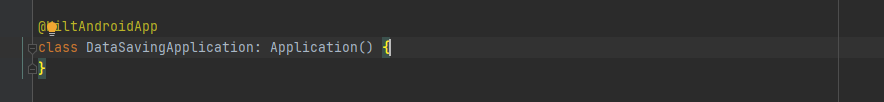
Câu hỏi mock:

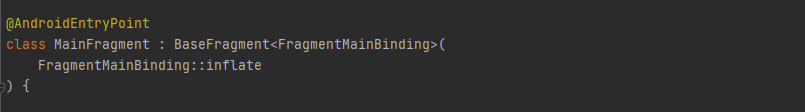
* Translation x : sự biến dạng
* Lấy giá trị width height trên máy thật : <https://stackoverflow.com/questions/63407883/getting-screen-width-on-api-level-30-android-11-getdefaultdisplay-and-getme>

Dagger Hilt

* Mục tiêu
* Đơn giản hóa dagger, khả năng mã hóa cao và dễ test
* Tạo tập hợp thành phần và phạm vi tiêu chuẩn, dễ đọc, dễ thiết lập
* Cung cấp các rang buộc khác nhau cho các loại bản dựng khác nhau
* Application: định nghĩa các công việc, vùng chứa phụ thuộc cấp ứng dụng

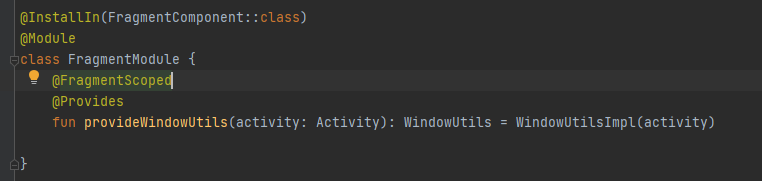


* Sau khi hilt đc thiết lập trong application, hilt cung cấp các phụ thuộc cho các lớp của android khác bằng chú thích @AndroidEntryPoint



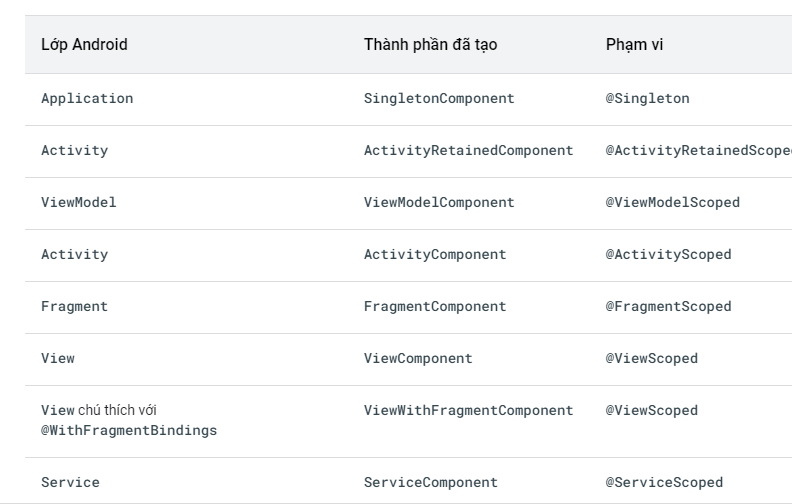
* @AndroidEntryPoint tạo 1 thành phần Hilt riêng lẻ, để lấy phụ thuộc từ 1 thành phần, hãy sử dụng @Inject để chèn trường

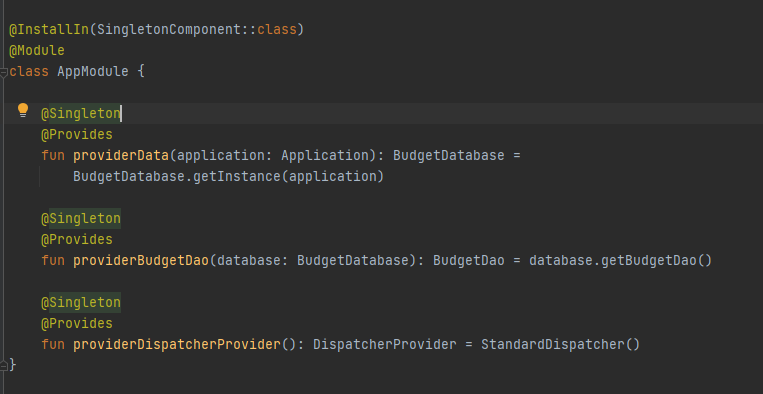




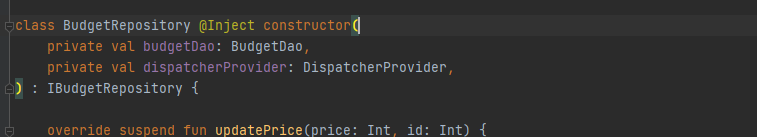
* @Installn(FragmentComponent::class): muốn hilt đưa phụ thuộc này vào fragment
* Tất cả các rang buộn ko đc lưu, khia start hilt sẽ tạo phiên bản mới nhất



