Navigation & routing mechanisms

Navigation V1 and Navigation V2

Navigation V1 VS **Navigation V2**

Flutter Developer

@ Ahmed Hussein
01114484229

Flutter Navigation V1 and V2

In Flutter, navigation is the process of moving between different screens or views in an application. There are two main versions of navigation: **Navigation V1** and **Navigation V2**

Feature	Navigation V1	Navigation V2
API Type	Imperative	Declarative
Navigation Control	Managed via `Navigator.push()` and `Navigator.pop()` calls	Uses `Router` and `RouteInformationParser` for more control
Deep Linking	Limited or requires manual setup	Built-in support with `RouteInformationProvider`
State Management	Stateful, relies on manual state tracking	Stateful, integrates better with app state and URLs
Complex Navigation	More difficult with nested navigation	Simplified with `RouterDelegate` and `NavigationRail`
URL Synchronization	Manual handling	Automatic URL sync using `RouteInformationParser`
Use Cases	Simple, single-page apps	Complex apps with deep links and nested navigation

@ Ahmed Hussein
01114484229

1- Navigation V1 Example

This is the imperative way of handling navigation, where we use **Navigator.push()** and **Navigator.pop().**

```dart
//* Navigate to the second screen using Navigator V1 (imperative way)
Navigator.push(
  context,
  MaterialPageRoute(builder: (context) => const SecondScreen()),
);
//* pop the current screen and back to the first screen using Navigator V1 (imperative way)
Navigator.pop(context);
```

2- Navigation V2 Example

With Navigation V2, we define a **Router**, **RouterDelegate**, and **RouteInformationParser** in MaterialApp.router.

```dart
final MyRouterDelegate _routerDelegate = MyRouterDelegate();
 final MyRouteInformationParser _routeInformationParser = MyRouteInformationParser();
 @override
 Widget build(BuildContext context) {
   return MaterialApp.router(
     routerDelegate: _routerDelegate,
     routeInformationParser: _routeInformationParser,
   );
 }
```


@ Ahmed Hussein
01114484229

2- Navigation V2 Example

RouterDelegate manages the navigation stack declaratively.

```dart
//*********  Router Delegate //****************** */
class MyRouterDelegate extends RouterDelegate<String> with ChangeNotifier, PopNavigatorRouterDelegateMixin<String> {
  @override
  final GlobalKey<NavigatorState> navigatorKey = GlobalKey<NavigatorState>();
  String? _selectedPage;

  @override
  Widget build(BuildContext context) {
    return Navigator(
      key: navigatorKey,
      pages: [
        MaterialPage(child: HomePage(onTapped: _handlePageTapped)),
        if (_selectedPage == 'details') MaterialPage(child: DetailsPage(onBack: _handleBack)),
      ],
      onDidRemovePage: (route) {
        if (route.canPop) {
          _selectedPage = null;
          notifyListeners();
        }
      },
    );
  }

  void _handlePageTapped() {
    _selectedPage = 'details';
    notifyListeners();
  }
  void _handleBack() {
    _selectedPage = null;
    notifyListeners();
  }

  @override
  Future<void> setNewRoutePath(String configuration) async {
    _selectedPage = configuration == '/details' ? 'details' : null;
  }

  @override
  String? get currentConfiguration => _selectedPage == 'details' ? '/details' : '/';
}
```

RouteInformationParser interprets the route information.

```dart
//*********  Route Information Parser //****************** */
class MyRouteInformationParser extends RouteInformationParser<String> {
  @override
  Future<String> parseRouteInformation(RouteInformation routeInformation) async {
    return routeInformation.uri.pathSegments.isEmpty ? '/' : routeInformation.uri.pathSegments.first;
  }

  @override
  RouteInformation? restoreRouteInformation(String configuration) {
    return RouteInformation(uri: Uri(path: configuration));
  }
}
```


@ Ahmed Hussein
01114484229

2- Navigation V2 With go_router Package

The above code works well, but it is **hard to maintain** and **complex**. By using **go_router Package**, developers can easily manage complex navigation scenarios, including nested routes and deep linking, especially in **web** applications where URL-based navigation is important. go_router has many Feature Please visit package documentation Link

```dart
//* GoRouter V2 Example (Declarative way)
  final GoRouter _router = GoRouter(
    routes: [
      GoRoute(
        path: '/',
        builder: (context, state) => const HomePage(),
      ),
      GoRoute(
        path: '/details',
        builder: (context, state) => const DetailsPage(),
      ),
    ],
  );

  @override
  Widget build(BuildContext context) {
    return MaterialApp.router(
      routerDelegate: _router.routerDelegate,
      routeInformationParser: _router.routeInformationParser,
      routeInformationProvider: _router.routeInformationProvider,
    );
  }
```

```dart
//* Navigate to the details page using GoRouter V2 (declarative way)
context.go('/details');
// //* pop the current screen and back to the first screen using Navigator V1 (imperative way)
context.go('/');
```


@ Ahmed Hussein
01114484229

Ahmed Hussein
Flutter Developer

THANK YOU

Was This Helpful?

Like, Share and Follow
for more Content