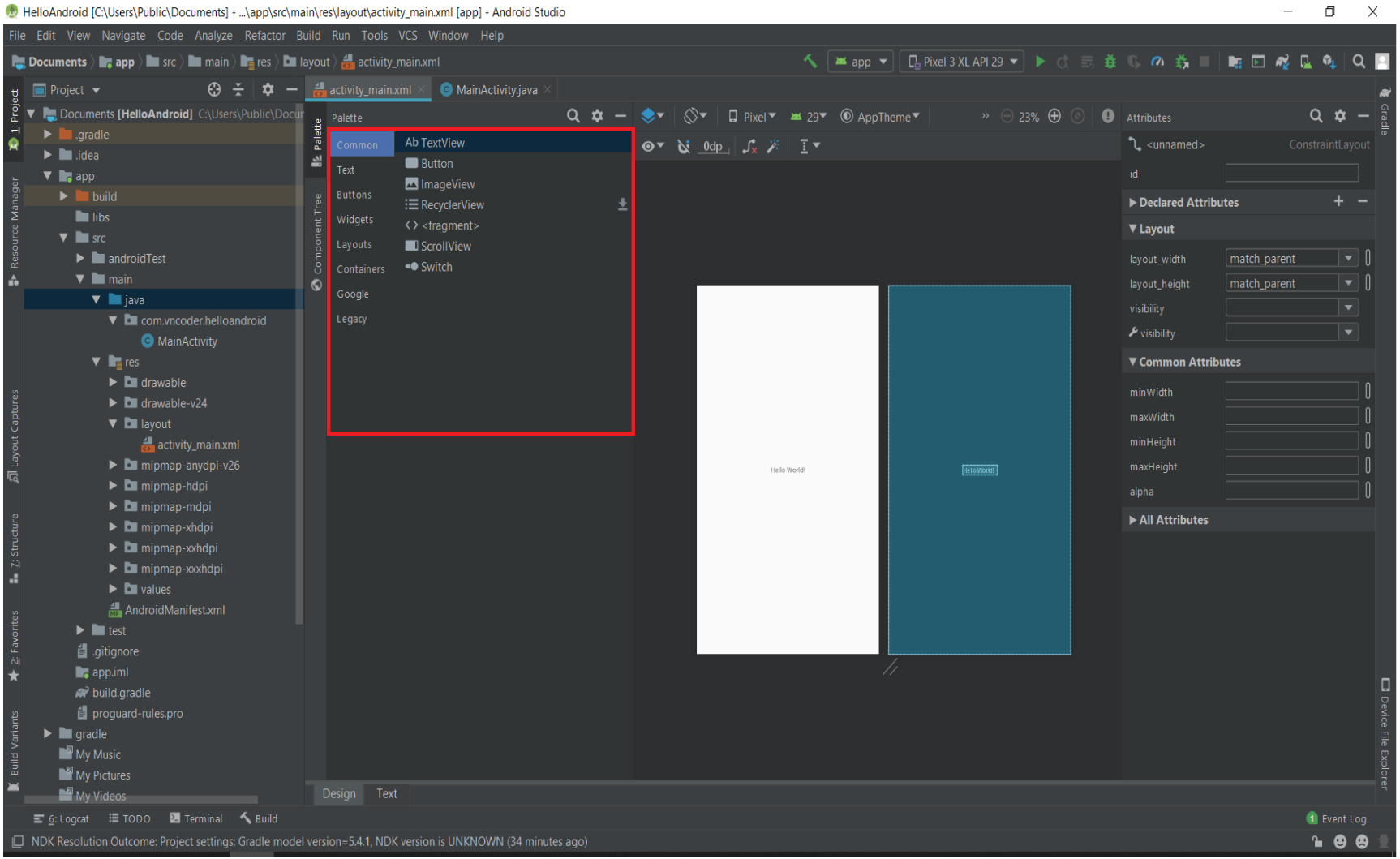
# View


View
ViewGroup
ConstraintLayout
Toolbar
RelativeLayout
LinearLayout
TableLayout
TableRow
GridLayout
FrameLayout
HorizontalScrollView
ScrollView
TabHost
TextView
CheckedTextView
EditText
AutoCompleteTextView
ExtractEditText
Button
CheckBox
RadioButton
Switch
ToggleButton
SurfaceView
GLSurfaceView
VideoView
ProgressBar
AbsSeekBar
RatingBar
SeekBar
ImageView
QuickContactBadge
ImageButton

