



ĐỀ CƯƠNG BÀI GIẢNG

BÀI 2. THỰC HÀNH XÂY DỰNG GIAO DIỆN ỨNG DỤNG

NỘI DUNG BÀI HỌC

Nội dung bài học trước khi lên lớp (trang 2 đến 17).

- Phiếu bài tập: bài 2.1 đến 2.4
- Màn hình thiết kế giao diện
- Thiết kế giao diện bằng mã lệnh
- Màn hình logcat
- Thiết kế giao diện với các layout hay dùng: RelativeLayout, TableLayout, LinearLayout.

Nội dung bài học thực hiện lên lớp (Trang 17 đến hết).

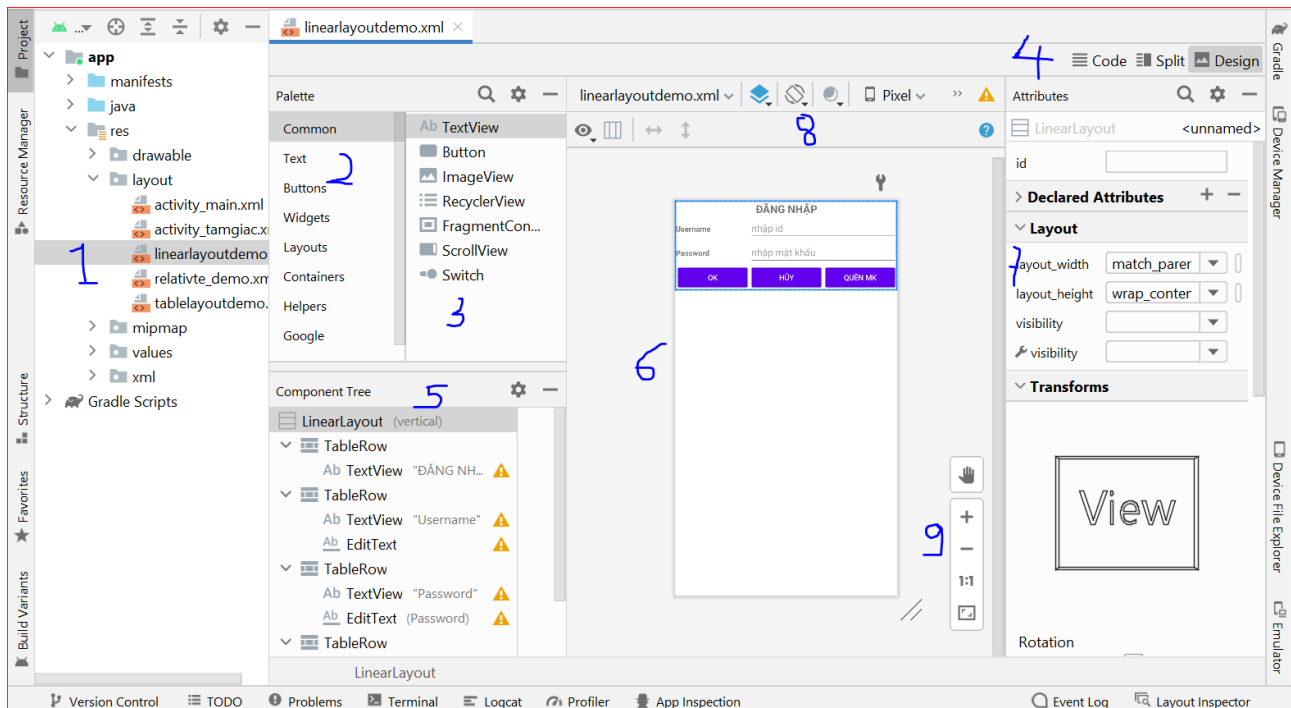
- Phiếu bài tập : bài 2.5
- Vòng đời ứng dụng Android
- Viết và xem nhật ký bằng logcat

Nội dung bài học sau khi lên lớp: Phiếu bài tập bài 2.6

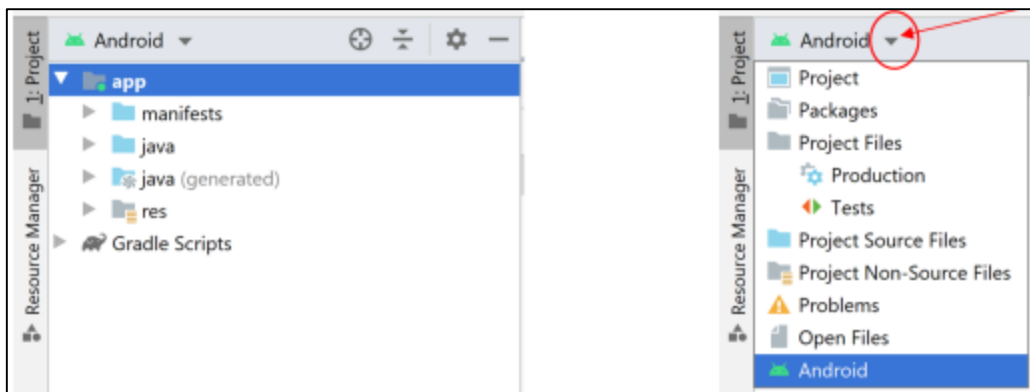
Mục lục

1.	MÀN HÌNH THIẾT KẾ GIAO DIỆN.....	2
2.	THIẾT KẾ GIAO DIỆN BẰNG MÃ LỆNH.....	10
3.	MÀN HÌNH LOGCAT	12
4.	THIẾT KẾ GIAO DIỆN VỚI TABLELAYOUT	13
5.	THIẾT KẾ GIAO DIỆN VỚI LINEARLAYOUT.....	15
6.	THIẾT KẾ GIAO DIỆN VỚI RELATIVELAYOUT.....	16
7.	VÒNG ĐỜI ỨNG DỤNG ANDROID.	17
8.	VIẾT VÀ XEM NHẬT KÝ BẰNG LOGCAT	20

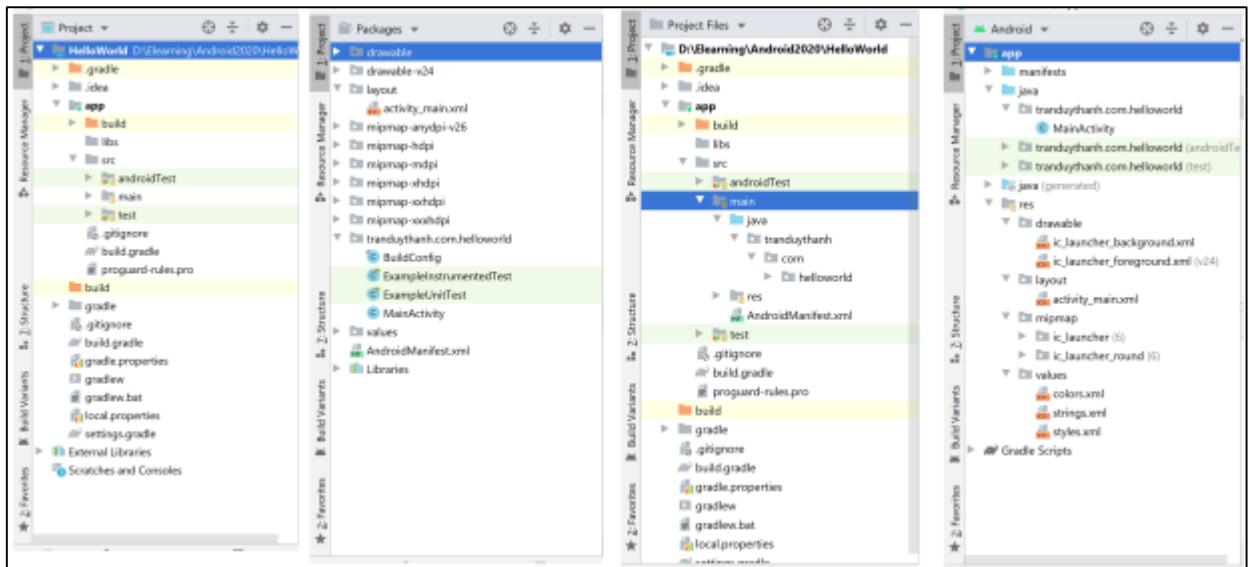
1. Màn hình thiết kế giao diện



1.1. Vùng 1: Vùng hiển thị cấu trúc thư mục của Project.



Mỗi layout quan sát thì cấu trúc Project sẽ hiển thị khác nhau. Thông thường nhóm hiển thị hay được sử dụng nhiều nhất vẫn là chế độ Android, dưới đây là ví dụ một số loại layout quan sát (lần lượt: Project, Packages, Project Files, Android):

