

Lập trình trên thiết bị di động






NOTIFICATION & SERVICE

GV: Nguyễn Huy Cường




Email: nh.cuong@hutech.edu.vn

Nội dung

1. NOTIFICATION

-  Tổng quan về Notification
-  Thành phần của một Notification
-  Hành động trong một Notification
-  Notification channels
-  Tạo một Notification

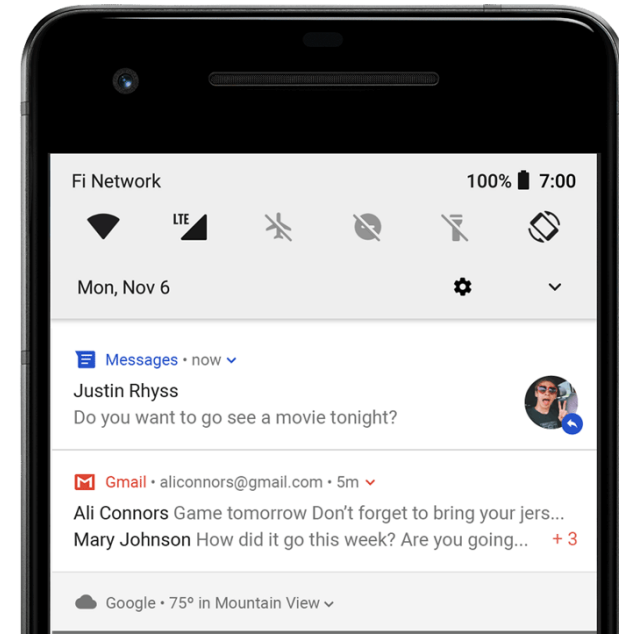
2. Service

-  Tổng quan
-  Phân loại
-  Vòng đời Service, độ ưu tiên

NOTIFICATION

Tổng quan Notification

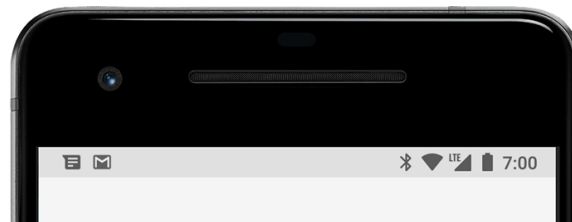
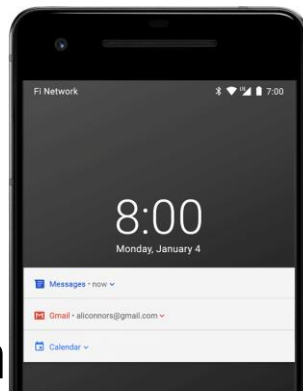
- **Notification** là một tin nhắn mà Android hiển thị **bên ngoài giao diện người dùng** của ứng dụng để:
 - ❑ Cung cấp cho người dùng lời nhắc
 - ❑ Thông tin liên lạc từ những người khác hoặc thông tin kịp thời khác từ ứng dụng.
 - ❑ Người dùng có thể nhấn vào thông báo đó để mở ứng dụng của bạn hoặc thực hiện hành động ngay trong thông báo.
- **Notification** có thể hiển thị cho người dùng ở nhiều vị trí và định dạng khác nhau