# ViewGroup

- Trong sơ đồ trên, có một lớp abstract kế thừa từ View là ViewGroup, nó cũng chính là một View nhưng có khả năng chứa các View khác bên trong (kể cả ViewGroup khác). Nó là lớp cơ sở để xây dựng nên các layout như trên sơ đồ ta thấy có: ConstraintLayout, RelativeLayout, LinearLayout, GridLayout, FrameLayout.
- Android có sẵn nhiều loại View, ViewGroup, Layout mà ở chế độ Design bạn có thể kéo thả để xây dựng giao diện
- Các ViewGroup cũng như một phần tử chứa các View con, và ViewGroup cũng được kế thừa trở thành phần gốc để xây dựng UI gọi là layout.

# ViewGroup


HelloAndroid [C:\Users\Public\Documents] - ...\app\src\main\res\layout\activity_main.xml [app] - Android Studio
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Documents app src main res layout activity_main.xml
app
Pixel 3 XL API 29
Project
Documents [HelloAndroid] C:\Users\Public\Docur
.gradle
.idea
app
build
libs
src
androidTest
main
java
com.vncoder.helloandroid
MainActivity
res
drawable
drawable-v24
layout
activity_main.xml
mipmap-anydpi-v26
mipmap-hdpi
mipmap-mdpi
mipmap-xhdpi
mipmap-xxhdpi
mipmap-xxxhdpi
values
AndroidManifest.xml
test
.gitignore
app.iml
build.gradle
proguard-rules.pro
gradle
My Music
My Pictures
My Videos
activity_main.xml
MainActivity.java
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Google
Legacy
Ab TextView
Button
ImageView
RecyclerView
<fragment>
ScrollView
Switch
Pixel
29
AppTheme
23%
0dp
Hello World!
Attributes
<unnamed>
ConstraintLayout
id
Declared Attributes
Layout
layout_width
match_parent
layout_height
match_parent
visibility
visibility
Common Attributes
minWidth
maxWidth
minHeight
maxHeight
alpha
All Attributes
Design
Text
6: Logcat
TODO
Terminal
Build
Event Log
NDK Resolution Outcome: Project settings: Gradle model version=5.4.1, NDK version is UNKNOWN (34 minutes ago)

# Các layout

- Các layout chính là các View (cụ thể nó kế thừa thừa ViewGroup) được thiết kế với mục đích chứa các View con và điều khiển, sắp xếp vị trí các View con đó trên màn hình, mỗi layout có cơ chế điều khiển vị trí View con riêng của mình. Có một số layout mà bạn tham khảo trước tiên như FrameLayout, ConstraintLayout, LinearLayout, RelativeLayout, GridLayout, TableLayout, CoordinatorLayout....
- Vấn đề này sẽ nói kỹ ở các bài sau.

