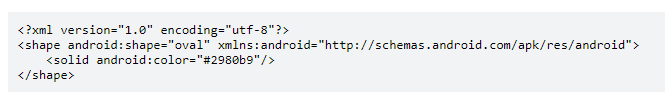
Drawable

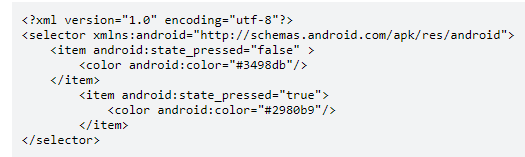
* Bitmap drawable



* vector drawable
* shape drawable: custom background cho view



* State list drawable: sử dụng trong button, kiểu như khi click sẽ đổi màu

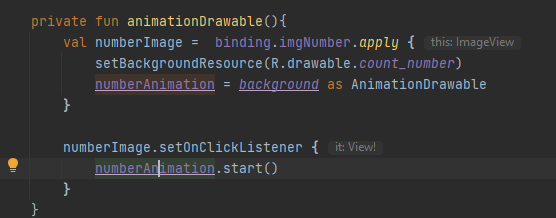


Animation

1. Drawable Animation

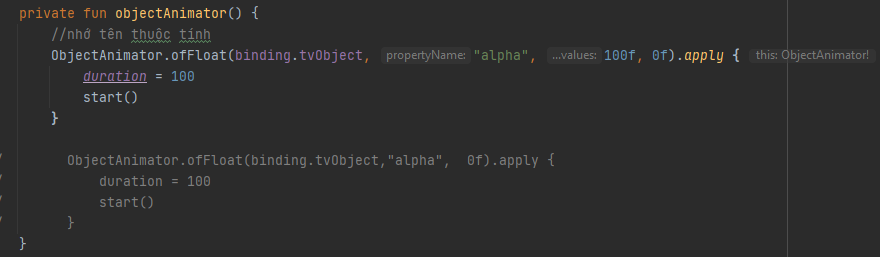
* Vector Drawable: kéo ảnh mà chất lượng ảnh không bị thay đổi
* oneshot: true sẽ quay 1 lần, false lặp lại

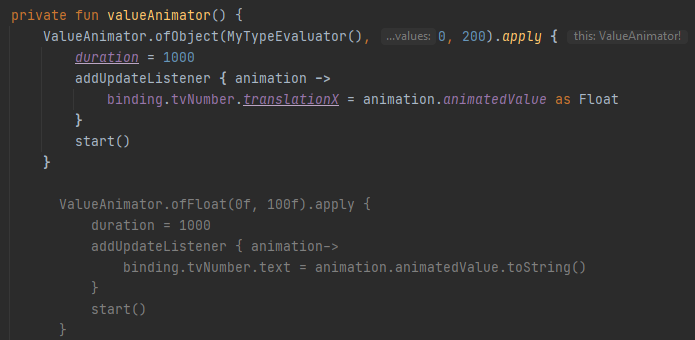




1. Property animation: Object Animator, Value Animator,Animator set

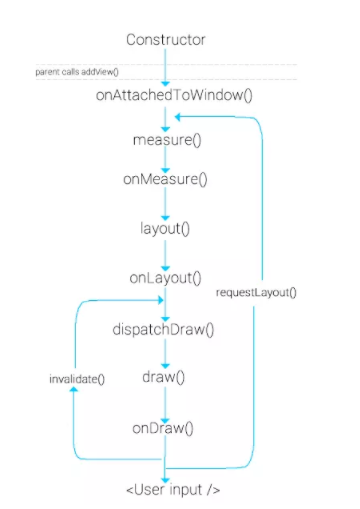
* Duration: thời gian chạy animation
* Time interpolation : chỉ định cách tính giá trị trong thời gian chạy
* Repeat count and behavior: chỉ định có lặp lại ko, số lần lặp lại, chỉ định xem phát ngược lại hay không
* Type Evaluator: set thuộc tính di chuyển(dùng trong value animator)
* Animator sets: nhóm các animation
* Objectanimator: là lớp con của Value Animator, tạo hiệu ứng cho 1 thuộc tính theo tên của nó
* Value Animator: chỉ định giá trị đầu cuối kiểu float, int hoặ ofObject
* Frame refresh delay:
* Animate sự dụng ValueAnimator:



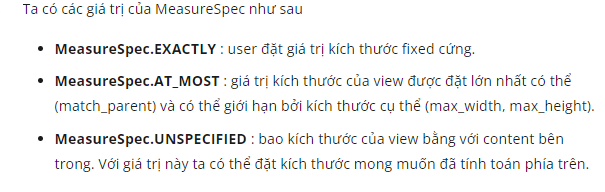




Custom View



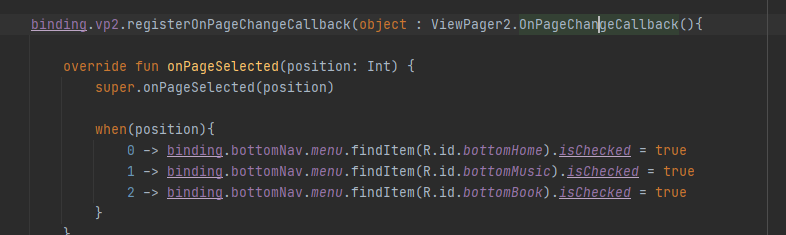
* Khi constructor được khởi tạo sẽ gọi attach to window để attach view vào window
* onMeasure: Tính tính toán kích thước của view so với cha, mesure được gọi rất nhiều lần để chia tỉ lệ. Lấy measurespec của custom view(size và mode của width, height) của custom view



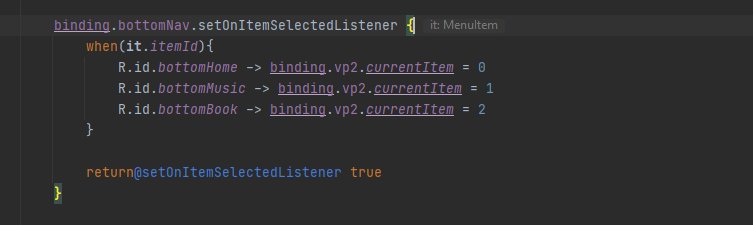
* onLayout: thực hiện việc chỉ định kích thước các view con bên trong custom view
* onDraw: vẽ liên lục, khi có bất kỳ sự thay đổi nào view sẽ được vẽ lại
* inValidate: được gọi để vẽ lại các view đơn giản
* request layout(): yêu cầu tính toán lại toàn bộ kích thước

Bottom Navigation

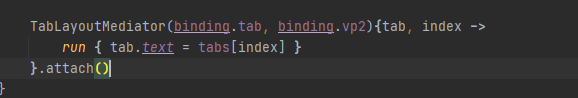
Vuốt page:



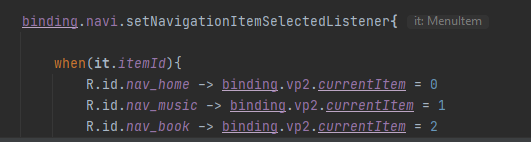
Click chọn page trên bottom:



Setup tab:



Navigation



Database

1. Internal storage (chụp ảnh lưu file vào đường dẫn nào đấy

* Là bộ nhớ trong của thiết bị
* Đảm bảo chỉ có ứng dụng truy xuất được dữ liệu mà các ứng dụng khác không thể, kể cả người dung
* Uninstall App, clear App Data -> bị xóa

1. External storage

* Bộ nhớ ngoài, có thể chia sẻ dữ liệu(SDCard, bộ nhớ trong những có thể chỉnh sửa bởi người dung)
* Uninstall App, clear App -> ko bị ảnh hưởng

1. Shared preference

* Là 1 interface cho phép lưu trữ dữ liệu kiểu key-value
* Lưu trong ứng dụng và thuộc internal storage
* Không lưu đc object (lưu đc nhưng phải convert về chuỗi json)
* Apply() và commit(): apply() lưu bất đồng bộ commit thì đồng bộ, apply trả về kqua còn commit không trả về kết quả. Thường sẽ dùng apply()

1. Database

<https://developer.android.com/training/data-storage/room/relationships>

<https://viblo.asia/p/android-database-relations-trong-room-gDVK2mEn5Lj>

<https://medium.com/mindorks/android-architecture-components-room-viewmodel-and-livedata-50611793e4a9>

https://medium.com/android-news/android-architecture-components-room-relationships-bf473510c14a

