Flutter Navigation 2.0: Advanced Routing

Subtitle: Learn how to implement declarative and dynamic navigation in Flutter using Navigator 2.0.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Navigator 2.0 allows for **more control over routing**, supporting **deep links, dynamic pages, and URL-based navigation**. This guide teaches how to implement advanced routing for modern apps.

Step 1: Basic Router Setup

```
class AppRouter extends RouterDelegate<RouteSettings>
    with ChangeNotifier, PopNavigatorRouterDelegateMixin<RouteSettings> {

  final GlobalKey<NavigatorState> navigatorKey = GlobalKey<NavigatorState>();

  @override
  RouteSettings? get currentConfiguration => RouteSettings(name: '/');

  @override
  Widget build(BuildContext context) {
    return Navigator(
      key: navigatorKey,
      pages: [
        MaterialPage(child: HomePage()),
      ],
      onPopPage: (route, result) => route.didPop(result),
    );
  }

  @override
  Future<void> setNewRoutePath(RouteSettings configuration) async {}
}
```

Exercise: Setup a basic RouterDelegate for your Flutter app.

Step 2: Adding Multiple Pages

```
List<Page> getPages() {
  return [
    MaterialPage(child: HomePage()),
    if(showDetails)
      MaterialPage(child: DetailsPage()),
  ];
}
```

Exercise: Add a boolean flag to control navigation to a details page.

Step 3: Using RouteInformationParser

```
class AppRouteParser extends RouteInformationParser<RouteSettings> {
  @override
  Future<RouteSettings> parseRouteInformation(RouteInformation routeInfo) async
    final uri = Uri.parse(routeInfo.location!);
    if(uri.pathSegments.isEmpty) return RouteSettings(name: '/');
    return RouteSettings(name: '/${uri.pathSegments.first}');
  }
}
```

Exercise: Parse incoming URLs to determine which page to show.

Step 4: Mini Project

Build a **Flutter Multi-Page App**:

- HomePage, DetailsPage, ProfilePage
- Use Navigator 2.0 with RouterDelegate and RouteInformationParser
- Support deep linking and URL navigation
- Bonus: Use query parameters to pass data between pages

Key Takeaways

- Navigator 2.0 supports **declarative and dynamic routing**
- Separates routing logic from UI for **cleaner architecture**
- Essential for **large apps with deep linking and complex navigation**

- Mastering Navigator 2.0 makes your Flutter apps **scalable and modern**

Skipping advanced routing limits your app to **basic navigation**, reducing flexibility for real-world use cases.

Visit **haas.dev** for more Flutter guides, tutorials, and complete project resources.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app