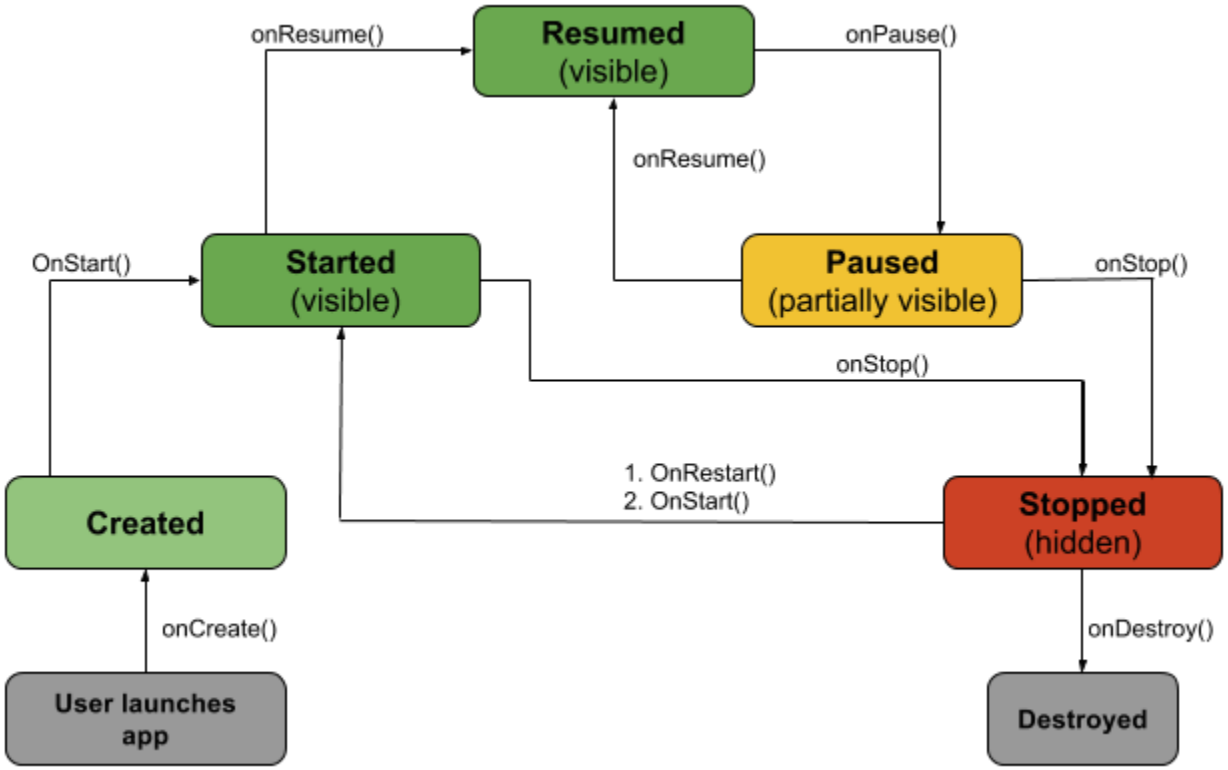
# Hệ thống cấp bậc các View

- Mỗi View trong hệ thống giao diễn được biểu diễn bởi một hình chữ nhật, nó vẽ trên nó thông tin gửi tới người dùng cũng như là nơi để nhận thông tin nhập vào như nhập dữ liệu, chạm, vuốt.
- Một giao diện người dùng được bắt đầu bằng một View gốc, bên trong nó chứa các View con, đến lượt các View con lại có thể chứa các View con khác ... Cứ như vậy, giao diện hình thành được biểu diễn như một đồ thị dạng cây nhiều nhánh (view hierarchy).

# Activities

- Activity là một thành phần của ứng dụng Android. Android Activity là nơi để ứng dụng tương tác trực tiếp với người dùng thông qua giao diện. Một ứng dụng có thể sẽ có nhiều màn hình và mỗi màn hình có thể là một Activity (nếu không sử dụng Fragment).
- Mỗi Activity thường hoạt động độc lập với nhau nhưng có thể tương tác và truyền dữ liệu qua nhau thông qua Intent. Chính vì Activity hoạt động độc lập nên sẽ có vòng đời riêng từ lúc được khởi tạo cho đến lúc được huỷ đi.

# Vòng đời Activity



# Intents

- Trong Android, khả năng gửi tin nhắn đi xung quanh được thực hiện bởi đối tượng Intent. Với sự trợ giúp của Intents, các thành phần của Android có thể yêu cầu chức năng từ các thành phần khác.
- Intent cũng giúp liên lạc giữa các phần của một ứng dụng dễ dàng. Di chuyển từ một màn hình (Activity) sang một màn hình khác được thực hiện thông qua Intents.
- Tất cả các thành phần (ứng dụng và màn hình) của thiết bị Android đều bị cô lập. Cách duy nhất chúng giao tiếp với nhau là thông qua Intents.
- Android hỗ trợ hai loại Intent : minh bạch và ngầm. Khi một ứng dụng định nghĩa thành phần đích của nó trong một Intent, đó là một Intent minh bạch. Khi ứng dụng không đặt tên cho một thành phần đích, đó là một Intent ngầm.