➤ Trung tâm thông báo
/ Ngăn xếp thông báo

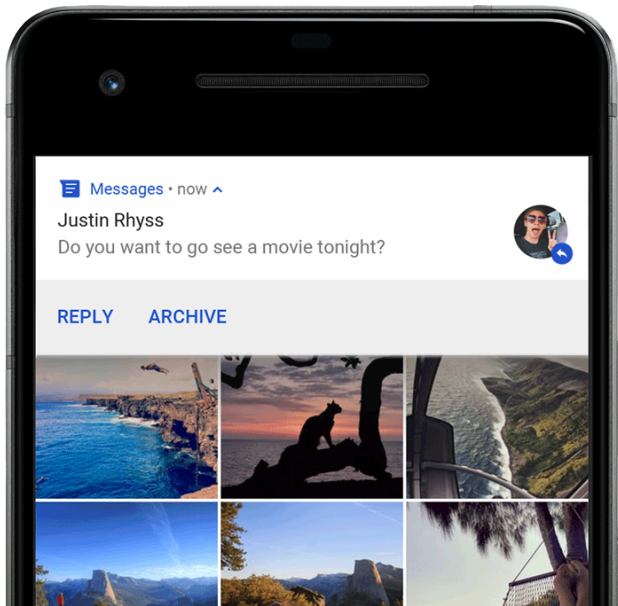
➤ Statusbar

➤ Màn hình khóa

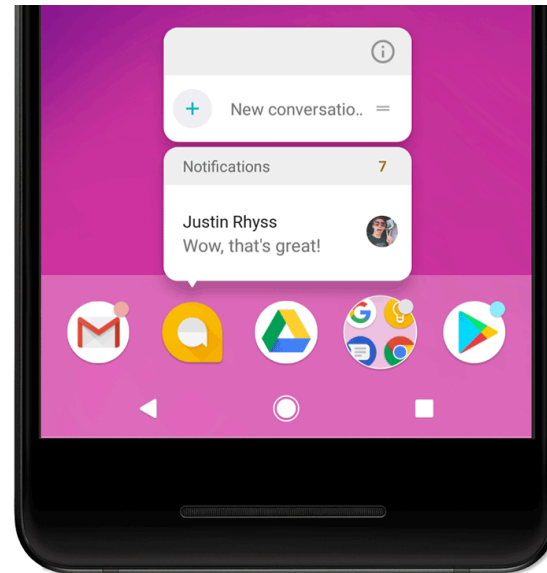


Tổng quan Notification

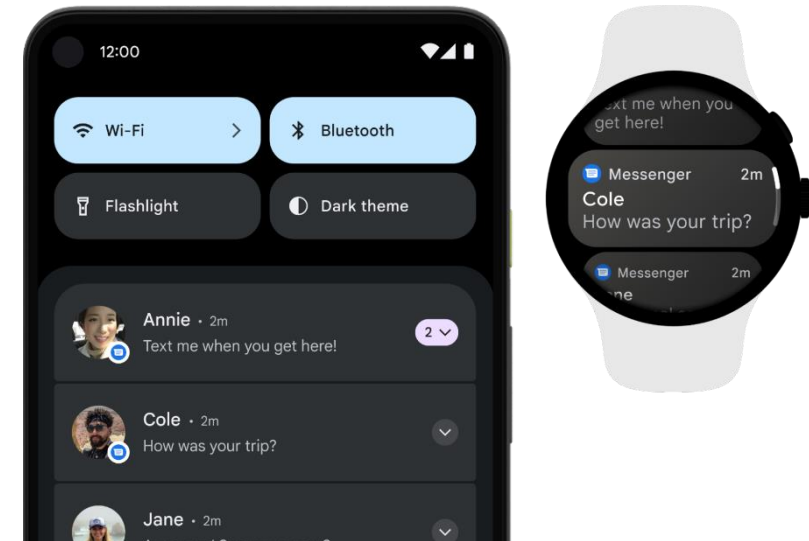
- Notification có thể hiển thị cho người dùng ở nhiều vị trí và định dạng khác nhau