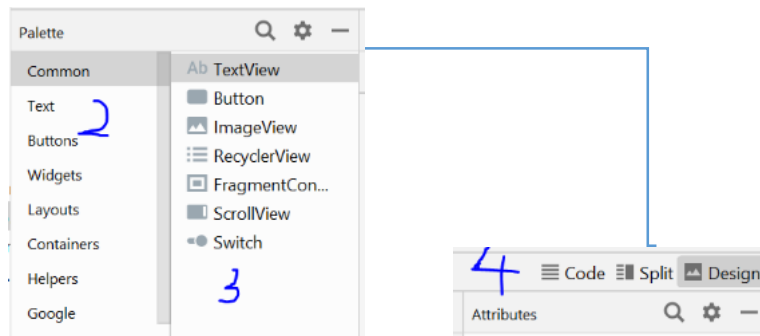


Vị trí nội dung chứa các file thiết kế giao diện: res/layout

1.2. Vùng 2: Palette : Vùng chứa các bảng phân loại View/ViewGroup

Vùng này chỉ được hiển thị nếu vùng 4 chọn ở chế độ Design

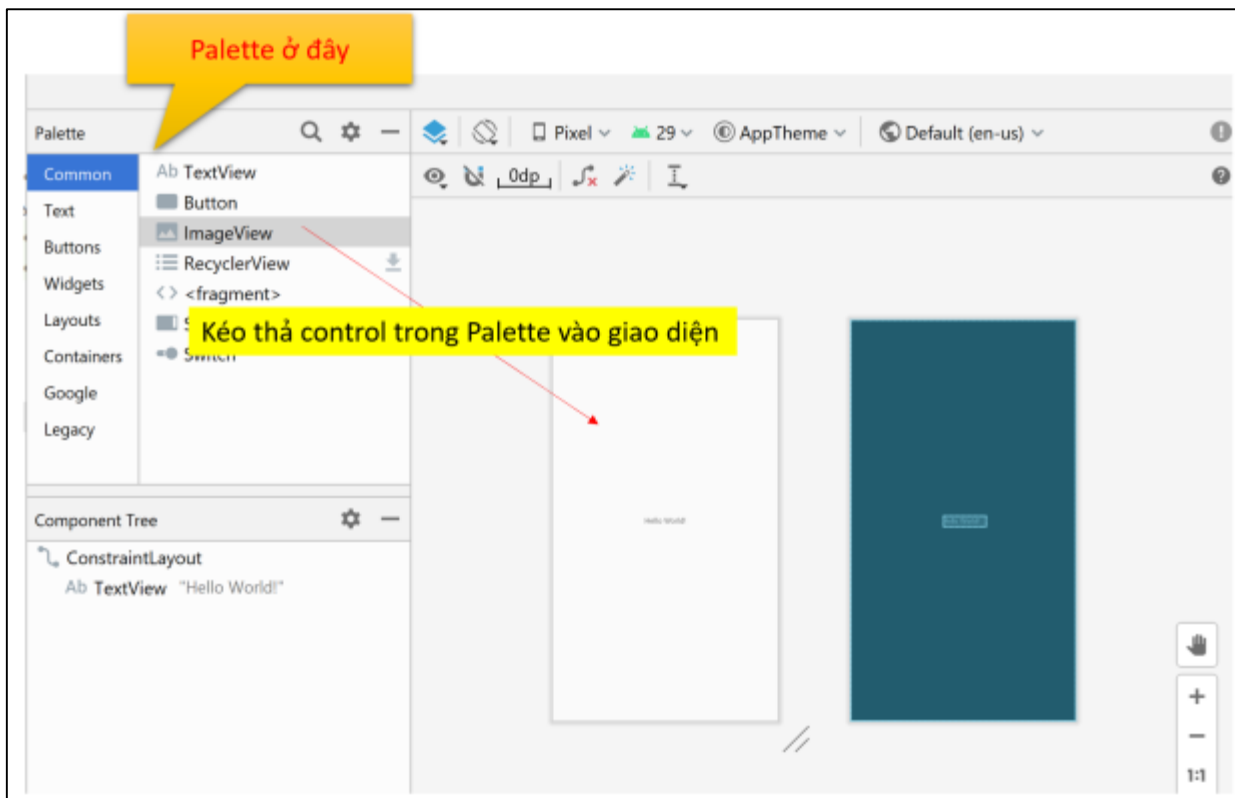
Android Studio phân nhóm các loại control giúp ta dễ dàng chọn lựa kéo thả ra màn hình thiết kế: Commons (textview, button...), Text (TextView, EditText...), Text(EditText, Password...), Layout (LinearLayout, FrameLayout...)



Vùng 2 sẽ hiển thị nhóm (group) các điển khiển chi tiết trong vùng 3.

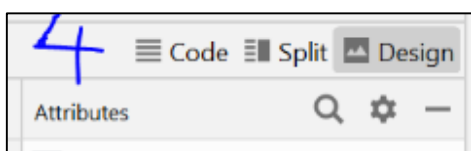
1.3. Vùng 3: Kéo thả thiết kế

Công cụ Palette cho phép ta kéo thả các control vào màn hình thiết kế của điện thoại:



Để kéo thả Control ra màn hình điện thoại ta chỉ cần chọn một control bất kỳ nào đó trong Palette rồi nhấn chuột kéo trực tiếp vào màn hình điện thoại màu trắng sau đó nhả chuột ra.

1.4. Vùng 4: Code, Split, Design.

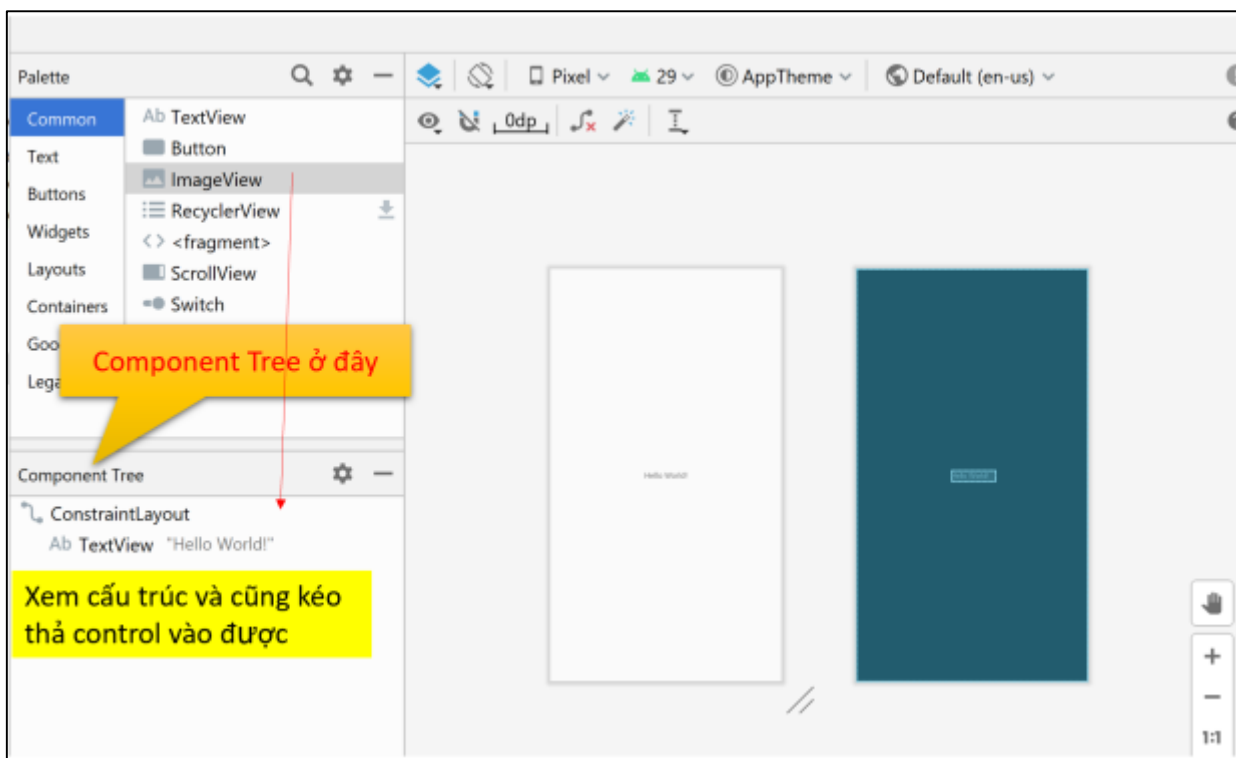


Vùng chứa các lựa chọn được dùng để thiết kế giao diện cho ứng dụng, phần **Design** để thiết kế giao diện bằng kéo thả, phần **Code** dùng để thiết kế bằng mã XML, phần **Split** để vừa thấy code XML vừa thấy giao diện biểu diễn tương ứng các chỉnh sửa XML. Kết quả hiển thị trong vùng 6

1.5. Vùng 5: Component Tree

Vùng này chứa cấu trúc cây thư mục thiết kế giao diện. Hiển thị rõ ViewGroup, View bố trí và thuộc tính mã (ID) đặt cho View/ViewGroup.