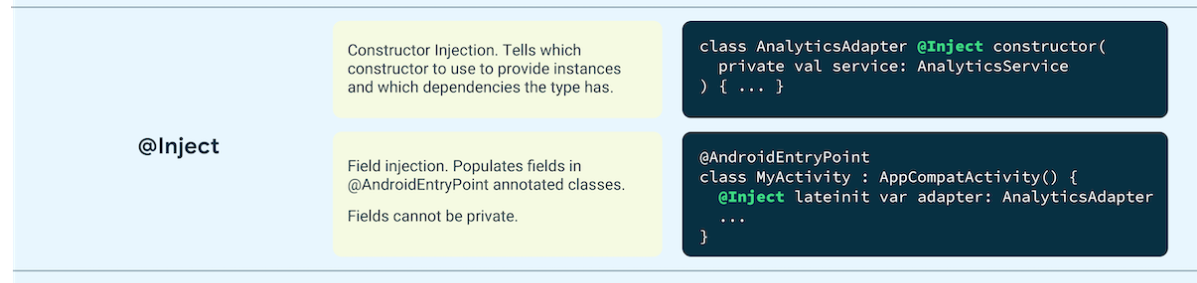




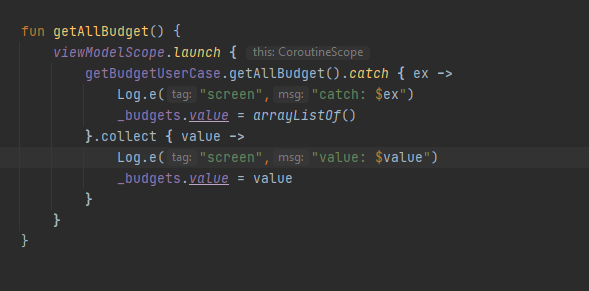
* @Singleton: Instance chia sẻ trên nhiều viewmodel khác nhau, đc phân chia phạm vi dung @singleton, nó là 1 scope anotion, mỗi hilt component sẽ có scope(pvi) annotation tương ứng
* @Provides: cho biết kiểu Instance mà hilt cung cấp ví dụ như Budget Database, BudgetDao như hình trên
* Kiểu trả về là kiểu ràng buộc, tham số phụ thuộc
* @InstallIn: cho biết thông tin này săn có trên vùng chưa (ví dụ singleton Component ở trên )



* Hilt tự biết cũng cấp các Instance cách chú thích hàm tạo của nó với @Inject



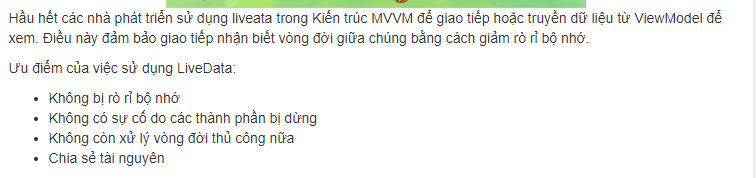
Flow



* Chỉ chạy khi collect
* Flow có thể chạy bất đồng bộ
* Catch: cho phép emit giá trị nguồn thu

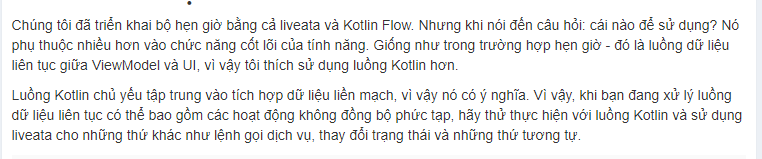
Live Data

* Livedata là 1 triển khai liflcle aware giúp nhận biện vòng đời, thành phần quan sát bị phá hủy khi view ko hoạt động



Livedata và Flow

* Livedata được sử dụng để quan sát dữ liệu mà không cần có bất kỳ phần mềm nào để xử lý các vấn đề trong vòng đời. Trong khi luồng Kotlin được sử dụng để tích hợp dữ liệu liên tục và nó cũng đơn giản hóa việc lập trình không đồng bộ.
* Sau một thời gian, phòng đã sử dụng Kotlin flow để giải quyết vấn đề này. Với Flow là loại trả lại, phòng đã tạo ra một khả năng mới là tích hợp dữ liệu liền mạch trên ứng dụng giữa cơ sở dữ liệu và giao diện người dùng mà không cần viết thêm bất kỳ mã nào.