➤ Cửa sổ thông báo quan trọng
/ Heads-up notification



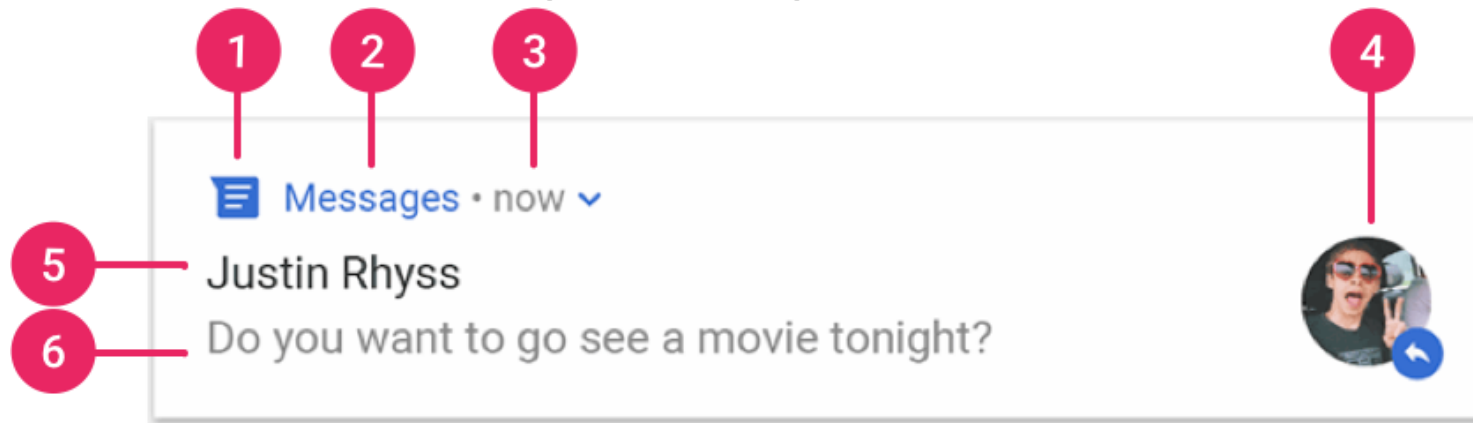
➤ App icon badge



➤ Thiết bị đeo

Thành phần của một notification

- Notification có thể hiển thị cho người dùng ở nhiều vị trí và định dạng khác nhau



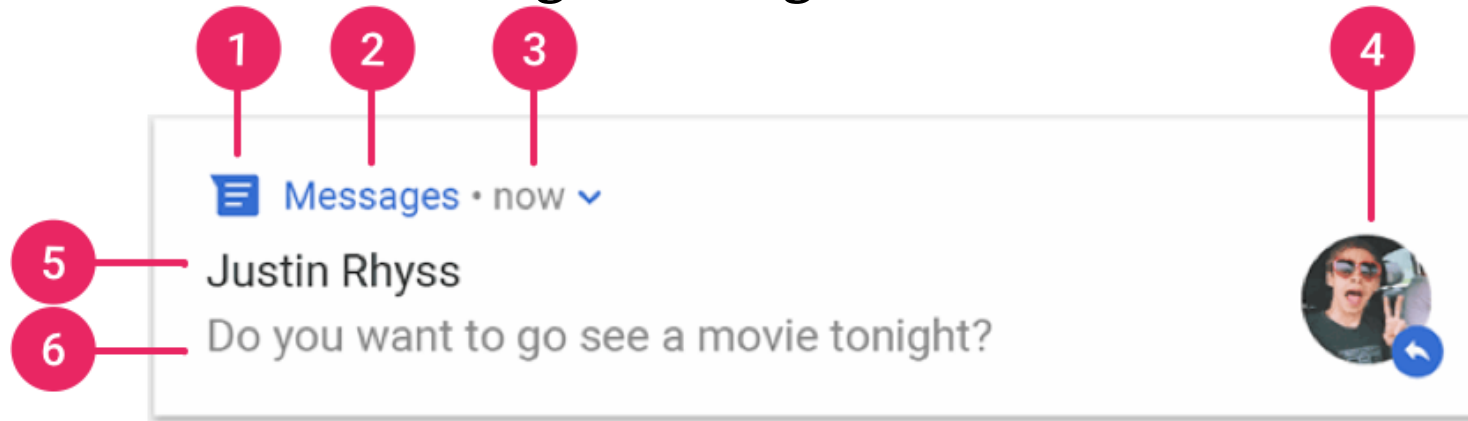
(1) Small icon: Đây là phần bắt buộc, sử dụng thuộc tính **setSmallIcon()**

(2) Tên ứng dụng: Do hệ thống cung cấp.

(3) Time stamp (Dấu thời gian): Thông tin được cung cấp bởi hệ thống, nhưng có thể ghi đè bằng cách sử dụng **setWhen()** hoặc ẩn đi bằng cách sử dụng **setShowWhen(false)**.

