

SwiftUI Generic Router

Reusable Navigation for Any
Module



```
/// A generic router for managing navigation between
different screens in SwiftUI
protocol NavigationDestination {
    associatedtype Destination: View

    var title: String { get }
    @ViewBuilder
    var destinationView: Destination { get }
}

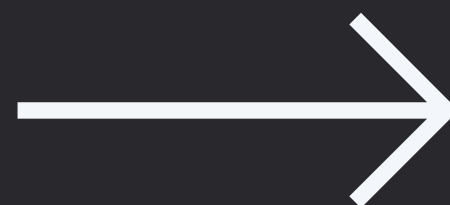
final class Router<Destination:
    NavigationDestination>: ObservableObject {

    /// Holds the stack of destinations for
    navigation
    @Published var navPaths: [Destination] = []

    func navigate(to destination: Destination) {
        navPaths.append(destination)
    }

    func navigateBack() {
        guard !navPaths.isEmpty else { return }
        navPaths.removeLast()
    }

    func navigateToRoot() {
        navPaths.removeLast(navPaths.count)
    }
}
```

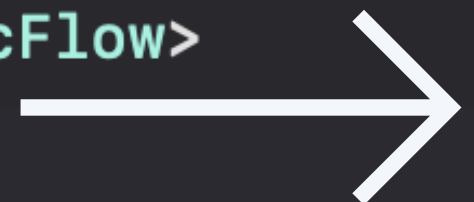


```
/// Enum representing the music flow with associated
    titles for navigation
enum MusicFlow: NavigationDestination {
    /// configure your screens
    case first
    case second
    case third

    /// Titles for each screen in the music flow
    var title: String {
        switch self {
            case .first:
                return "First Screen – Overview of Music"
            case .second:
                return "Second Screen – Explore Genres"
            case .third:
                return "Third Screen – Final Music
                    Choices"
        }
    }

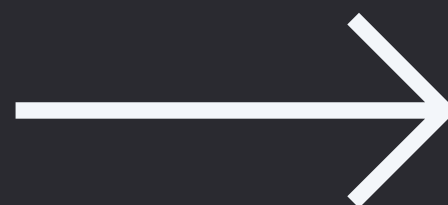
    /// Destination views for each screen
    var destinationView: some View {
        switch self {
            case .first: FirstScreenView()
            case .second: SecondScreenView()
            case .third: ThirdScreenView()
        }
    }
}

typealias MusicFlowRouter = Router<MusicFlow>
```



```
/// ContentView contains a TabView to switch between
    Music and Movie flows
struct ContentView: View {
    var body: some View {
        TabView {
            Tab("Musics", systemImage:
                "music.note.house.fill") {
                MusicView()
            }

            Tab("Movies", systemImage:
                "movieclapper.fill") {
                MovieView()
            }
        }
    }
}
```



```

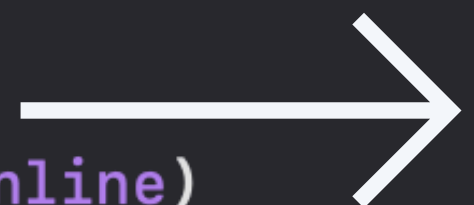
/// MusicView manages the Music flow and uses Router
    to handle navigation
struct MusicView: View {

    /// Router managing navigation between the screens
    @StateObject private var router = MusicFlowRouter()

    var body: some View {
        NavigationStack(path: $router.navPaths) {
            mainView
                .navigationDestination(for:
                    MusicFlow.self) { destination in
                        destination
                            .destinationView
                                /// Dynamic title based on flow
                                .navigationTitle(destination
                                    .title)
                                .toolbarRole(.editor)
                            }
            }
        }
        /// Inject router for global navigation
        management
        .environmentObject(router)
    }

    private var mainView: some View {
        VStack {
            Button("Go to first screen") {
                router.navigate(to: .first) /// IMP
            }
        }
        .navigationTitle("Musics")
        .navigationBarTitleDisplayMode(.inline)
    }
}

```



Need a full video or code?

Check the link in the post!

**Will you give
this a try?**

Leave a comment below