# Broadcast - receiver

- Broadcast Receiver là một trong 4 component lớn trong Android, với mục đích là lắng nghe các sự kiện, trạng thái của hệ thống phát ra thông qua Intent nhờ đó mà các lập trình viên có thể xử lí được các sự kiện hệ thống ở bên trong ứng dụng của mình.
- Broadcast Receiver có thể hoạt động được cả khi ứng dụng bị tắt đi, nghĩa là ở background chính vì vậy nó thường được sử dụng với service.

# Services

- Một Service là một thành phần (component) có thể thực hiện các hoạt động lâu dài trong background và nó không cung cấp một giao diện người dùng. Một thành phần khác của ứng dụng có thể start nó, và nó tiếp tục chạy trong background ngay cả khi người dùng chuyển sang ứng dụng khác.
- Ngoài ra một thành phần có thể liên kết (bind) với một Service để tương tác với Service đó, thậm chí là thực hiện truyền thông liên tiến trình IPC (interprocess communication - IPC bạn có thể hiểu là một hoạt động chia sẽ dữ liệu qua nhiều tiến trình, thông thường sử dụng giao thức truyền thông và nó phải có Client và Server). Ví dụ: một Service có thể thực hiện các giao dịch mạng, chơi nhạc, ra vào file I/O hoặc tương tác với một content provider, tất cả đều từ background.

# Phân loại Service

- Foreground Service
- Background Service
- Bound Service

# Foreground Service

- Một Foreground Service thực hiện một số thao tác mà người dùng chú ý, có thể thấy rõ ràng.
- Ví dụ một ứng dụng nghe nhạc có thể chơi một bản nhạc và điều khiển nó bằng Foreground Service. Một điều bắt buộc là Foreground Service phải hiện thị một Notification. Foreground Service sẽ tiếp tục chạy ngay cả khi người dùng không tương tác với ứng dụng.

# Background Service

- Một Background Service sẽ thực hiện các hoạt động mà không được người dùng chú ý trực tiếp.
- Ví dụ một ứng dụng sử dụng một service để thu gom bộ nhớ chẳng hạn thì service là một Background Service, hoạt động mà người dùng không cần thiết phải để ý.

# Bound Service

- Là loại service được gọi bằng phương thức **bindService**().
- Một component (ví dụ như Activity chẳng hạn) gọi Service bằng phương thức **bindService**(). Activity sẽ liên kết với Service theo dạng client – server. Lúc này Activity có tương tác với Service để gửi và nhận kết quả từ Service.
- Ví dụ, mình tạo một Service có nhiệm vụ cập nhật thời tiết. Activity sẽ bind vào Service và yêu cầu Service cập nhật dữ liệu thời tiết bất kì lúc nào, sau đó trả kết quả cho activity hiển thị cho người dùng.

# Content provider

- Content provider là một thành phần để quản lý truy cập dữ liệu, nó cung cấp các phương thức khác nhau để các ứng dụng có thể truy cập dữ liệu từ một ứng dụng khác bằng cách sử dụng ContentResolver.
- Content Provider có thể giúp cho một ứng dụng quản lý quyền truy cập đến dữ liệu được lưu bởi ứng dụng đó, hoặc các ứng dụng khác, và đó là một cách để ta có thể chia sẻ dữ liệu cho các ứng dụng khác nhau.
- **Content Provider** hoạt động rất giống với một cơ sở dữ liệu, bạn có thể truy vấn, chỉnh sửa nội dung, cũng như là thêm xóa các nội dung sử dụng các phương thức: **insert(), update(), delete(), query()**.

# Content provider