Thành phần của một notification

- Notification có thể hiển thị cho người dùng ở nhiều vị trí và định dạng khác nhau

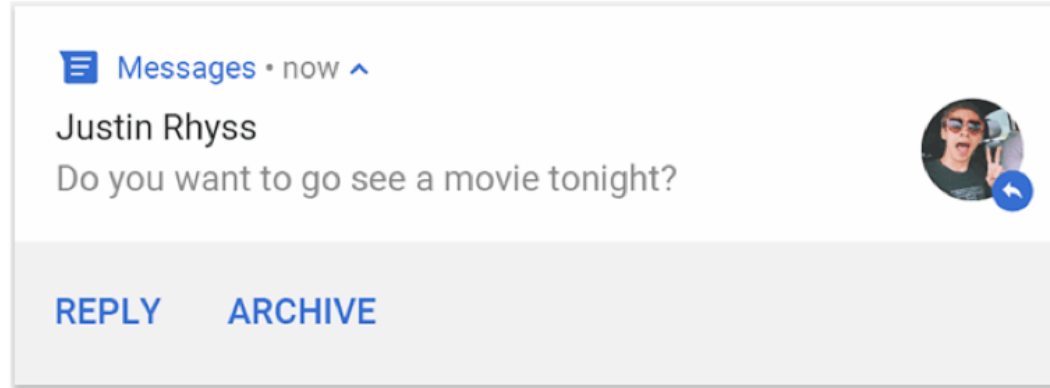


(4) Large icon: Phần này không bắt buộc (thường dùng cho ảnh liên hệ, không dùng cho biểu tượng ứng dụng), sử dụng thuộc tính **setLargeIcon()**.

(5) Tiêu đề: Phần này không bắt buộc, sử dụng thuộc tính **setContentTitle()**.

(6) Nội dung: Phần này không bắt buộc, sử dụng thuộc tính **setContentText()**.