Màn hình Component Tree giúp ta quan sát được cấu trúc của các control kéo thả trên giao diện, hỗ trợ rất tốt cho việc điều chỉnh thiết kế:



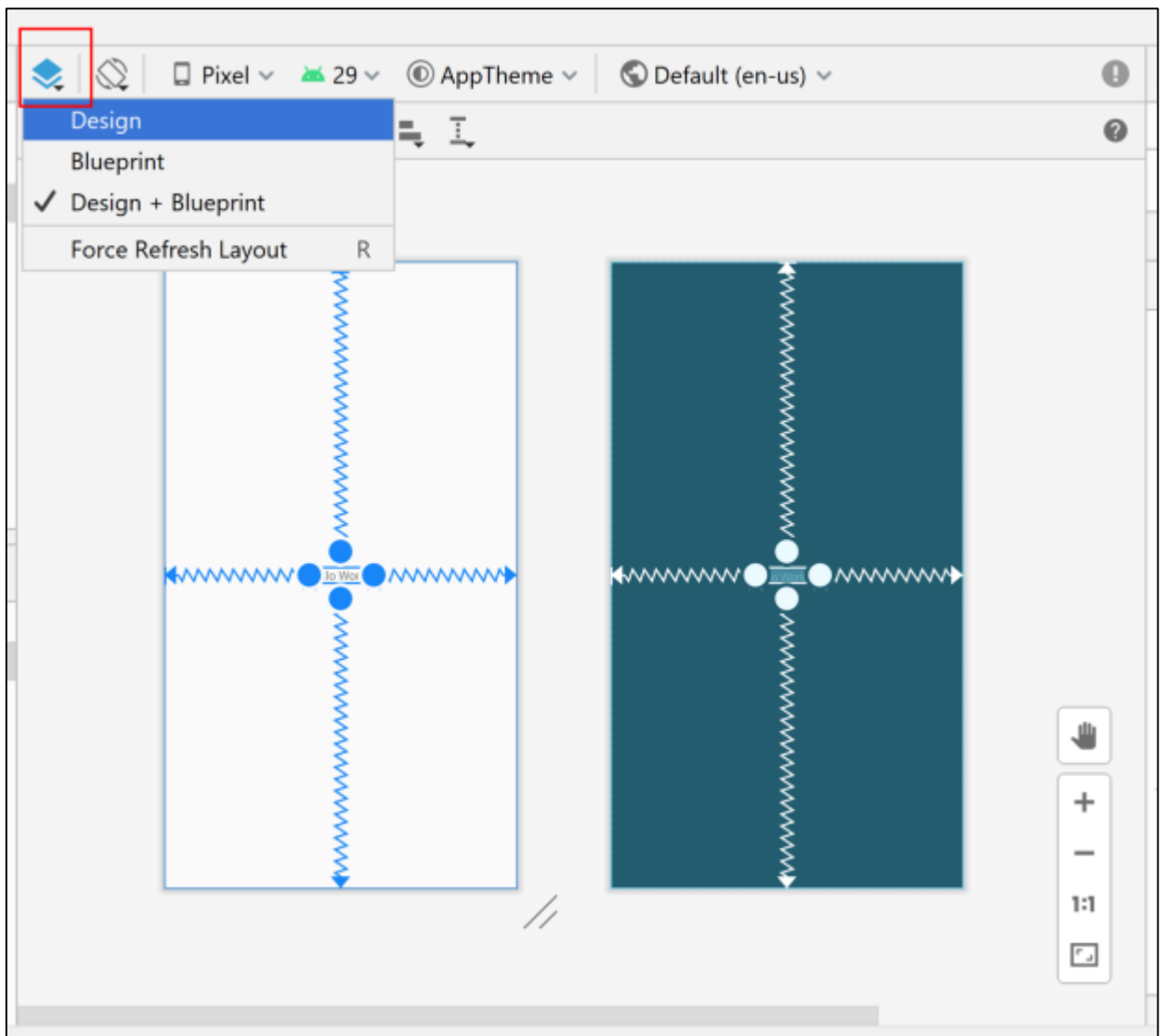
Đặc biệt ta cũng có thể kéo thả control trực tiếp vào Component Tree để thiết kế giao diện

1.6. Vùng 6: Kết quả hiển thị thiết kế phác thảo.

Khi kéo các điều khiển ở vùng 3 vào vùng 6. Nếu vùng 4 vẫn để chế độ Design ta sẽ nhìn thấy kết quả thiết kế giao diện

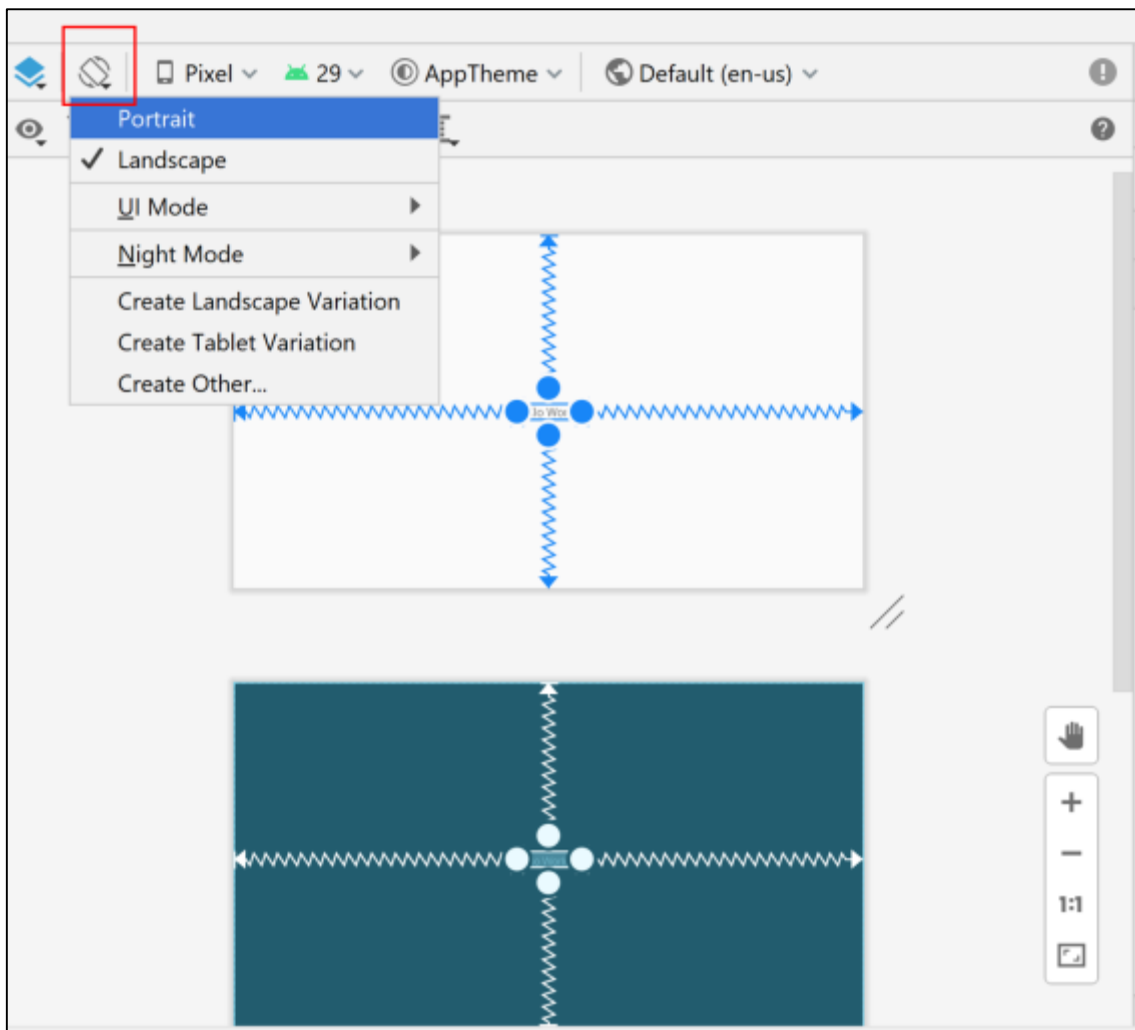
Chế độ hiển thị điện thoại: Android Studio cung cấp cho ta màn hình điện thoại để thiết kế:

- Chọn Design: Hiển thị khung trắng để kéo thả điều khiển (control)
- Chọn Blueprint: Hiển thị khung xanh xem cấu trúc
- Chọn Design & Blueprint: Hiển thị cả 2 khung trắng khung xanh

