**BÁO CÁO CUỐI NGÀY**

Họ tên : Nguyễn Quốc Bảo

Môn: Android Advance

Ngày: 19/01/2018

Kế hoạch: Lesson 5 – Broadcast Receiver, Service and Notification

1. Broadcast Receiver

* Đã bao giờ bạn tự hỏi hệ thống Android làm cách nào mà có thể nhận biết được tất cả những sự thay đổi của hê thống như lắng nghe tin nhắn đến, hành động rút - cắm sạc, hay hành động bật tắt mạng,... đó chính xác là hệ thống sử dụng Broadcast Receiver để làm
* Là một trong 4 component lớn trong Android gồm Activity, Services, Content providers, Broadcast receriver
* Với mục đích là lắng nghe các sự kiện, trang thái của hệ thông 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.
* Điểm nổi bật của Broadcast Receiver là nó 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
* Có 2 cách để sử dụng Broadcast Receiver
* Đăng ký trong AndroidManifest.xml (Tĩnh): ưu điểm của cách đăng ký này là khi bạn tắt một ứng dụng thì nó vẫn lắng nghe .

+ Cần phải khai báo hành động trong AndroidManifest.xml và cần phải xin cấp quyền. VD

<receiver android:name=".NetworkChangeReceiver" >

            <intent-filter>

                <action android:name="android.net.conn.CONNECTIVITY\_CHANGE" />

            </intent-filter>

</receiver>

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>

+ Đăng ký bằng Java code (Động) : cần sử dụng hai methods registerReceiver và unregisterReceiver

* Để lắng nghe kết quả cần override method onReceive(Context, Itent)

1. Service

* Là một thành phần luôn luôn chạy dưới lớp background của hệ thống và không cần sự tương tác của người dùng . Ví dụ : Khi nghe một bài hát , khi người dùng thoát khỏi ứng dụng (chuyển qua ứng dụng khác) thì bài nhạc đó vẫn được chạy
* Một service có hai trạng thái

+ Trạng thái Started: Một service được bắt đầu khi ứng dụng được chạy và gọi phương thức startService(). Nó sẽ chạy vô thời hạn nếu như không hủy nó . (phương thức stopService() or stopSelf() )

+ Trạng thái Bound: khi ứng dụng gọi phương thức bindService . Liên kết với service theo dạng client – service, cho phép components tương tác với service như gửi yêu cầu , nhận kết quả

1. Notification

* Notification là một thông báo hiển thị thông tin đến cho người dùng bên ngoài ứng dụng. Thông thường sẽ xuất hiện bến trên cùng màn hình. Để xem chi tiết thông báo, bạn cần kéo Notification drawer xuống. Trong các dự án thực tế, việc sử dụng đến Notification là không thể thiếu
* Notification Manager: Notification Manager là lớp quản lý các Notification. Nó cho phép đặt các Notification vào thanh tiêu đề của ứng dụng. Người dùng có thể mở Notification và tương tác với Notification để mở Activity tương ứng
* Sử dụng method notify() của đối tượng NotificationManager để đẩy thông báo cho người dùng bằng cách truyền vào hai tham số : Id của notification và đối tượng Notification. **Tham số 0 hiểu là ID của đối tượng noti**
* Trong lớp Notification có một lớp Builder, lớp này dùng để tạo giao diện cho 1 thông báo. Ta sử dụng lớp PendingIntent để bắt sự kiện khi kích vào thông báo
* Bạn có thể xóa một **Notification** nhất định bằng lệnh notiManager.cancel(notiID); hoặc xóa tất cả notiManager.cancelAll();
* Dưới đây là một số phương thức quan trọng và thường được sử dụng của lớp NotificationCompat.Builder:
* **Notification build()** - kết hợp tất cả tùy chọn mà đã được thiết lập và trả về một đối tượng Notification mới.
* **NotificationCompat.Builder setAutoCancel(boolean autoCancel)** - thiết lập flag này để thông báo tự động bị hủy khi người dùng click nó trong panel
* **NotificationCompat.Builder setContent (RemoteViews views)** - cung cấp một Custom RemoteViews để sử dụng thay cho RemoteView chuẩn.
* **NotificationCompat.Builder setContentInfo(CharSequence info)** - thiết lập khổ text lớn tại cạnh phải của Notification.
* **NotificationCompat.Builder setContentIntent(PendingIntent intent)** - cung cấp một PendingIntent để gửi khi thông báo được click.
* **NotificationCompat.Builder setContentText(CharSequence text)** - thiết lập text hàng thứ hai của thông báo trong một Notification chuẩn.
* **NotificationCompat.Builder setContentTitle(CharSequence title)** - thiết lập text hàng thứ nhất của thông báo, trong một Notification chuẩn.
* **NotificationCompat.Builder setDefaults (int defaults)** - Thiết lập các tùy chọn thông báo mặc định mà sẽ được sử dụng.
* **NotificationCompat.Builder setLargeIcon (Bitmap icon)** - Thiết lập icon lớn mà được hiển thị trong thông báo.
* **NotificationCompat.Builder setNumber (int number)** - Thiết lập large number tại bên phải của Notification.
* **NotificationCompat.Builder setOngoing (boolean ongoing)** - Thiết lập xem có hay không đây là một thông báo phát ra từ hệ thống ra ngoài.
* **NotificationCompat.Builder setSmallIcon (int icon)** - Thiết lập small icon để sử dụng trong layout của thông báo.
* **NotificationCompat.Builder setStyle (NotificationCompat.Style style)** -
* Thêm một style để được áp dụng tại builder time.
* **NotificationCompat.Builder setTicker (CharSequence tickerText)** - Thiết lập text mà được hiển thị trong status bar khi thông báo đến lần đầu.
* **NotificationCompat.Builder setVibrate (long[] pattern)** - Thiết lập vibration pattern để sử dụng.
* **NotificationCompat.Builder setWhen (long when)** - Thiết lập thời gian mà sự kiện xảy ra. Các thông báo trong panel được xếp thứ tự theo thời gian này.