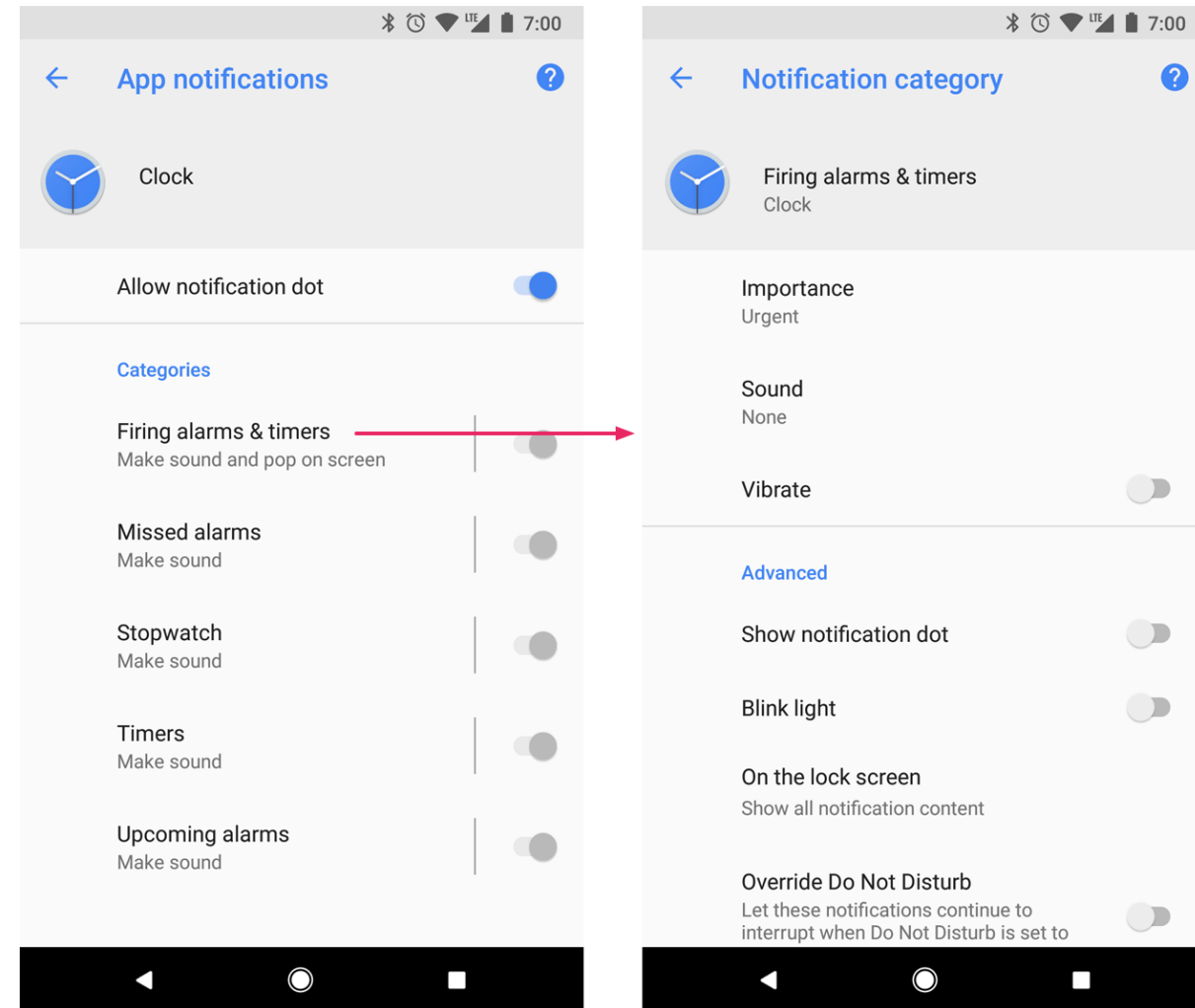
Hành động trong một notification



- Mặc dù không bắt buộc, nhưng khi người dùng nhấn vào thông báo, sẽ hiển thị một số nút thực hiện một số hành động. Ngoài hành động mặc định trong thông báo, ta có thể thêm các nút hành động khác để hoàn tất một nhiệm vụ liên quan đến ứng dụng.
- Kể từ Android 7.0 (API cấp 24), có thể thêm một hành động để trả lời tin nhắn hoặc để nhập văn bản khác ngay trong thông báo.
- Kể từ Android 10 (API cấp 29), hệ thống có thể tự động tạo các nút hành động bằng các hành động dựa trên ý định đề xuất.

Notification channels

- Từ Android 8.0 (API cấp 26), phải chỉ định channels cho tất cả thông báo thì thông báo mới xuất hiện. Bằng cách phân loại thông báo vào các channels, người dùng có thể tắt các channels thông báo cụ thể của ứng dụng (thay vì tắt tất cả các thông báo), đồng thời có thể kiểm soát các tùy chọn hình ảnh và âm thanh cho từng channels. Ngoài ra, người dùng còn có thể nhấn và giữ một thông báo để thay đổi cách hoạt động của channels liên kết.





Tạo một Notification

ref:

<https://developer.android.com/develop/ui/views/notifications/build-notification>

Bước 1: Trong Android Manifest khai báo quyền Post Notification cho ứng dụng

Bước 2: Tạo một notification

```
private static String CHANNEL_ID = "HUTECH";
private static final int NOTIFICATION_ID = 1;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    createNotificationChannel();

    // Create the notification
    NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL_ID)
        .setSmallIcon(R.drawable.ic_launcher_background)
        .setContentTitle("My Notification")
        .setContentText("This is a demo notification!")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);

    // Show the notification
    NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);
    notificationManager.notify(NOTIFICATION_ID, builder.build());
}

// Create the notification channel for devices running Android Oreo (API 26) and above.
private void createNotificationChannel() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = "My Channel";
        String description = "Channel for my notifications";
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
        channel.setDescription(description);

        NotificationManager notificationManager = getSystemService(NotificationManager.class);
        if (notificationManager != null) {
            notificationManager.createNotificationChannel(channel);
        }
    }
}
```

SERVICE

Tổng quan

- Service không cung cấp giao diện cho người dùng
- Service chạy ngầm để thực hiện các công việc như nghe nhạc, thực hiện ghi và đọc file, hoặc tương tác với Content Provider
- Service là thành phần ứng dụng có thể thực hiện các thao tác đòi hỏi tốn nhiều thời gian và tài nguyên