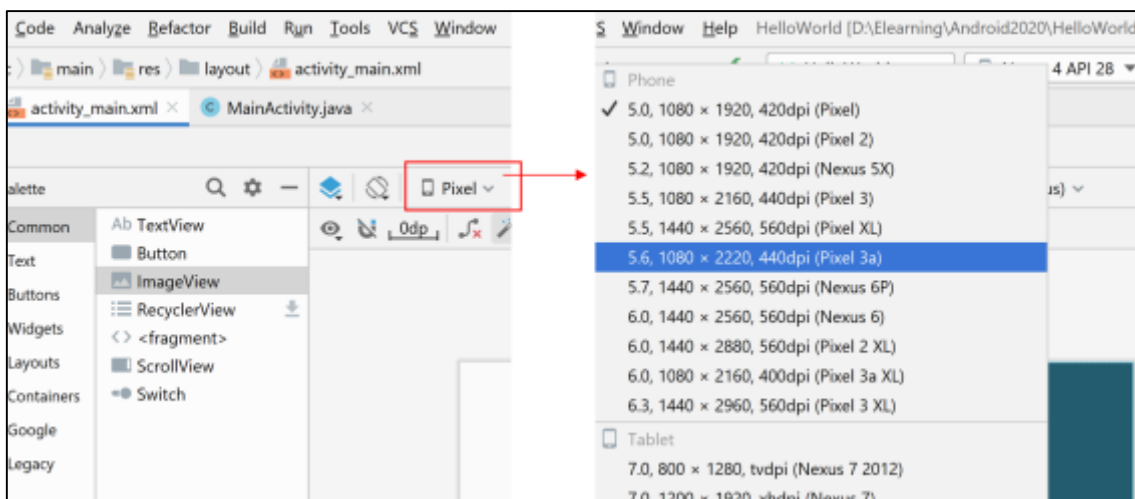


Xoay màn hình thiết kế và chế độ Test thử độ phân giải

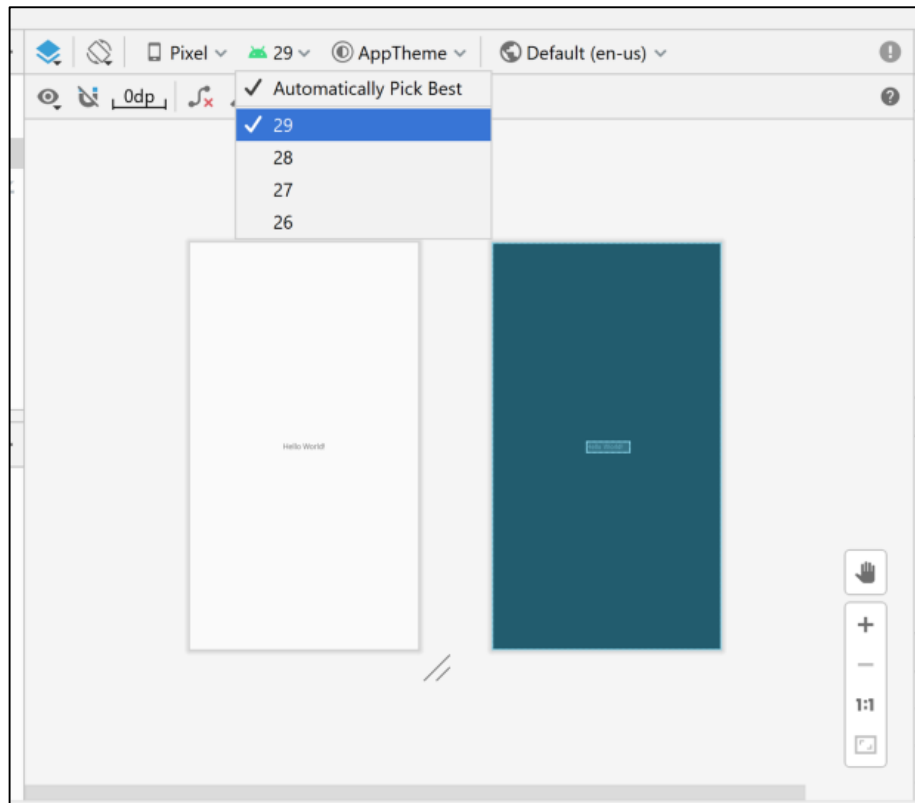
- Ở màn hình sau ta có thể xoay đứng màn hình (Portrait), xoay ngang màn hình (Landscape).



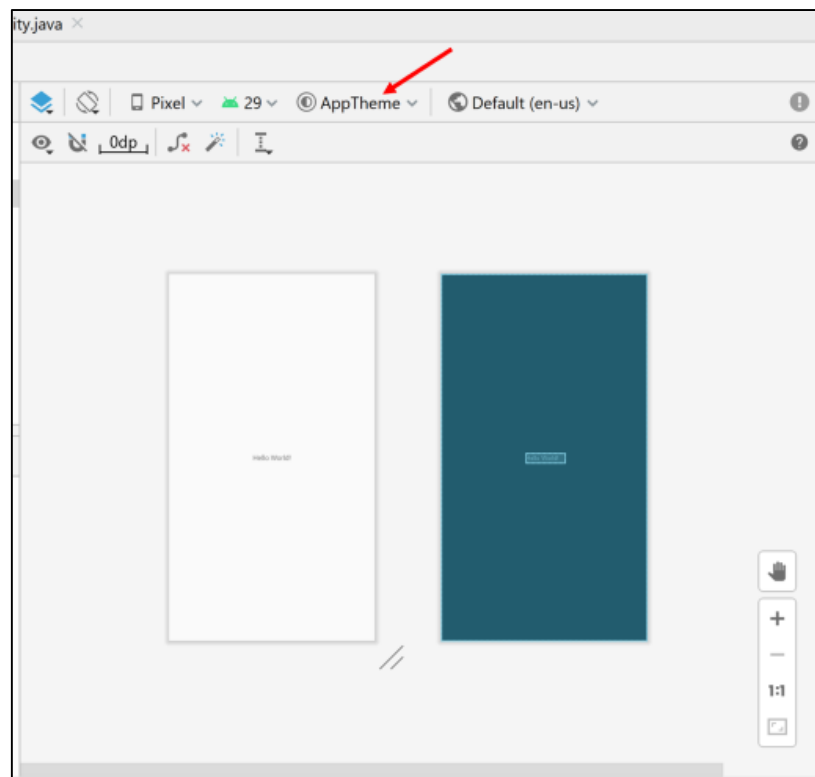
Đổi độ dòng máy để test thử:



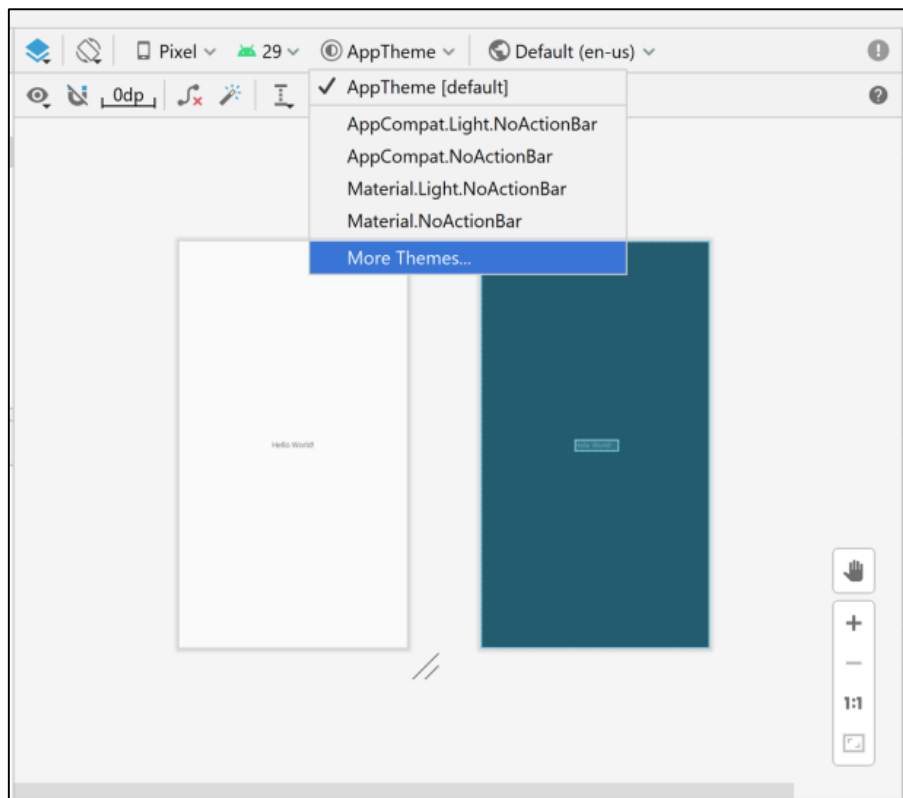
Đổi phiên bản API: Khi thay đổi biên dịch phần mềm với phiên bản Android SDK nào thì ta chọn API đó.



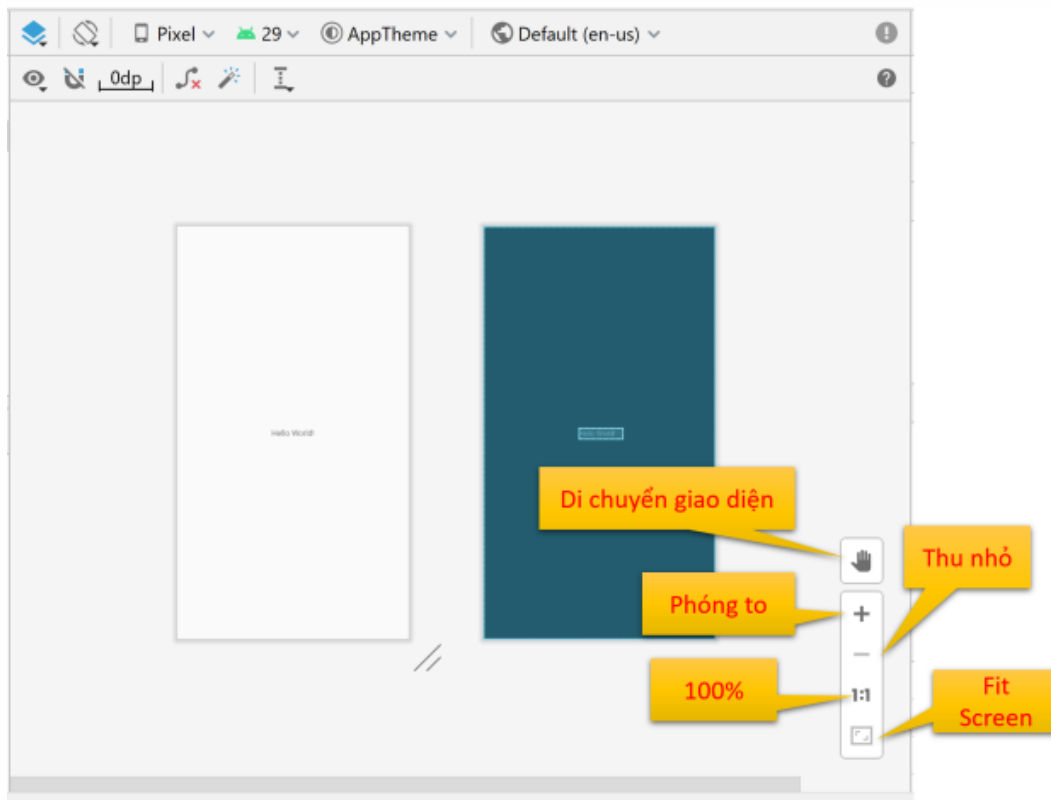
Đổi Theme: Android Studio cho phép ta đổi Theme điện thoại tùy theo nhu cầu sử dụng, để đổi Theme ta chọn AppTheme:



Khi chọn AppTheme, Android Studio sẽ hiển thị màn hình danh sách các Theme cho phép ta thay đổi:

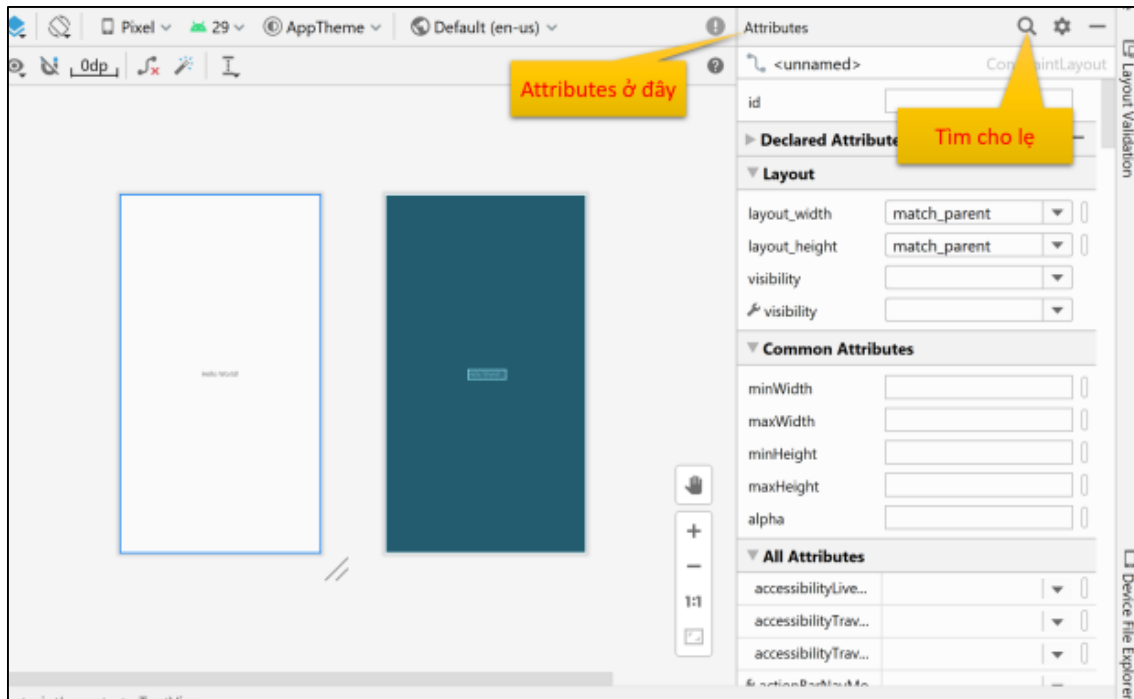


Phóng to, thu nhỏ, di chuyển giao diện: Trong quá trình thiết kế giao diện, việc phóng to thu nhỏ màn hình rất tiện lợi cho lập trình viên:



1.7. Vùng 7: Màn hình Attributes

Màn hình **Attributes** rất quan trọng, được sử dụng thường xuyên trong quá trình thiết lập các trạng thái cho control trên giao diện (đặt Id, độ rộng, độ cao, font chữ, vị trí...):



2. Thiết kế giao diện bằng mã lệnh

Vùng 4 nếu chọn Code sẽ cho phép người lập trình thiết kế giao diện thông mã lệnh XML

2.1. Mô tả ViewGroup

Mỗi phần tử ViewGroup được đặt trong **cặp thẻ mở** <Tên ViewGroup> và thẻ đóng </Tên ViewGroup>. Các thuộc tính của ViewGroup được đặt trong cặp thẻ tự mở. Ví dụ


```
<TableRow  
    android:layout_height="match_parent"  
    android:layout_width="match_parent" >  
</TableRow>
```

2.2. Mô tả View

Mỗi phần tử View được mô tả thông qua cặp thẻ theo tự đóng mở <View />. Ví dụ:

```
<TextView  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_gravity="center"/>
```

Minh họa mô tả hai View trong ví dụ trên TextView và Button và ViewGroup là LinearLayout. Kết quả của bố cục như sau:



```
<?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:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:orientation="vertical">  
  
    <TableRow  
        android:layout_height="match_parent"  
        android:layout_width="match_parent" >  
  
        <TextView  
            android:layout_width="match_parent"  
            android:layout_height="wrap_content"  
            android:layout_gravity="center"  
            android:layout_weight="1"  
            android:gravity="center"  
            android:text="ĐĂNG NHẬP"  
            android:textSize="20sp"  
            android:textStyle="bold" />  
  
        </TableRow>
```

2.3. Sử dụng tài nguyên đã định nghĩa

a. Nhóm các loại tài nguyên

1. Các tài nguyên đặt trong thư mục **res/**. Thông thường tài nguyên có một số thư mục nổi bật như hình sau:

- Res/ drawable: Lưu trữ tài nguyên dạng ảnh (*.jpg, *.gif, *.png) và các tài nguyên có thể kéo thả được khác
- Res/ layout: Thư mục layout chứa các tập tin XML xác định giao diện người dùng

- Res/mipmap: Tương tự như drawable, thư mục mipmap chấp nhận các hình ảnh bitmap, nhưng nó được sử dụng đặc biệt cho các biểu tượng launcher của ứng dụng
- **Res/ values:** có thể chứa nhiều tập tin XML bao gồm các giá trị đơn giản được sử dụng bên trong các ứng dụng

```
MyProject/  
  src/  
    MainActivity.java  
  res/  
    drawable/  
      graphic.png  
    layout/  
      main.xml  
      info.xml  
    mipmap/  
      icon.png  
    values/  
      strings.xml
```

b. Truy cập tài nguyên

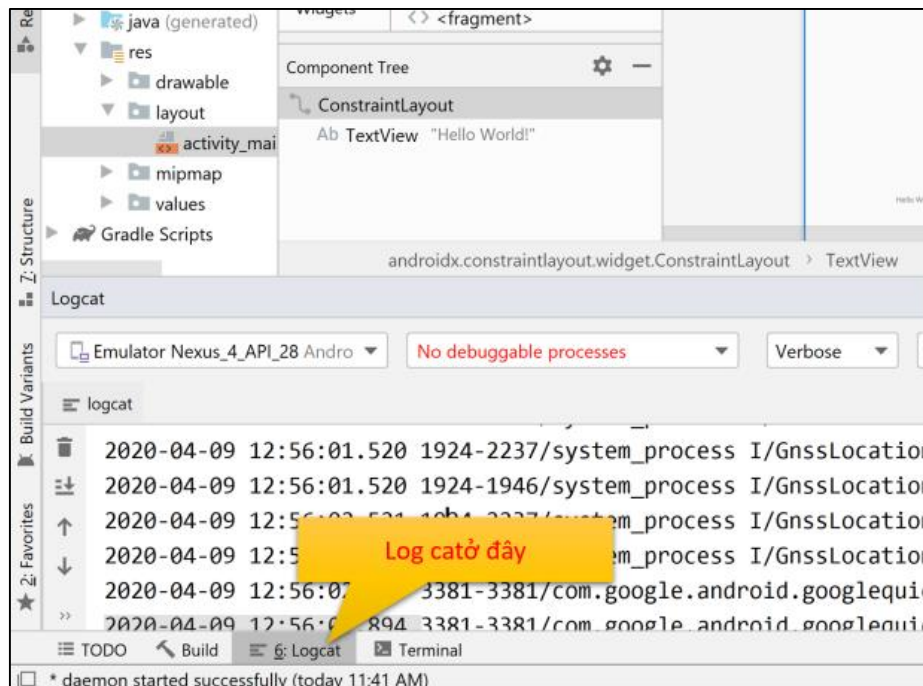
- Truy cập tài nguyên trong java: Truy cập thông qua R.id của tập tin. Ví dụ **R.drawable.kittens**
- Truy cập tài nguyên trong XML. Sử dụng mẫu lệnh “@tên_thư_mục/tên_tài_nguyên. Ví dụ

```
<application  
    android:allowBackup="true"  
    android:dataExtractionRules="@xml/data_extraction_rules"  
    android:fullBackupContent="@xml/backup_rules"  
    android:icon="@mipmap/ic_launcher"  
    android:label="@string/app_name"
```