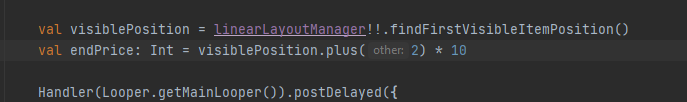
RequireActivity và RequireContext

* fragment năm trong 1 fragment khác thì requireContext sẽ tham chiếu đến fragment cha, nếu fragment năm trong 1 activity thì requireContext sẽ tham chiếu đến activity cha
* còn requireActivity tham chiếu đến activity cha

Viewpager

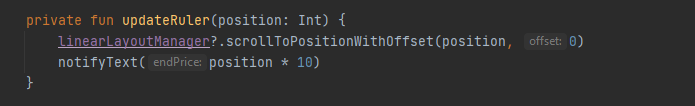
* clipToPadding: cho phép toàn bộ chiều rông đc mở rộng với padding
* clipChidrent: các phần bên trái và bên phải đc hiển thị
* offScreenPageLimit: giới hạn số trang ngoài màn hình
* getChildAt(0).overScrollMode = NEVER : xóa animation đổ bóng cuộn qua viewpager

Ruler

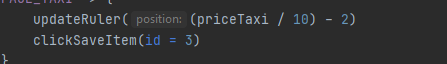


Visible : lấy vị trí đầu tiên còn nhìn thấy trên màn hình

enPrice :\*10 vì bước nhảy là 10, giá trị tròng chucj, 1 vạch đơn vị tương ứng với giá trị 10

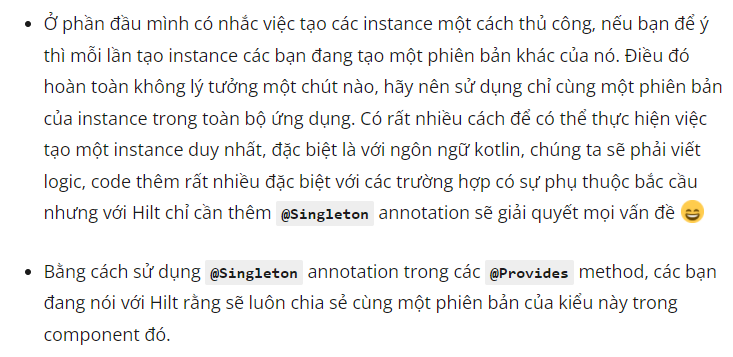


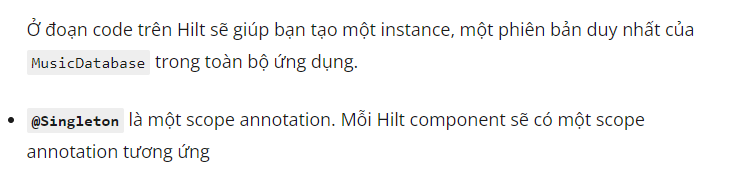
* scroll…. Cuộn đến vị trí còn nhìn thấy đầu tiên



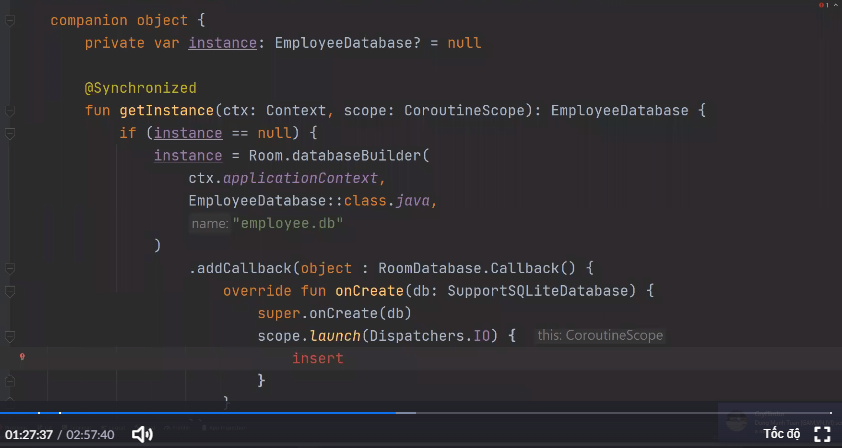
* -2 vì đánh dấu vị trí là 2, để đánh lạc hướng vạch có giá trị 2