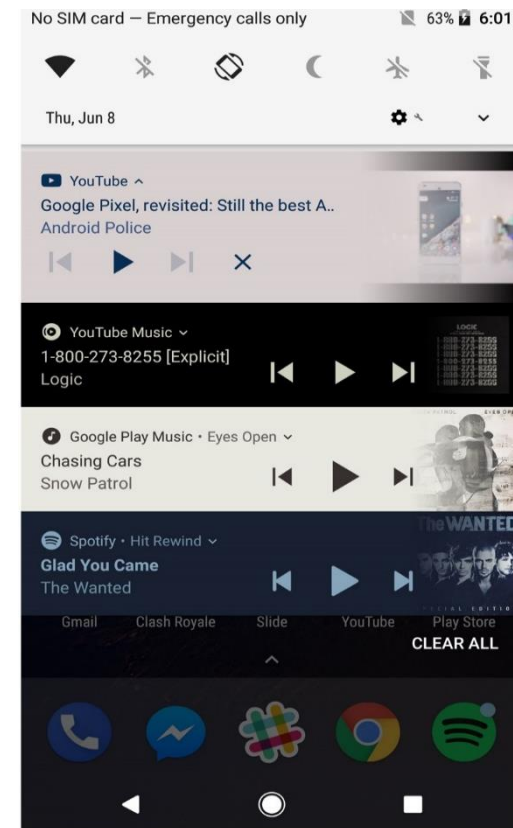


Phân loại service

- **Foreground Service:** Là Service mà người dùng biết là Service đang chạy
 - ❑ Hệ thống sẽ không Kill Service khi bộ nhớ xuống thấp
 - ❑ Phải cung cấp một Notification trên Status bar thể hiện Service đang chạy trừ khi Service bị dừng hoặc bị hủy từ Foreground