3. Màn hình logcat

Màn hình Logcat rất quan trọng cho lập trình viên trong quá trình theo dõi lỗi phát sinh khi chạy phần mềm, dựa vào các mô tả chi tiết lỗi trong Logcat mà ta có thể dễ dàng tìm ra các giải pháp để sửa lỗi một cách nhanh chóng. Để hiển thị màn hình này

thực hiện quan sát dưới cùng màn hình Android Studio/ mục Android Monitor. Chọn mục này để xuất hiện màn hình Logcat:

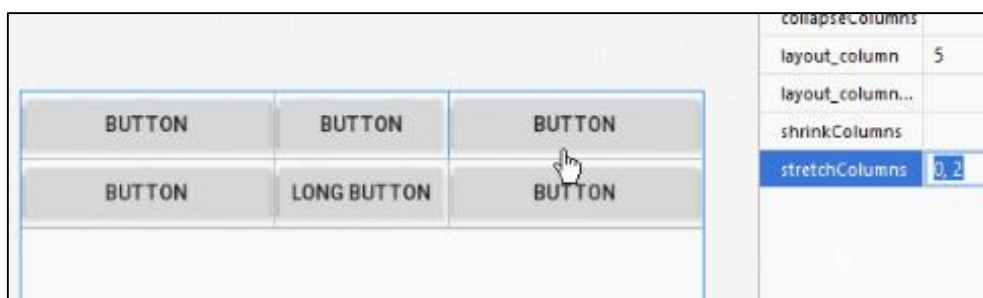


4. Thiết kế giao diện với TableLayout

TableLayout sắp xếp các View bên trong nó dưới dạng bảng. TableLayout là một ViewGroup chứa một hoặc nhiều TableRow, mỗi TableRow là một hàng (row) trong bảng chứa các ô (cell). Các View con có thể được đặt trong một ô hoặc đặt trong một ô hợp nhất từ các ô liên tiếp của một hàng, **không thể hợp nhất các ô liên tiếp trên cùng một cột**. Cột / hàng trong bảng bắt đầu từ số 0.

Các thuộc tính cần lưu ý

- **android:stretchColumns**="col1, col2 ..." (ký hiệu * tất cả các cột). sẽ thiết lập độ rộng cột giãn ra lớn lấp đầy không gian còn trống của layout. **Ví dụ:**

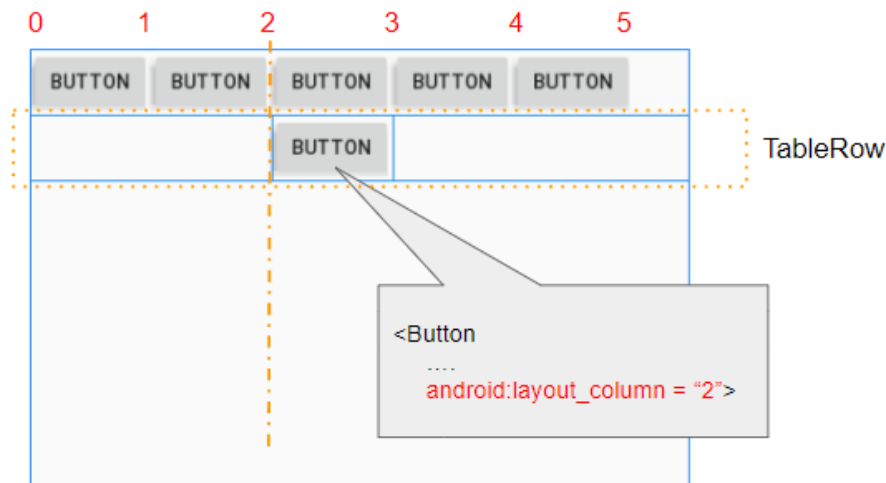


- **android:shrinkColumns** ="col1, col2 ...". Chỉ định cột co lại để tránh các View con bị tràn ra ngoài TableLayout. Ví dụ:

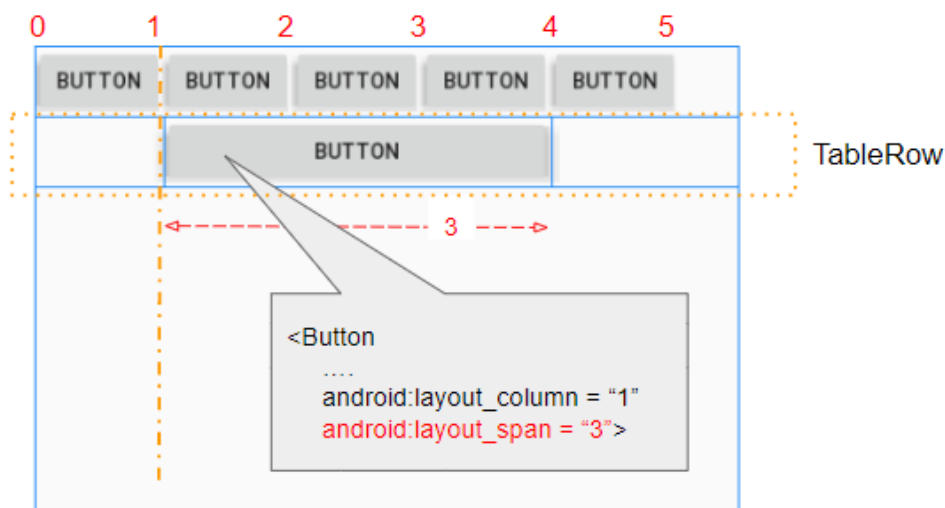
Học kết hợp



- **android:collapseColumns**= "col1, col2 ...". Chỉ định cột bị ẩn đi (độ rộng=0).
- **android:layout_column**=""col1, col2 ..." chỉ định số cột cụ thể của View.

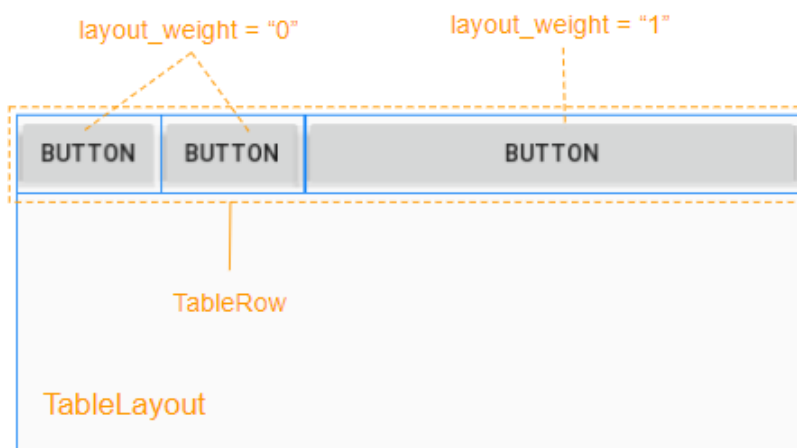
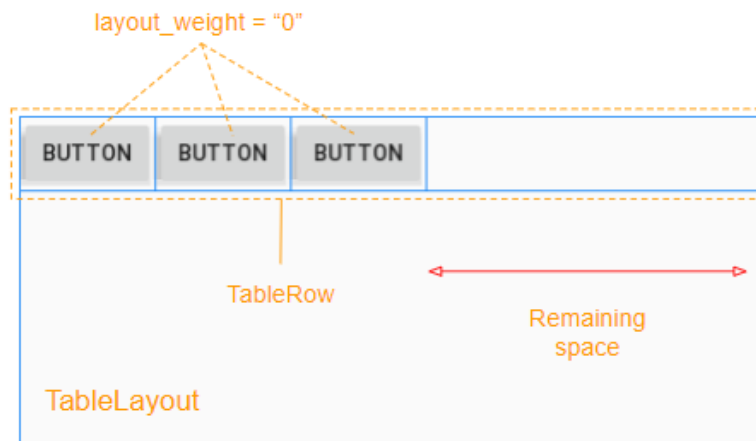


- **android:layout_span**=”numCol). Thiết lập độ rộng gộp các ô trong cùng hàng theo số numCol. Ví dụ android:layout_span="2" thì cell có độ rộng mở rộng ra 2 cột.



