CODE NOTES - NavDrawerDemo App

All quotes in this document from developer.android.com
No linked pages are required reading in this document

Although it's possible to code a drawer from scratch, if you know you are going to use one it's easiest to use the Navigation Drawer Activity template as you create your new project.

NAVIGATION DRAWER vs. OPTIONS MENU

"The options menu is the primary collection of menu items for an activity. It's where you should place actions that have a global impact on the app, such as "Search," "Compose email," and "Settings.""

"The nav drawer spans the height of the screen, with everything behind it visible but darkened by a scrim."

"As per the Android Design guide, any drawers positioned to the left/start should always contain content for navigating around the application, whereas any drawers positioned to the right/end should always contain actions to take on the current content. This preserves the same navigation left, actions right structure present in the Action Bar and elsewhere."

It would appear, in general, the options menu is for actions and left opening navigation draws are for navigation. Remember that Option menu items can appear as action buttons in the App bar. Navigation drawer menu items cannot do this.

MATERIAL DESIGN

Gradle Scripts/build.gradle (Module App).

Note the dependency: compile 'com.android.support:design:…'.

This design library includes many Views (UI components) such as the DrawerLayout and NavigationView and other classes such as ActionBarDrawerToggle that implement the look/feel and behaviour of a UI according to Material Design principles. Some other MD related components such as CardView and RecyclerView require their own libraries to be added.

The appcompat-v7 library also supplies MD support with backward compatibility

continued …

TO SET UP A DRAWER YOU NEED:

- An instance of the DrawerLayout component in an Activity's launch layout (drawer_layout in activity_main in this case)
    - It's the launch Activitiy's top View (even above any layout, it's a ViewGroup but still surprising!!!)
    - It coordinates the visual interaction between the drawer and the UI without the drawer open
        - i.e. it doesn't represent the Drawer itself. Possible clue as to why it's the top View
    - Must be coded, it's not in the component palette
- An instance of the NavigationView component (nav_view in this case) as a child of the DrawerLayout
    - Represents the drawer itself
    - Must be coded, it's not in the component palette
    - Attributes point to a layout for the drawer's header and an XML description of the drawer's menu
    - In code a reference to the NavigationView is obtained and used to set a listener for menu item clicks
- A layout describing the header section of the drawer (nav_header_main in this case)
    - Optional. NavigationView's headerLayout attribute value points at this
- A menu layout specifying the options to be displayed within the drawer (activity_main_drawer in this case)
    - NavigationView's menu attribute value points at this
    - Same XML vocabulary as for Options menu layout.
- A listener object containing event handling code to respond to user selection of each of the drawer's menu items. This listener is assigned to the NavigationView in code using its setNavigationItemSelectedListener method in the launch Activity's onCreate lifecycle callback
    - Not the same as Options menu where its Activity automatically listens for the menu to be opened by the user (onCreateOptionsMenu) and for an a menu item to be selected (onOptionsItemSlected)
        - Here we set the Activity instance as the designated Listener:
            - Make the application promise to implement the appropriate interface i.e. MainActivity implements NavigationView.OnNavigationItemSelectedListener
            - Code the interface's only method onNavigationItemSelected in the Activity
            - Handle the possible menu item selections in a conditional control structure
- An ActionBarDrawerToggle instance
    - "This class provides a handy way to tie together the functionality of DrawerLayout and the framework ActionBar to implement the recommended design for navigation drawers."
    - This includes
        - Inserting a hamburger icon in the AppBar that rotates on drawer open/close by whatever means (swipe, hamburger icon, code)
        - Actually opening and closing the drawer using the hamburger icon and synchronising the two
    - It's set up by passing an instantiated instance as the only parameter to the DrawLayout's addDrawerListener method in the launch Activity's onCreate lifecycle callback
        - This should be followed by an immediate call to the instance's syncState method for initial synchronisation of the drawer's open state and its hamburger icon in the AppBar

OPENING AND CLOSING THE DRAWER

To open and close the drawer from code (rather than user interaction) use the DrawerLayout's closeDrawer(…) and openDrawer(…) methods.

Should be done after every menu item selection when the appropriate response has executed. So, after the conditional control structure in the onNavigationItemSelected method.

In addition, if the drawer is open when the back button is clicked intercept the back button's usual consumption and action by Android and instead make it just close the drawer. This can be coded in the Activity's onBackPressed() method.

LAYOUT ANALYSIS

- activity_main (launch layout)
    - Contains DrawerLayout (coordinates draw and UI with draw closed)
    - Includes app_bar_main
        - Contains CoordinatorLayout coordinating:
            - AppBarLayout containing Toolbar
            - content_main
                - which is just a TextView in this template version
            - FAB
                - Default template version
    - Contains NavigationView (represents the drawer itself)
        - Attribute value of which specifies nav_header_main
            - The drawer's header layout
        - Attribute value of which specifies menu activity_main_drawer
            - A menu layout file (top element menu) that specifies the drawer's menu
                - Same XML vocabulary as for Options menu
- main
    - Nothing to do with the drawer just the layout of Options menu (default template version)

So, counter-intuitively the top element is a DrawerLayout not a CoordinatorLayout but that is because it doesn't represent the drawer but rather the interaction of the drawer and the UI with the drawer closed.

WHAT ABOUT THE PRIMARY CONTENT OF THE UI?

"To use a DrawerLayout, position your primary content view as the first child with width and height of match_parent and no layout_gravity. Add drawers as child views after the main content view and set the layout_gravity appropriately. Drawers commonly use match_parent for height with a fixed width."

It helps to use an include for the primary content. This keeps the layout complexity down and allows you to focus on editing at a given layout in the View hierarchy.

The drawer's layout_gravity should be "start" for draws opening from the left and "end" for drawers opening from the right. Google are trying to deprecate "left" and "right" for some reason and replace them with "start" and "end". Same move in Java's Swing libraries so must be a good reason.