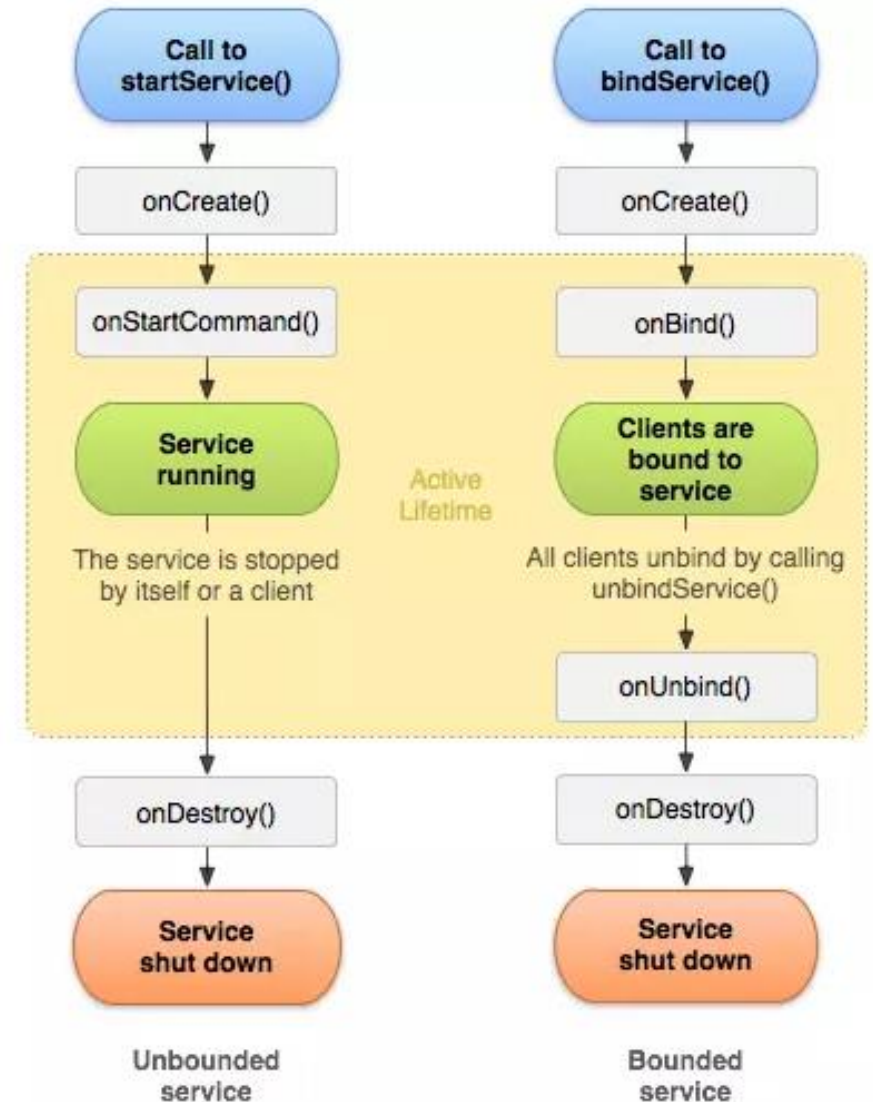
Ví dụ: Nghe nhạc,

- **Background Service:** Là Service chạy mà người dùng không được là Service đang chạy và không có tương tác
- VD: Báo thức, nhận tin nhắn đến, nhận cuộc gọi đến



Vòng đời của service

- Unbound Service (Started Service): service không ràng buộc
- Bound Service: service ràng buộc



Started Service

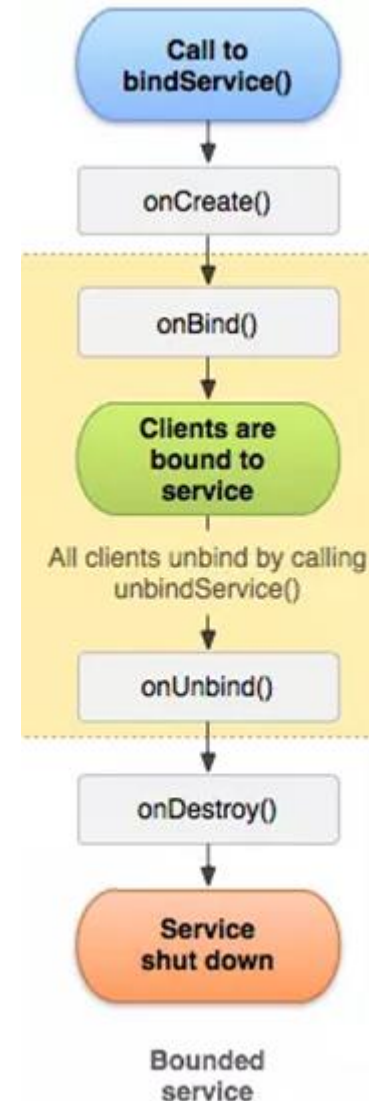
- Service đã khởi tạo (**Started Service**) là service được khởi tạo khi một thành phần ứng dụng (ví dụ như Activity) khởi tạo Service bằng lời gọi phương thức **startService()**
- Khi khởi tạo xong, Service có thể **chạy nền vô hạn** kể cả khi thành phần khởi tạo Service (ví dụ như Activity khởi tạo Service này) đã bị hủy
- Thông thường, Service đã khởi tạo thực hiện một thao tác đơn giản và không trả lại một kết quả cụ thể.

Ví dụ: download và upload một file trên mạng. Khi Service hoàn thành nhiệm vụ việc download và upload, Service sẽ tự động dừng lại.



Bound Service

- Service ràng buộc (bound service) khi một thành phần ứng dụng ràng buộc Service bằng lời gọi **bindService()**
- Service ràng buộc có giao diện client-server cho phép các thành phần ứng dụng tương tác với Service, gửi yêu cầu, nhận kết quả
- Khi tất cả thành phần không ràng buộc Service nữa, Service sẽ bị hủy
- Một Service có thể hoạt động theo cả 2 cách:
 - ❑ được khởi tạo (chạy vô hạn) **onStartCommand()**. Để dừng service gọi phương thức **stopSelf()** hoặc **stopService()**.
 - ❑ cho phép ràng buộc (binding) **onBind()** .
- có thể khai báo Service là private trong file AndroidManifest.xml (khai báo **android:exported="false"** cho Service trong AndroidManifest)



Độ ưu tiên của các loại service

- Độ ưu tiên theo khó bị tiêu hủy nhất: Bound Service > Foreground Service > Background Service.