* slideDownt : fillafter :trở lại vị trí ban đầu

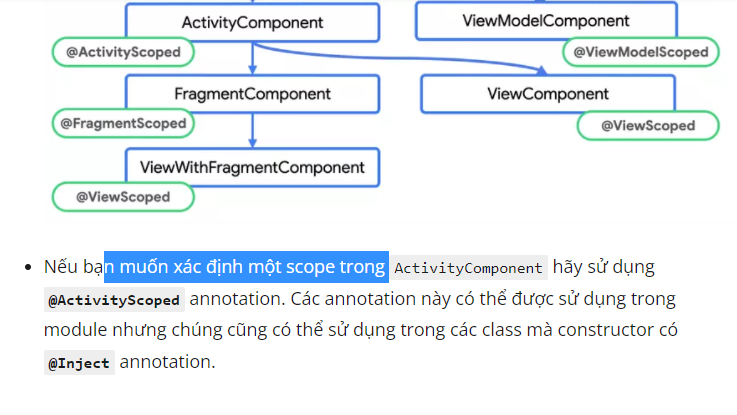


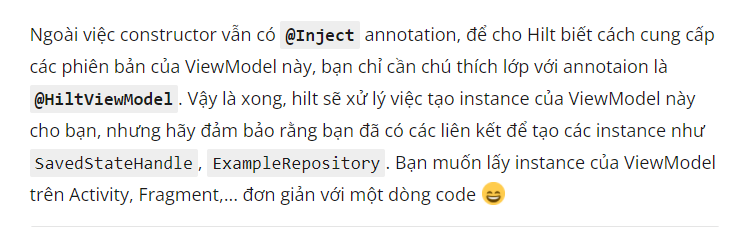
RoomDatabase

* fallbackToDestructiveMiration(): cho phép room tạo lại 1 cách triệt để bảng cơ sở dữ liệu nếu Miration ko tìm thấy cơ sở dữ liệu thì sẽ chuyển sang bản cơ sở dữ liệu mới nhất
* Export Schema: nếu true nó import ra file json : cấu trúc các version để theo dõi, version này có cột này, version kia lại them cột kia
* @Volatile : có 2 thread cùng gọi, thằng nào chạy xong trc thì notify cho thằng kia biết là ko tạo nữa, biến instance nhận đc kết quả khi các thread cùng truy cập đến

USE CASE

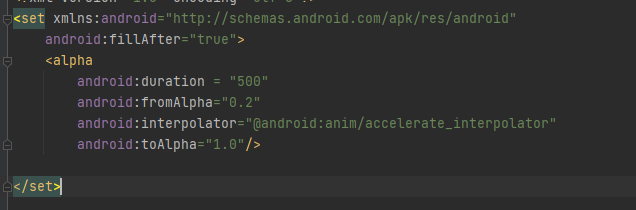
* ViewModel gọn gang hơn
* Có logic data flow tg tự, có thể tái sử dụng
* Implement data flow logic
* Logic data flow có thể dung ở các viewmodel kh



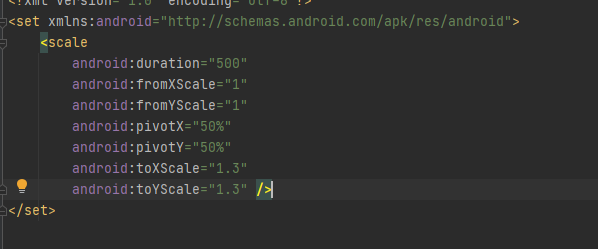


<https://viblo.asia/p/tong-quan-ve-hilt-trong-loat-bai-ky-nang-modern-android-development-mad-Do754zeQZM6>

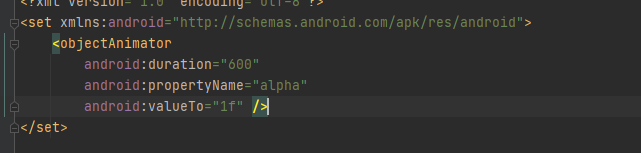
fadeIn



zoomIn



hiện alpha



Utils