//database relationships

https://medium.com/androiddevelopers/database-relations-with-room-544ab95e4542

\_binding và binding get(): backing properties

SharePreference

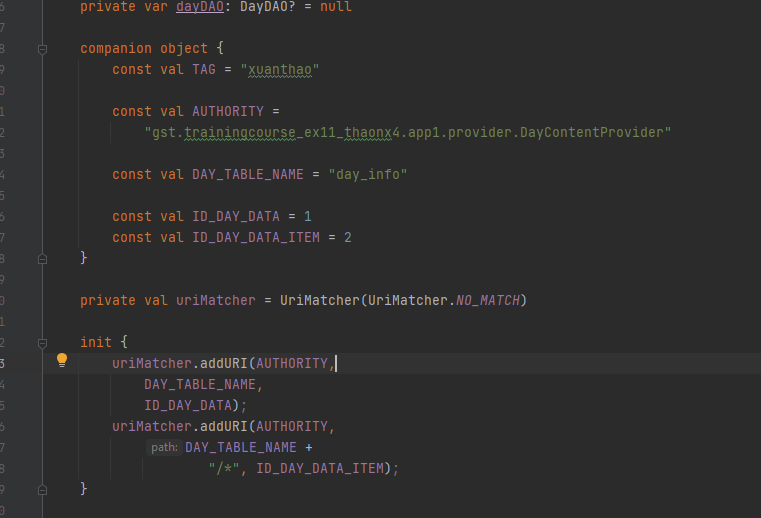
* SharedPreference là 1 thành phần giúp lưu trữ dữ liệu theo kiểu key, value trong xml.
* Nó thường lưu trữ setting, account info, high score...vv
* SharePreference thường để lưu dữ liệu nhỏ. Và không lưu được trực tiếp object. Nếu muốn lưu object ta sẽ lưu từng attribute của object đó.

DiffUntil

* ViewHolder : là 1 view được thiết lập theo ý của bạn
* Item Callback : đc gọi để kiểm tra xem 2 đối tượng có đại diện cho cùng 1 mục hay ko
* Được gọi để kiểm tra xem 2 đối tượng có cùng 1 mục hay không. Nếu đúng 2 mục đại diện cho cùng 1 đối tượng hoặc sai nếu chúng khác nhau
* are Items The Same : kiểm tra xem 2 mục có cùng dữ liệu hay không. Để phát hiện sự thay đổi nội dung của 1 mục
* are Content The Same : khi areItemTheSame trả về true cho 2 mục và areContentTheSame(T,T) trả về false cho chúng, PT này đc gọi để nhận sự thay đổi

Content Provider

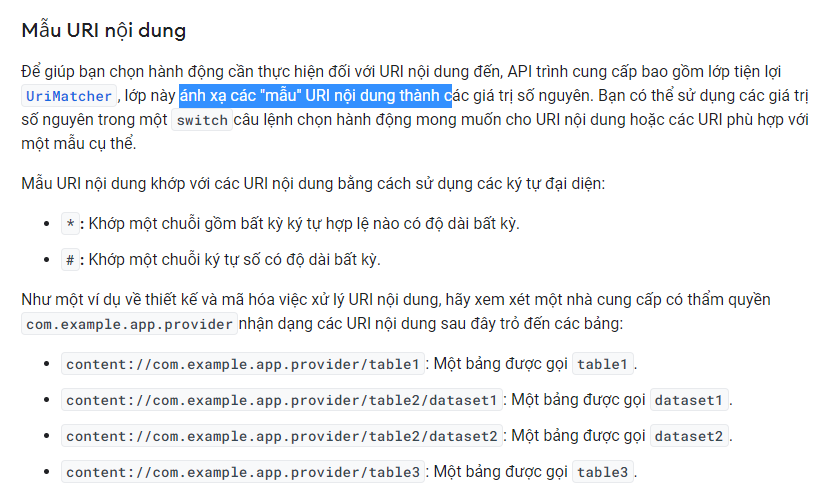
* Nó hoạt động như 1 interface cho phép bạn lưu trữ và truy xuất dữ liệu, có thể chia sẻ dữ liệu giữa các ứng dụng khác nhau. Bất kì ứng dụng nào được cấp quyền đều có thể thêm, xóa, sửa và truy xuất dữ liệu của ứng dụng khác
* Content Resolver : gửi yêu cầu đến content provider, cung cấp các phương thức query(), insert(), update(), delete() để truy cập dữ liệu từ Content Provider
* CursorLoader: truy vấn không đồng bộ