Thực hiện theo hướng dẫn Labo8

WORKMANAGER

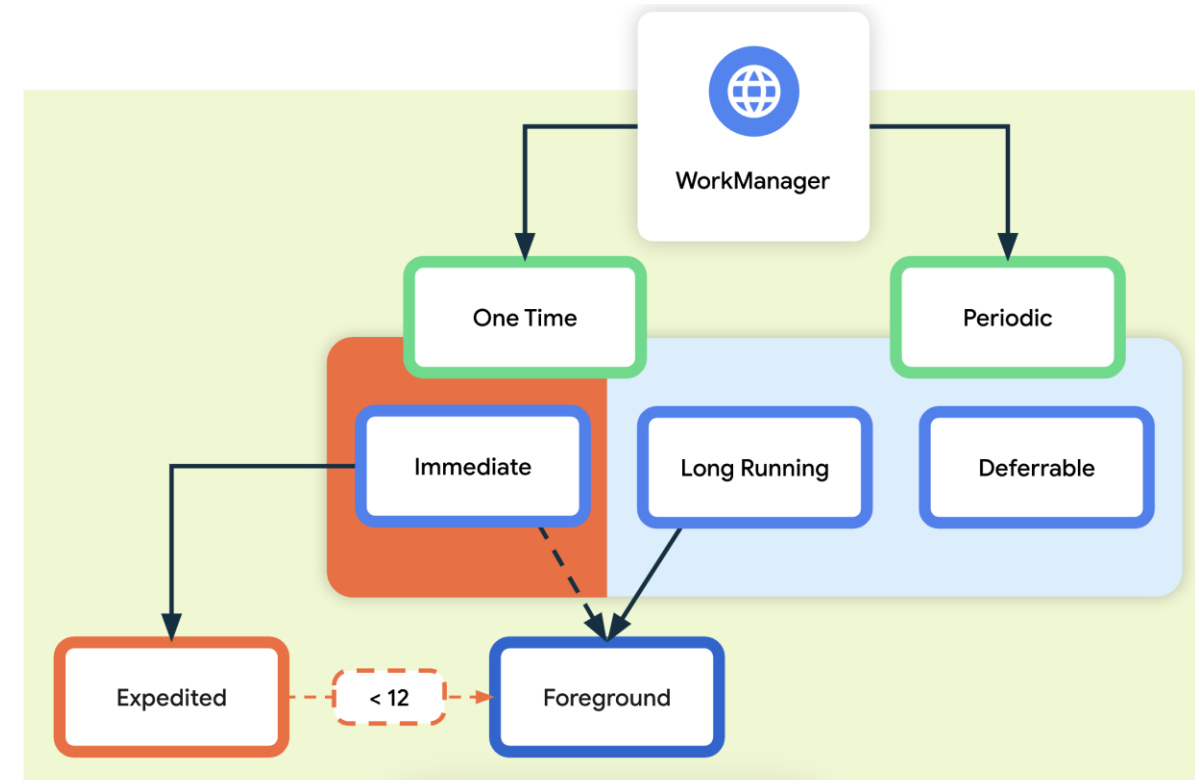
Tổng quan

- **WorkManager** là giải pháp được đề xuất cho công việc liên tục. Khả năng làm việc liên tục khi được lên lịch thông qua các lần khởi động lại ứng dụng và khởi động lại hệ thống. Vì hầu hết quá trình xử lý nền đều diễn ra hiệu quả nhất thông qua khả năng làm việc liên tục, **WorkManager** là API chính nên dùng để xử lý nền.



Các loại công việc liên tục

- **WorkManager** xử lý ba loại công việc liên tục:
 - ❑ **Ngay lập tức (Immediate):** Những tác vụ phải bắt đầu ngay và hoàn thành sớm. Có thể được ưu tiên.
 - ❑ **Lâu dài (Long Running):** Những tác vụ có thể chạy lâu hơn và có thể kéo dài hơn 10'.
 - ❑ **Có thể trì hoãn (Deferrable):** Những tác vụ đã lên lịch bắt đầu sau và có thể chạy định kỳ.





WORKMANAGER