Navigation in Flutter. Navigate to the new screen using Navigator.push and Navigator.pushNamed

Navigation in flutter

Almost every app developed for real-world problems contains multiple screens. Whether its social media apps like Facebook, Instagram or video watching platform like YouTube and Tik-Tok. Every complex app contains multiple screens.

Every platform has its own way to navigate between different screens. Whether its Android, iOS, React Native, Ionic, Apache Cordova, or Xamarin and the same thing go for Flutter as well.

Terminology: In flutter, Screens are called **Routes**. In Android, a *route* is equivalent to *Activity*, and in iOS, a route is equivalent to a *ViewController.*

In flutter, just like everything else, the route is also a widget. To manage routes, flutter uses the Navigator widget. Navigator manages all the routes and also provides methods to navigate between them like `Navigator.push()` and `Navigator.pushNamed()`.

In flutter, there are two ways to navigate to a new route aka Screen.

1. Using Navigator.push()
2. Using Navigator.pushNamed()

Using Navigator.push()

If you have a limited route (like one or two) then you can use *Navigator.push()* method. The *push()* method adds a Route to the stack of the routes managed by the Navigator.

Flutter gives you `MaterialPageRoute`, which transitions to the new route using a platform-specific animation. But flutter is so flexible. You can create your own route and transitions animation for a specific platform.

Navigator.push sample code

Return to the first route using Navigator.pop()

When we use *Navigator.push()* then to return to the first route, we have to use *Navigator.pop()* method. The *pop()* method removes the current Route from the stack of the routes managed by the Navigator widget.

Navigator.pop sample code

Sample app using Navigator.push()

Sample app using Navigator.push()

Using Navigator.pushNamed()

Navigator.push() method works when you have only two or three routes. But when it comes to complex apps, you might have multiple routes. In that case, if we use *Navigator.push()* method then it will result in a lot of code duplication.

So when you have multiple routes in your app then you should use *Navigator.pushNamed()*. To identify and differentiate between multiple routes, we can give a name to the routes. This makes easy to navigate between different routes.

To use *Navigator.pushNamed(),* we have to follow two steps:

1. declare `routes` property in the `MaterialApp` constructor.
2. call the *Navigator.pushNamed()* method when needed

To define routes, we have to provide two additional properties to the `MaterialApp` constructor.

1. initialRoute
2. routes

Just like the `home` property of the `MaterialApp`, the `initialRoute` property defines from which route the app should start. Like the starting screen of the app.

Warning: *When using* `initialRoute`*,* ***don't*** *define a* `home` *property.*

The `routes` property defines the ***available routes*** and ***the widgets to build*** when the route is called.

Properties of MaterialApp constructor

Once we define the properties in `initialRoute` and `routes`, we can now call the *Navigator.pushNamed()* method whenever we need it.

When we call the *Navigator.pushNamed()* method, we need to define the route name. After getting the route name, flutter knows which screen (widget) it needs to build.

```dart
// Navigate to the second screen using a named route.
  Navigator.pushNamed(context, 'routeName');
```

Navigator.pushNamed() sample code

Sample app using Navigation.pushNamed()

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(MaterialApp(
    title: 'Named Routes Demo',
    // Start the app with the "homeScreen" named route. In this case, the app starts
    // on the HomeScreen widget.
    initialRoute: 'homeScreen',
    routes: {
      // When navigating to the "homeScreen" route, build the HomeScreen widget.
      'homeScreen': (context) => HomeScreen(),
      // When navigating to the "secondScreen" route, build the SecondScreen widget.
      'secondScreen': (context) => SecondScreen(),
    },
  ));
}

```

```dart
class FirstScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('First Screen'),
      ),
      body: Center(
        child: RaisedButton(
          child: Text('Launch screen'),
          onPressed: () {
            // Navigate to the second screen using a named route.
            Navigator.pushNamed(context, 'secondScreen');
          },
        ),
      ),
    );
  }
}

class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Second Screen"),
      ),
      body: Center(
        child: RaisedButton(
          onPressed: () {
            // Navigate back to the first screen by popping the current route
            // off the stack.
            Navigator.pop(context);
          },
          child: Text('Go back!'),
        ),
      ),
    );
  }
}
```