Pedro

The following are some tips when setting up a Xamarin.Android app using Navigation Components using a navigationgraph.xml.

Normally when creating an Android app, the template provided for say a Blank app runs out of the box as soon as you build it and deploy it. It certainly doesn’t do much, but it does run.

The same can’t be said if the app contains an activity\_main.xml which contains an androidx.fragment.app.FragmentContainerView. If you deploy and run it will fail on SetContentView(Resource.Layout.activity\_main) with an exception.

Android.Views.InflateException

Message=Binary XML file line #1 in com.companyname.navigationadvancedsample:layout/activity\_main: Binary XML file line #1 in com.companyname.navigationadvancedsample:layout/activity\_main: Error inflating class androidx.fragment.app.FragmentContainerView

At first glance, on looking at the minimal Navigation code in the MainActivity you wouldn’t expect that, as the code looks pretty much normal. However as soon as you use a FragmentContainerView, you automatically then require the rest of the components that make up a single Activity/multiple Fragments Navigation Component app.

Obviously, the nav\_graph as in app:navGraph="@navigation/nav\_graph" is necessary, but there are also other requirements before you will get the app to run. Just having a populated nav\_graph.xml is not enough.

The nav\_graph refers to various Fragments, therefore the code for those Fragments must also exist. The implementation needs only to be minimal to successfully run the app. Containing a public parameterless constructor (required by Android for configuration changes) plus the OnCreateView containing the xml file of the view that is required for the inflation of the xml of that fragment. The fully qualified class name will also be required as the android:name attribute for each fragment in the graph. As per the example below for the Title fragment from the NavigationSampleAdvanced.

android:name="com.companyname.NavigationAdvancedSample.Fragments.Home.Title"

Notice that all navigation is done via Resource.Id’s or matching MenuItemId’s. See examples android:id in nav.graph and in the fragment classes Title and Register in the event handler Navigation.FindNavController((View)sender).Navigate(Resource.Id.action\_title\_to\_about), and finally android:id in the menu bottom\_nav.

If you haven’t changed the default namespace after first building the app you will need to change it to, for example, com.companyname.NavigationAdvancedSample or whatever you wish as the namespace name and then change it in two places in Resources.Designer.cs at the top of the file, first the attribute line and then the namespace itself and also the namespace of the MainActivity. Rebuild and correct any errors if you have forgotten to change one of the files.

I’d also suggest creating a Fragments folder so that all the fragments of the app are created in that folder. If you require subfolders such as used in this NavigationAdvancedSample app (using a nested nav\_graph), then create them too. Just remember that whatever namespace names you chose they all have to match when it comes to building the nav\_graph. One other tip, which I’ve ignored here, is naming each fragment. Naming a fragment Title is not very informative as is done in this sample. Naming it TitleFragment is far more informative and readable.

Another tip, the default using statements automatically added when you create a new Fragment Class using Add/New Item/Fragment, defaults to “using Android.App”. This should have been changed by Xamarin years ago because Android.App.Fragment was deprecated long ago. Comment that out and right-click on Fragment and the using AndroidX.Fragment.App will appear in the *usings suggestion* and fix all those green squigglies.

The NavigationComponents also cause another fault or more precisely the nav.graph.xml file. Sometimes the fault doesn’t appear straight away, but it eventually will. In this example, it was evident once the xml of the first fragment was created. When opening an xml file representing the view of the fragment with the Xamarin.Android designer, it will always error in the designer window which of course results in an xml view where you can’t view the components of your view. It however does not affect the successful building of the app. Xamarin has known of this problem since Feb 2020 and nothing has been done to fix it.

The easiest way to work around it is to design all your layouts in Android Studio and then either copy/paste them into a new xml file in Visual Studio. Since most Xamarin.Android developers now use ConstraintLayout as the default layout, which the Xamarin.Android designer has never had support for, it is probably a moot point anyway that the Xamarin.Android designer doesn’t receive updates any longer. The only disadvantage is that you should resist making quick small changes via the Xamarin.Android designer as this will get them out of sync. Alternatively, to avoid copy/paste mistakes do consider using Beyond Compare and load both the VS layout and the Android Studio layout to quickly make changes in the Android Studio version and send those changes (via BC ) back to the VS layout.

You can also temporarily exclude the Navigation folder from the project and then open the xml file with the Xamarin.Android designer to at least view the xml without error. Just remember to include it again before you rebuild.

**Please note that this project is not complete as compared to the original Android sample. In other words the Leaderboard is not complete. I’ll leave that for you to complete. I lost interest when I realised the the project really doesn’t do that much other than demonstration multi backstack support.**

**However, the Xamarin verson does show you how to get around the particular problems you experienced.**

Documentation

Android source.

<https://github.com/android/architecture-components-samples/tree/master/NavigationAdvancedSample>.

Article – Navigation – Multiple back stacks

https://medium.com/androiddevelopers/navigation-multiple-back-stacks-6c67ba41952f

Article: Multiple back stacks

<https://medium.com/androiddevelopers/multiple-back-stacks-b714d974f134>

Article: Fragments rebuilding the internals

<https://medium.com/androiddevelopers/fragments-rebuilding-the-internals-61913f8bf48e>