- **android:Layout_Weight**: chỉ định có bao nhiêu không gian mà View con này sẽ chiếm trong View cha (TableRow) theo chiều ngang. View con có thể chỉ định một giá trị layout_weight, và sau đó bất kỳ không gian còn lại trong View cha sẽ

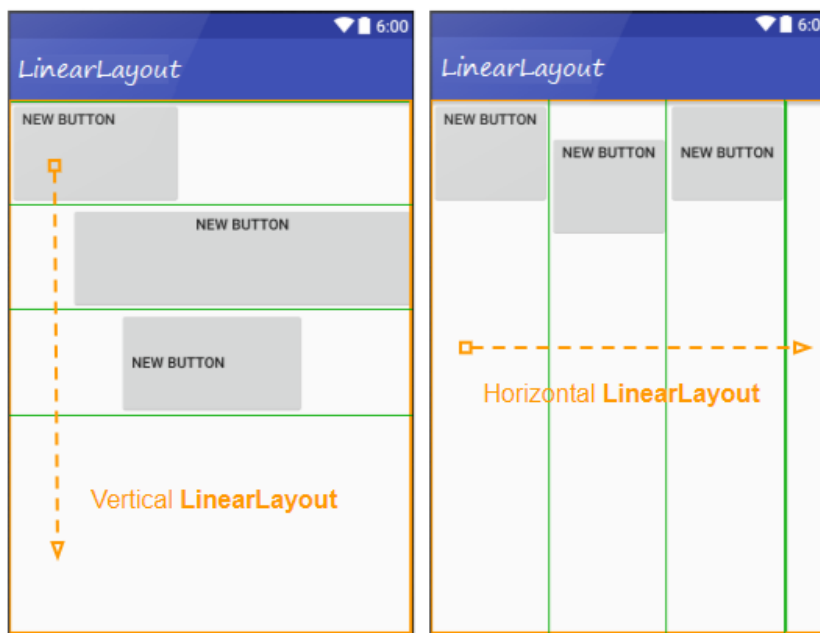
được gán cho các View con theo tỷ lệ `layout_weight` của chúng. Ví dụ:



- **android:sumWeight:** Tổng số weight của các View con

5. Thiết kế giao diện với `LinearLayout`

`LinearLayout` là một `ViewGroup` nó sắp xếp các View con theo một hướng duy nhất, theo chiều dọc hoặc chiều ngang. Bạn có thể chỉ định hướng (orientation) của nó bằng cách sử dụng thuộc tính: `android:orientation`



Các thuộc tính cần ghi nhớ:

- **android:layout_gravity, android:layout_weight; android:Sumweight:**
tương tự như trong TableLayout
- **android:orientation:** Hướng sắp xếp view chứa trong layout

6. Thiết kế giao diện với RelativeLayout

RelativeLayout là layout mà các View con được xác định vị trí bởi các mối liên hệ với View cha hoặc với View con như View con nằm dưới một View con khác, View con căn thẳng lề phải với View cha

6.1. Định vị View con bằng liên hệ với View cha trong RelativeLayout

Vị trí của View con trong RelativeLayout có thể thiết lập bằng cách chỉ ra mối liên hệ vị trí với view cha như căn thẳng cạnh trái View cha với View con, căn thẳng cạnh phải View cha với View con. Các thuộc tính thực hiện chức năng này như sau:

Thuộc tính	Ý nghĩa
android:layout_alignParentBottom	Căn thẳng cạnh dưới view con với cạnh dưới View cha
android:layout_alignParentLeft	Căn thẳng cạnh trái view con với cạnh trái View cha
android:layout_alignParentRight	Căn thẳng cạnh phải view con với cạnh phải View cha
android:layout_alignParentTop	Căn thẳng cạnh trên view con với cạnh trên View cha
android:layout_centerInParent	Căn view con vào giữa View cha
android:layout_centerHorizontal	Căn view con vào giữa View cha theo chiều ngang
android:layout_centerVertical	Căn view con vào giữa View cha theo chiều đứng

6.2. Định vị View con bằng liên hệ giữa chúng với nhau

View con trong RelativeLayout ngoài mối liên hệ với View cha như trên, chúng có thể thiết lập liên hệ với nhau ví dụ như View con này nằm phía trên một View con khác, nằm phía dưới một view con khác. Các thuộc tính thực hiện chức năng này như sau:

Thuộc tính	Ý nghĩa
<code>android:layout_below</code>	Nằm phía dưới View có ID được chỉ ra
<code>android:layout_above</code>	Nằm phía trên View có ID được chỉ ra
<code>android:layout_toLeftOf</code>	Nằm phía trái View có ID được chỉ ra
<code>android:layout_toRightOf</code>	Nằm phía phải View có ID được chỉ ra
<code>android:layout_alignBottom</code>	Căn thẳng cạnh dưới với cạnh dưới của View có ID được chỉ ra
<code>android:layout_alignLeft</code>	Căn thẳng cạnh trái với cạnh trái của View có ID được chỉ ra
<code>android:layout_alignRight</code>	Căn thẳng cạnh phải với cạnh phải của View có ID được chỉ ra
<code>android:layout_alignTop</code>	Căn thẳng cạnh trên với cạnh trên của View có ID được chỉ ra

6.3. Các `android:layout_margin` của View con

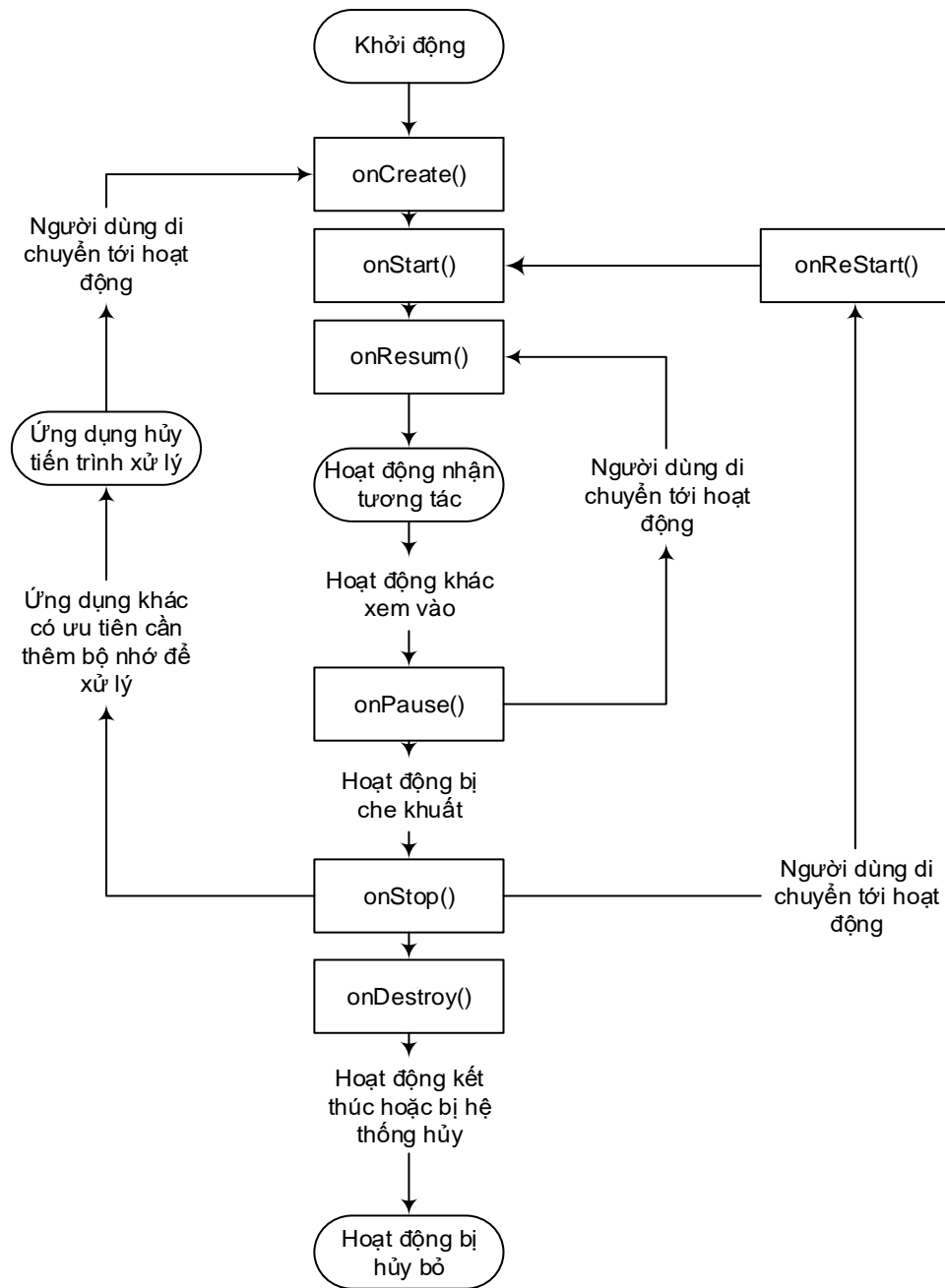
View con (left, top, right, bottom) nếu có mối liên hệ với View cha hoặc View con thì theo phía đó có thể thiết lập thêm thuộc tính về margin như:

`android:layout_marginLeft`, `android:layout_marginTop`,

`android:layout_marginRight`, `android:layout_marginBottom` để thiết lập khoảng cách của mối liên hệ đó.