<provider  
 android:name="com.nxt.tên project.tên provider"  
 android:authorities=" com.nxt.tên project.tên provider "  
 android:exported="true"  
 android:multiprocess="true" />

CONTENT\_URI : "content://$AUTHORITY/DATABASE\_NAME"

* UriMatcher : chọn hành động thực hiện với Uri
* AUTHORITY: Quyền truy cập



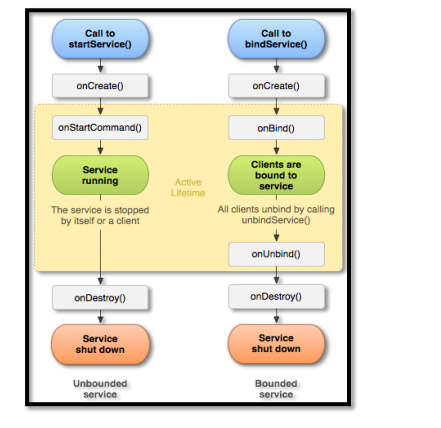
SERVICE

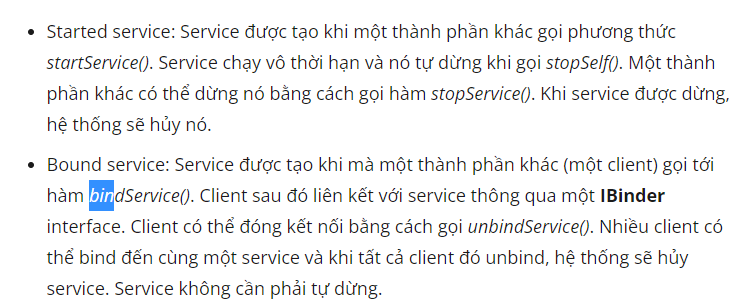
3 LOẠI : Foreground Service(unbound service) : có thể nhìn thấy nó đang chạy ngay cả khi user không tương tác với ứng dụng , phải thực hiện 1 cái notification để biết nó đang chạy(vd trình phát nhạc)

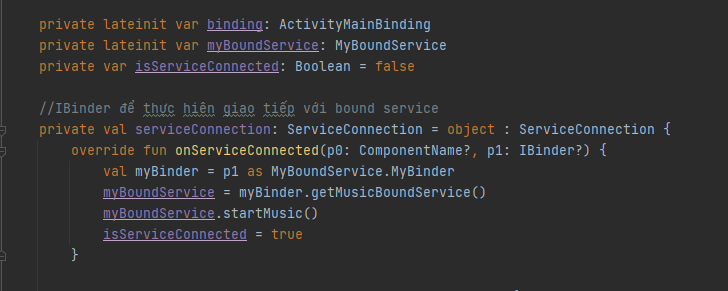
BackGround Service(unbound Service) : không trực tiếp nhìn thấy , không cần notification, Tự động stop sau 1 khoảng thời gian khi app ở chế độ background (để giảm tiêu tốn bin)

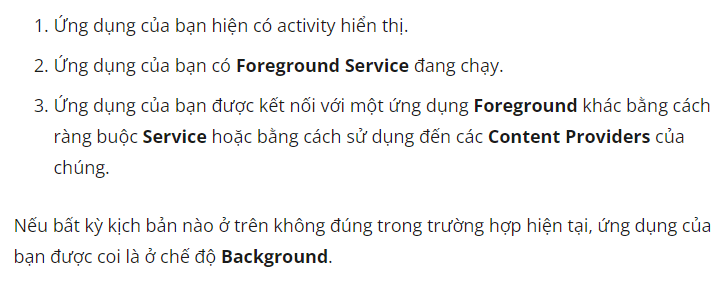
BOUND SERVICE : Hành động ở service sẽ được gọi qua phần liên kết ràng buộc, khác ở foreground là khi activity bị destroy thì nó cũng dừng vì bị ràng buộc ,

\*Có thể sử dụng nhiều service trong ứng dụng , độ ưu tiên : Bound > ForeGround > BackGround

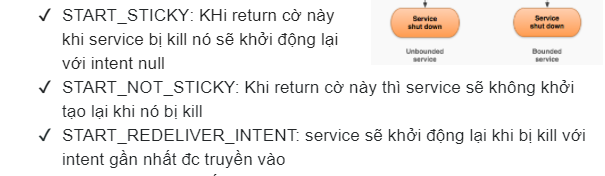








Background Service(JOBSERVICE)từ android O trở đi chạy đc ở chế độ background nhưng quá 1 phút sẽ tự động stop(khi thực hiện service hoàn thành có sendbroadcast về đc activity)



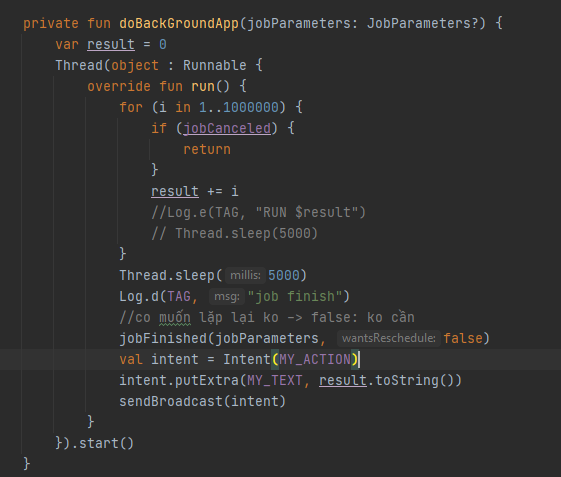
Service và Intent Service

* Intent service là 1 subclass của service. Nó sẽ tạo ra 1 worker thread để xử lý các request từ client gửi đến các request được đưa vào 1 hàm đợi onHandleIntent() để xử lý, khi ko còn request thì service sẽ bị hủy
* Khác nhau:

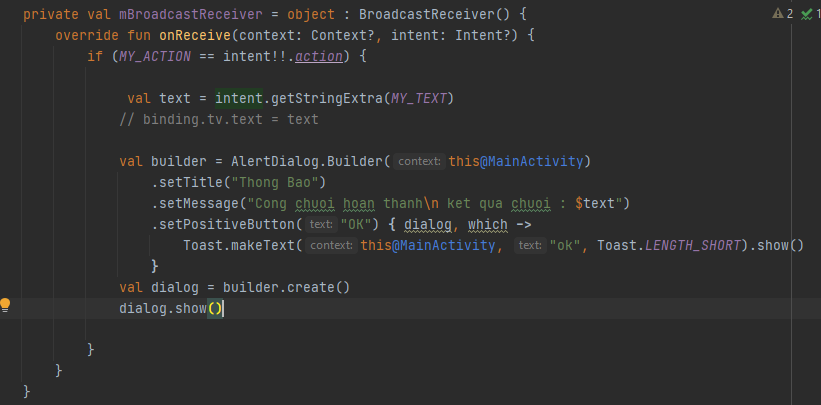
1. luồng thực thi: service chạy trên main thread, các request từ client được xử lý đồng thời trên 1 luồng. Intent service xử lý trên luồng worker thread do nó tạo ra
2. cách xử lý request: Service xử lý đồng đời còn Intent service xử lý lần lượt request trong hàm đợi bằng onHandleIntent
3. về cách stop service: service được dừng bằng cách gọi stopService() hoặc stopSelf() còn với Intent Service nó sẽ tự động dừng

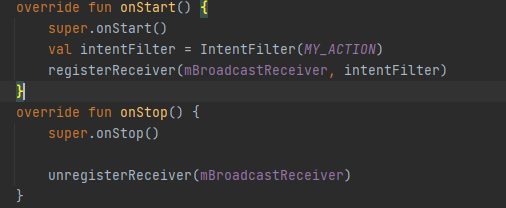
Broadcast Receiver

* B1: sendBroadcast bên gửi

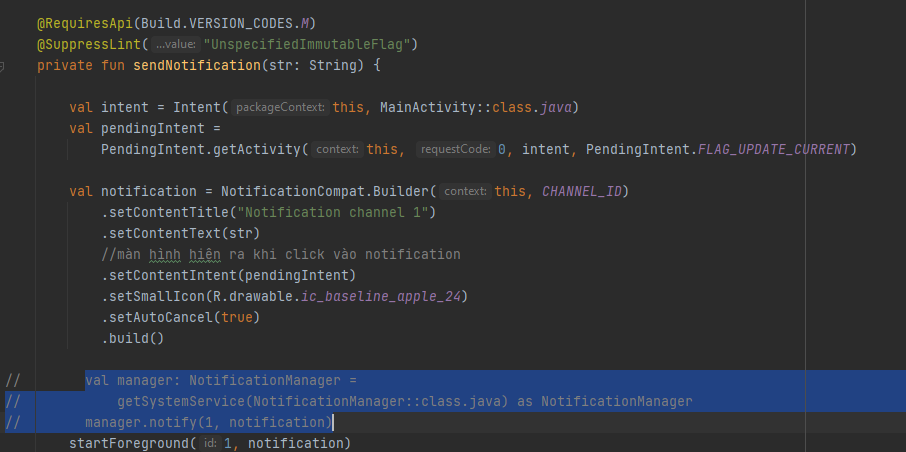


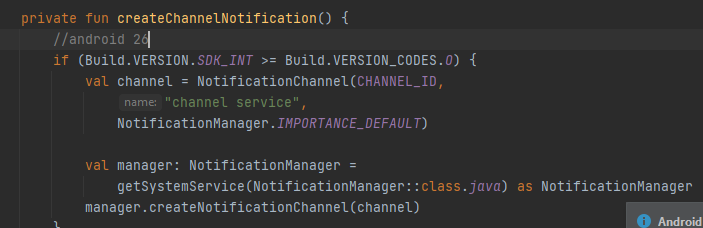
* B2: đăng ký nhận trong onStart() và hủy trong onStop()





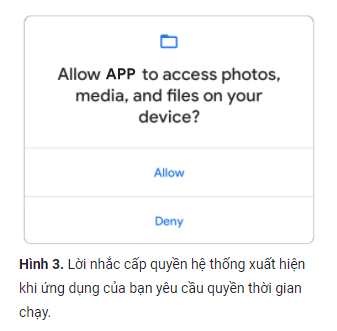
Notification





Permission

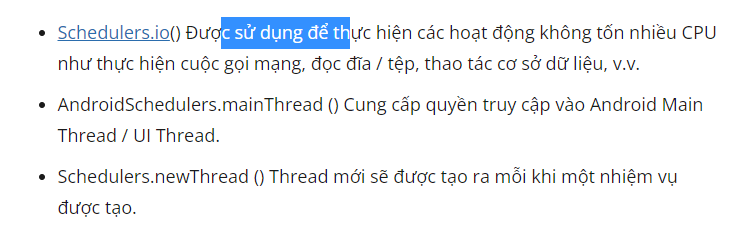
1. Normal Permission: đc cấp tại thời thiểm cài đặt và ko thể thu hồi(Internet,bluetooth)
2. Run time permission: 3 bước thêm quyền tai manifest, kiểm tra quyền và yêu cầu quyền, còn gọi là quyền nguy hiểm, hạn chế ảnh hưởng đến hệ thống và ứng dụng khác

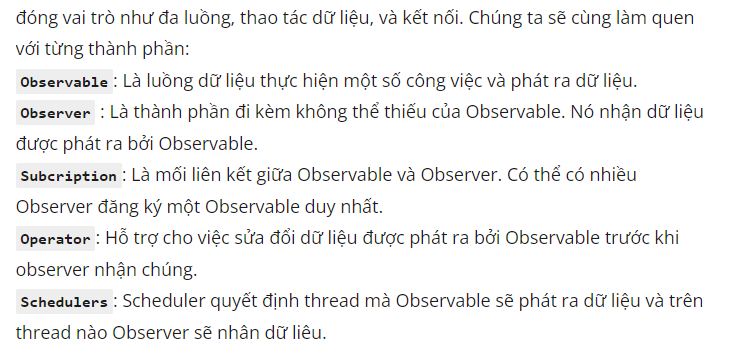


1. Intall\_time permission: cho phép ứng dụng hoạt động hạn chế ảnh hưởng đến ứng dụng khác hoặc hệ thống (ví dụ: run khi khởi động, xem kết nối wifi..)
2. Signature Permission: tự tạo quyền của riêng mjh, hệ thống sẽ cấp quyền cho ứng dụng có chữ ký đầu tiên ở thời điểm cài đặt
3. Special Permission: quyền đặc biệt, tương ứng với các hoạt động cụ thể

RX Java

* Là thư viện cung cấp các sự kiện không đồng bộ, thay đổi và sử dụng dữ liệu bằng observer, cung cấp nhiều loại operator như map, filter,..
* Rx Android : Scheduler bổ sung cho RXAndroid nhằm hỗ trợ phân luồng





<https://viblo.asia/p/rxjava-rxandroid-co-ban-E375z0rjZGW>

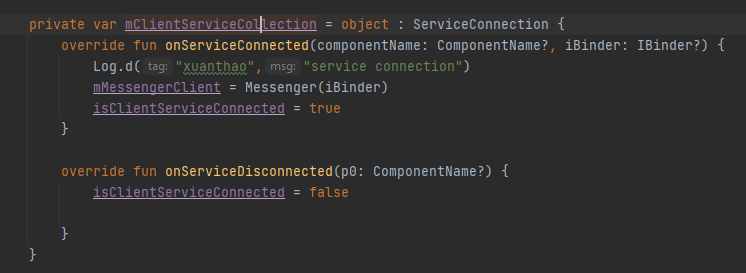
IPC(INTER PROCESS COMMUNICATION)

Giao tiếp giữa các process

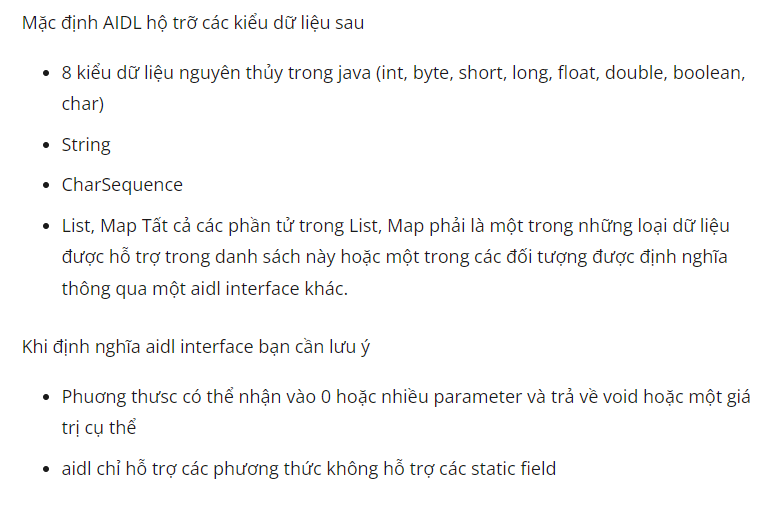
* Tại sao cần IPC:

1. share data giữa 2 user
2. Broadcast cũng là 1 cách giao tiếp IPC
3. Tăng tốc độ xử lý
4. chạy nhiều tác vụ cùng 1 lúc

* IPC Messenger: Chỉ cho phép handle request tại 1 thời điểm
* Sử dụng khi muốn hoạt động trên các process. Có thể tạo giao diện cho service bằng cách sử dụng messenger. Khi đó service sẽ định nghĩa 1 lớp handler để phản hồi lại các message
* Ibinder: Cung cấp giao diện cho user có thể tương tác với service(trả về hàm onBind của service)
* 1 client liên kết với service bằng bindservice. Khi thực hiện nó cung cấp 1 service connection để quan sát
* onServiceConnected: khi client bắt đầu liên kết với service
* onServiceDisconnected: khi ngừng liên kết



* AIDL: Cho phép client và server có thể giao tiếp với nhau thông qua IPC
* Sử dụng AIDL khi khác process
* Các bước tạo file .aidl: định nghĩa 1 lớp interface



* b2: build project để gen ra file java để có thể sử dụng
* m Binder là khởi tạo của Stub Class đã implement
* biến này trả về client thông qua IBinder
* các phương thức ko đảm bảo chạy trên main thread nên sử dụng IO
* Cho phép gửi nhiều request cùng 1 lúc đó là điểm hơn IPC Messenger