7. Vòng đời ứng dụng Android.

Sơ đồ vòng đời ứng dụng Android



Bảng 2.1. Bảng mô tả ý nghĩa các trạng thái Activity

Sự kiện	Mô tả	Có thể bị hủy bỏ	Các sự kiện tiếp sau
onCreate()	Được gọi khi Activity tạo lần đầu. Công việc: Thiết lập thông số hiển thị, kết nối dữ liệu, ghép nối dữ liệu với đối tượng lưu trữ.	Không	onStart()



onStart	<p>Được gọi trước khi Activity hiển thị trên màn hình thiết bị.</p> <p>Lúc này tương tác người dùng – giao diện chưa được thiết lập.</p> <p>Sau onStart sẽ có 2 sự kiện tùy thuộc vào tình huống.</p> <ul style="list-style-type: none"> - Sự kiện theo sau onResume() nếu Activity hiển thị lên màn hình tương tác, - onStop() nếu Activity bị ẩn. 	Không	onResume() hoặc onStop()
onResume()	<p>Được gọi ngay trước khi Activity sẵn sàng nhận tương tác với người dùng.</p> <ul style="list-style-type: none"> - Activity nằm ở trên đỉnh ngăn xếp. - Tiếp sau là sự kiện onPause() nếu có 1 Activity khác chen ngang hiển thị. 	Không	onPause()
onPause()	<p>Được gọi khi hệ thống sắp bắt đầu một Activity khác.</p> <p>Mọi công việc thực hiện trong khâu này hệ thống sẽ thực hiện rất nhanh, vì hoạt động tiếp theo sẽ không được tiếp tục tới khi Activity kết thúc..</p> <ul style="list-style-type: none"> - Sự kiện onResume() được gọi nếu Activity chuẩn bị để gọi hiển thị - onStop() nếu không hiển thị với người dùng. 	Có	onResume() hoặc onStop()
onStop()	<p>Được gọi khi Activity không còn hiển thị với người dùng.</p> <ul style="list-style-type: none"> - Được theo sau hoặc bởi onStart() nếu Activity được gọi trở lại để tương tác với người dùng. - onDestroy() nếu Activity bị hủy. 	Có	onRestart() hoặc onDestroy()
onDestroy()	<p>Được gọi trước khi Activity bị hủy.</p> <p>Dùng hàm finish();</p>	Có	không có gì
onRestart()	<p>Được gọi sau khi hoạt động đã được dừng, ngay trước khi hoạt động được bắt đầu lại.</p> <p>Luôn được theo sau bởi onStart()</p>	Không	onStart()

Như vậy về mặt cơ bản một Activity có thể tồn tại ở 3 trạng thái: **tiếp tục, tạm dừng và dừng**.

Tiếp tục: Trạng thái này tồn tại khi Activity ở **trên đỉnh của ngăn xếp** và nhận tương tác từ người dùng. Trạng thái này đôi khi còn gọi là “đang chạy”.

Tạm dừng: Là khi Activity **bị che khuất 1 phần**, khi đó Activity không nhận được tương tác. Tuy nhiên trong tình huống khẩn cấp thì Activity có thể bị hệ thống “hủy bỏ”.

Dừng: là trạng thái khi Activity **mất tương tác**, không nhìn thấy hoàn toàn giao diện. Trong trường hợp này Activity có thể bị hệ thống thu hồi mọi tài nguyên trong bất cứ tình huống nào.

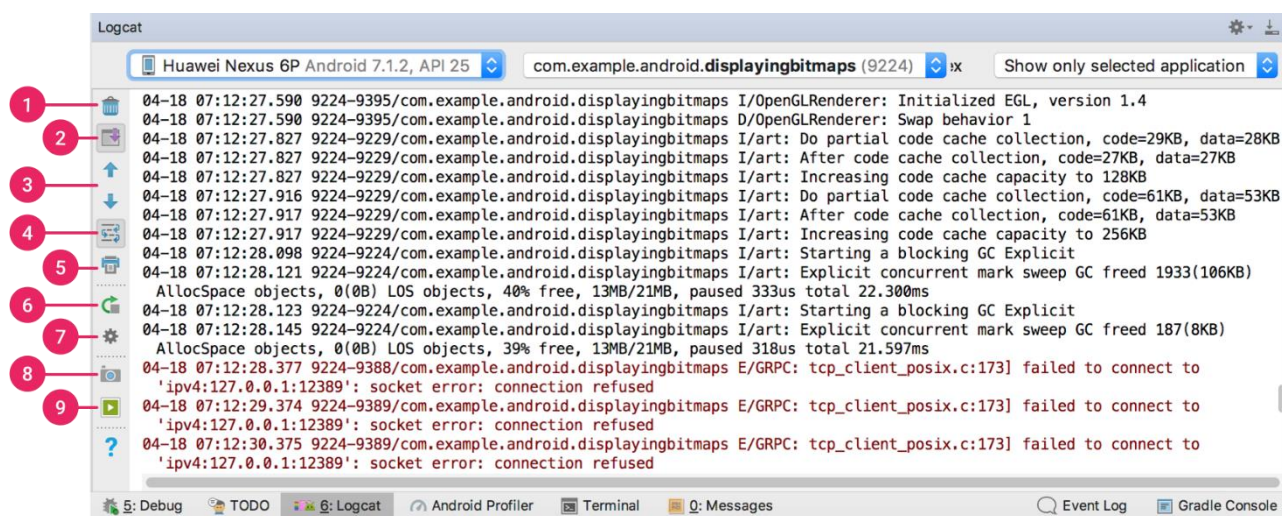
8. Viết và xem nhật ký bằng Logcat

Cửa sổ Logcat trong Android Studio cho thấy các thông điệp hệ thống, cũng như các thông điệp mà bạn bổ sung vào ứng dụng qua lớp Log. Cửa sổ này hiện thông điệp theo thời gian thực và lưu nhật ký để bạn có thể xem các thông điệp cũ.

8.1. Xem nhật ký ứng dụng

- Tạo và chạy ứng dụng trên thiết bị.
- View (Xem) > Tool Windows (Cửa sổ công cụ) > Logcat (hoặc nhấp vào biểu tượng Logcat trong thanh công cụ).

Cửa sổ Logcat cho thấy thông điệp nhật ký cho những ứng dụng được chọn trong danh sách thả xuống ở đầu cửa sổ, như ví dụ minh họa trong hình 1.



1. Xóa logcat : Nhấp để xóa nhật ký mà bạn thấy.

Học kết hợp



2. Cuộn xuống dưới cùng : Nhấp để chuyển đến cuối nhật ký và xem các thông điệp nhật ký mới nhất. Nếu sau đó bạn nhấp vào một dòng trong nhật ký, khung nhìn sẽ tạm dừng cuộn tại thời điểm đó.
3. Dấu vết ngăn xếp trên và Dấu vết ngăn xếp dưới : Nhấp để di chuyển giữa các dấu vết ngăn xếp trên và dưới trong nhật ký, chọn tên tệp tiếp theo (và xem số dòng tương ứng trong trình chỉnh sửa) xuất hiện trong các ngoại lệ đã in. Cách làm này tương tự như khi bạn nhấp vào một tên tệp trong nhật ký.
4. Dừng chế độ ngắt dòng mềm : Nhấp để bật tính năng cuộn xuống dòng và ngăn thao tác cuộn theo chiều ngang (mặc dù mọi chuỗi không thể ngắt vẫn cần cuộn ngang).
5. In : Nhấp để in các thông điệp logcat. Sau khi chọn các chế độ in ưu tiên trong hộp thoại vừa xuất hiện, bạn cũng có thể chọn lưu thành một tệp PDF.
6. Khởi động lại : Nhấp để xóa nhật ký rồi khởi động lại logcat. Không giống như nút Xóa logcat, tính năng này sẽ khôi phục và cho thấy các thông điệp nhật ký trước đó, vì vậy hữu ích nhất là trong trường hợp logcat không phản hồi và bạn không muốn mất thông điệp nhật ký.
7. Tiêu đề logcat : Nhấp để mở hộp thoại Configure Logcat Header (Định cấu hình tiêu đề logcat). Tại đây, bạn có thể tùy chỉnh giao diện của từng thông điệp logcat, chẳng hạn như việc có hiện ngày giờ hay không.
8. Chụp ảnh màn hình : Nhấp để chụp ảnh màn hình.
9. Ghi màn hình : Nhấp để quay video về thiết bị (tối đa 3 phút).

8.2. Soạn thông điệp nhật ký.

Lớp **Log** cho phép tạo thông điệp nhật ký xuất hiện trong logcat. Nhìn chung, khi tạo long ta cũng nên sử dụng các phương thức ghi nhật ký sau đây, liệt kê theo thứ tự ưu tiên từ cao nhất đến thấp nhất (hoặc ít đến nhiều chi tiết nhất):

Log.e(String, String) (lỗi)

Log.w(String, String) (nhắc nhở)

Log.i(String, String) (thông tin)

Log.d(String, String) (gỡ lỗi)

Log.v(String, String) (chi tiết)

Ví dụ:

- `private static final String TAG = "MyActivity";`
- `Log.i(TAG, "MyClass.getView()-item number " + position);`

Mức độ ưu tiên là một trong những giá trị sau:

V: Chi tiết (mức độ ưu tiên thấp nhất)

D: Gỡ lỗi

I: Thông tin

W: Cảnh báo

E: Lỗi

A: Khẳng định

8.3. Tìm kiếm thông điệp logcat

Để tìm kiếm các thông điệp đang xuất hiện trong logcat:

- Chọn Regex (Biểu thức chính quy) (không bắt buộc) nếu bạn muốn sử dụng một quy luật tìm kiếm dạng biểu thức chính quy.
- Nhập một trình tự ký tự vào trường tìm kiếm .
- Màn hình đầu ra logcat thay đổi tương ứng.
- Nhấn Enter để lưu chuỗi tìm kiếm trên trình đơn trong phiên này.
- Để tìm kiếm lại một nội dung, hãy chọn nội dung đó trên trình đơn tìm kiếm. Chọn hoặc bỏ chọn Regex (Biểu thức chính quy) nếu cần (chế độ cài đặt này không được lưu lại).

Ví dụ:

