Flutter: Điều hướng

Bùi Võ Quốc Bảo

Khoa CNTT | Trường CNTT-TT | Đại học Cần Thơ

Tài liệu tham khảo

- https://docs.flutter.dev/cookbook/navigation
- https://docs.flutter.dev/cookbook/design

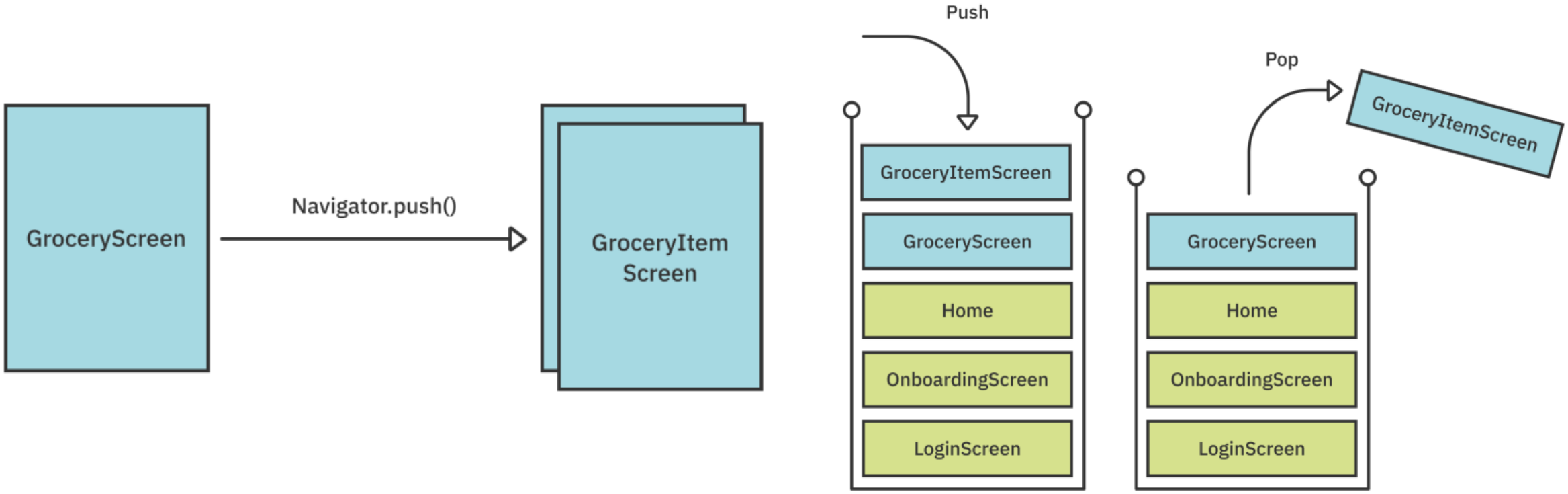
Điều hướng trong Flutter

- Điều hướng: chuyển đổi giữa các màn hình (screen) khác nhau
- Flutter hỗ trợ hai cơ chế điều hướng:
  - Dạng mệnh lệnh (imperative) với Navigator API (Navigator 1.0)
  - Dạng khai báo (declarative) với Router API (Navigator 2.0)
- Trong Flutter, các thuật ngữ "màn hình" (screen), "trang" (page) và "tuyến đường" (route) có ý nghĩa tương đương nhau (và đều là widget)

Navigator 1.0 (Navigator API)

- Các trang được tổ chức theo dạng ngăn xếp (stack) và được quản lý bởi widget Navigator
- Trường hợp thông thường chỉ cần sử dụng widget Navigator được tạo và cấu hình thông qua MaterialApp
- Trường hợp khác, ví dụ như cần điều hướng lồng nhau (nested navigation), có thể trực tiếp tạo widget Navigator
- Truy xuất đến widget Navigator gần nhất trong cây widget thông qua **Navigator.of**

Navigator 1.0 (Navigator API)



Chuyển đến trang mới và trở lại

Tạo hai trang: FirstRoute và SecondRoute

```dart
class FirstRoute extends StatelessWidget {
  const FirstRoute({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('First Route'),
      ),
      body: Center(
        child: ElevatedButton(
          child: const Text('Open route'),
          onPressed: () {
            // Navigate to second route when tapped.
          },
        ),
      ),
    );
  }
}
```

```dart
class SecondRoute extends StatelessWidget {
  const SecondRoute({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Second Route'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            // Navigate back to first route when tapped.
          },
          child: const Text('Go back!'),
        ),
      ),
    );
  }
}
```

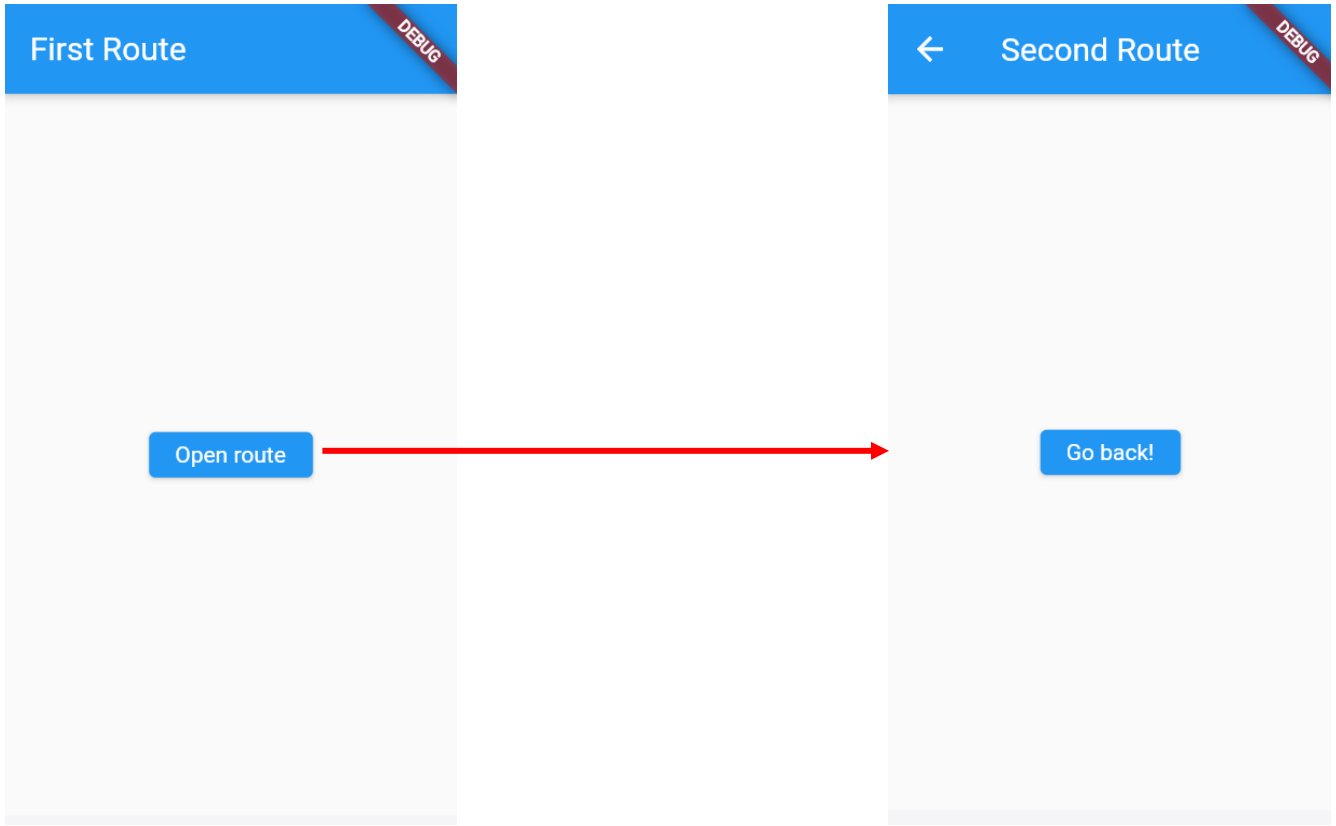
Chuyển đến trang mới và trở lại

Chuyển từ FirstRoute đến SecondRoute dùng **Navigator.of(context).push()** hoặc **Navigator.push()**

```
// Within the `FirstRoute` widget
onPressed: () {
  Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => const SecondRoute()),
  );
}
```

Chuyển đến trang mới và trở lại

Chuyển từ FirstRoute đến SecondRoute dùng **Navigator.of(context).push()** hoặc **Navigator.push()**



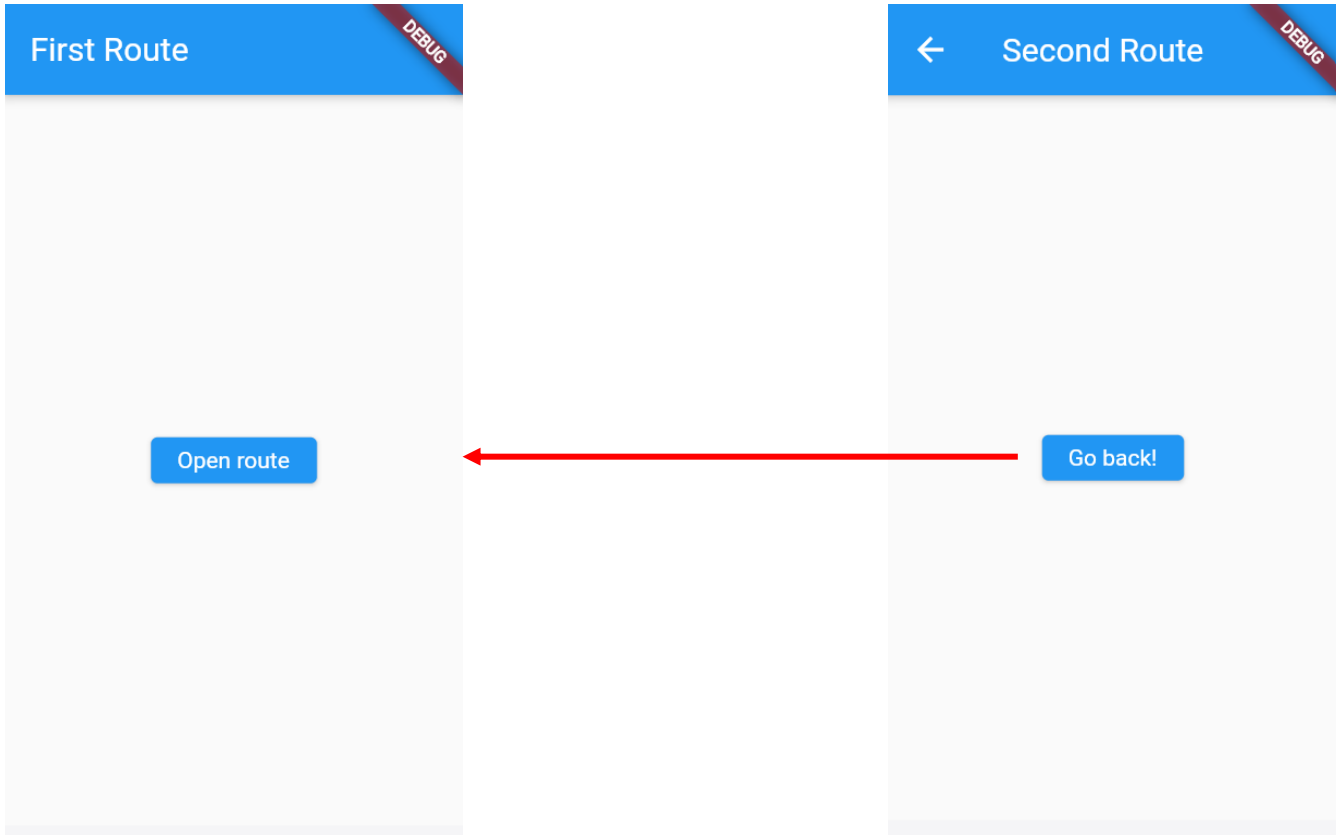
Chuyển đến trang mới và trở lại

Trở lại FirstRoute từ SecondRoute dùng
Navigator.of(context).pop() hoặc **Navigator.pop(context)**

```
// Within the SecondRoute widget
onPressed: () {
  Navigator.pop(context);
}
```

Chuyển đến trang mới và trở lại

Trở lại FirstRoute từ SecondRoute dùng **Navigator.of(context).pop()** hoặc **Navigator.pop(context)**



Gửi dữ liệu cho trang mới

Truyền tham số với **Navigator.push()**

```
ListView.builder(
  itemCount: todos.length,
  itemBuilder: (context, index) {
    return ListTile(
      title: Text(todos[index].title),
      // When a user taps the ListTile, navigate to the DetailScreen.
      // Notice that you're not only creating a DetailScreen, you're
      // also passing the current todo through to it.
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => const DetailScreen(),
            // Pass the arguments as part of the RouteSettings. The
            // DetailScreen reads the arguments from these settings.
            settings: RouteSettings(
              arguments: todos[index],
            ),
          ),
        );
      },
    );
  },
)
```

Gửi dữ liệu cho trang mới

Đọc tham số trực tiếp trong widget sử dụng **ModalRoute.of()**

```dart
class DetailScreen extends StatelessWidget {
  const DetailScreen({super.key});

  @override
  Widget build(BuildContext context) {
    final todo = ModalRoute.of(context)!.settings.arguments as Todo;

    // Use the Todo to create the UI.
    return Scaffold(
      appBar: AppBar(
        title: Text(todo.title),
      ),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Text(todo.description),
      ),
    );
  }
}
```

Trả về dữ liệu từ một trang

```dart
// A method that launches the SelectionScreen and awaits the result from
// Navigator.pop.
Future<void> _navigateAndDisplaySelection(BuildContext context) async {
  // Access BuildContext before an async call
  final scaffold = ScaffoldMessenger.of(context);

  // Navigator.push returns a Future that completes after calling
  // Navigator.pop on the Selection Screen.
  final result = await Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => const SelectionScreen()),
  );

  // After the Selection Screen returns a result, hide any previous snackbars
  // and show the new result.
  scaffold
    ..removeCurrentSnackBar()
    ..showSnackBar(SnackBar(content: Text('$result')));
}
```

Trả về dữ liệu từ một trang

```
ElevatedButton(
  onPressed: () {
    // Close the screen and return "Yep!" as the result.
    Navigator.pop(context, 'Yep!');
  },
  child: const Text('Yep!'),
)
```

Điều hướng với route có tên (named route)

Định nghĩa hai trang: FirstScreen và SecondScreen

```dart
class FirstScreen extends StatelessWidget {
  const FirstScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('First Screen'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            // Navigate to the second screen when tapped.
          },
          child: const Text('Launch screen'),
        ),
      ),
    );
  }
}
```

```dart
class SecondScreen extends StatelessWidget {
  const SecondScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Second Screen'),
      ),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            // Navigate back to first screen when tapped.
          },
          child: const Text('Go back!'),
        ),
      ),
    );
  }
}
```

Điều hướng với route có tên (named route)

Định nghĩa các route của ứng dụng

```
MaterialApp(
  title: 'Named Routes Demo',
  // Start the app with the "/" named route. In this case, the app starts
  // on the FirstScreen widget.
  initialRoute: '/',
  routes: {
    // When navigating to the "/" route, build the FirstScreen widget.
    '/': (context) => const FirstScreen(),
    // When navigating to the "/second" route, build the SecondScreen widget.
    '/second': (context) => const SecondScreen(),
  },
)
```

Lưu ý: không đặt giá trị cho thuộc tính **home** của **MaterialApp** nếu đã sử dụng **initialRoute**

Điều hướng với route có tên (named route)

Chuyển từ FirstScreen sang SecondScreen với **Navigator.pushNamed()**

```
// Within the `FirstScreen` widget
onPressed: () {
  // Navigate to the second screen using a named route.
  Navigator.pushNamed(context, '/second');
}
```

Điều hướng với route có tên (named route)

Trở về FirstScreen từ SecondScreen

```
// Within the SecondScreen widget
onPressed: () {
  // Navigate back to the first screen by popping the current route
  // off the stack.
  Navigator.pop(context);
}
```

Điều hướng với route có tên (named route)

Truyền tham số với **Navigator.pushNamed()**

```
// A button that navigates to a named route.
// The named route extracts the arguments
// by itself.
ElevatedButton(
  onPressed: () {
    // When the user taps the button,
    // navigate to a named route and
    // provide the arguments as an optional
    // parameter.
    Navigator.pushNamed(
      context,
      ExtractArgumentsScreen.routeName,
      arguments: ScreenArguments(
        'Extract Arguments Screen',
        'This message is extracted in the build method.',
      ),
    );
  },
  child: const Text('Navigate to screen that extracts arguments'),
),
```

```
// You can pass any object to the arguments parameter.
// In this example, create a class that contains both
// a customizable title and message.
class ScreenArguments {
  final String title;
  final String message;

  ScreenArguments(this.title, this.message);
}
```

Điều hướng với route có tên (named route)

Đọc tham số trực tiếp trong widget sử dụng **ModalRoute.of()**

```dart
// A Widget that extracts the necessary arguments from
// the ModalRoute.
class ExtractArgumentsScreen extends StatelessWidget {
  const ExtractArgumentsScreen({super.key});

  static const routeName = '/extractArguments';

  @override
  Widget build(BuildContext context) {
    // Extract the arguments from the current ModalRoute
    // settings and cast them as ScreenArguments.
    final args = ModalRoute.of(context)!.settings.arguments as ScreenArguments;

    return Scaffold(
      appBar: AppBar(
        title: Text(args.title),
      ),
      body: Center(
        child: Text(args.message),
      ),
    );
  }
}
```

Điều hướng với route có tên (named route)

Đọc tham số trong **onGenerateRoute()** và gửi cho widget

```
MaterialApp(
  // Provide a function to handle named routes.
  // Use this function to identify the named
  // route being pushed, and create the correct
  // Screen.
  onGenerateRoute: (settings) {
    // If you push the PassArguments route
    if (settings.name == PassArgumentsScreen.routeName) {
      // Cast the arguments to the correct
      // type: ScreenArguments.
      final args = settings.arguments as ScreenArguments;

      // Then, extract the required data from
      // the arguments and pass the data to the
      // correct screen.
      return MaterialPageRoute(
        builder: (context) {
          return PassArgumentsScreen(
            title: args.title,
            message: args.message,
          );
        },
      );
    }

    assert(false, 'Need to implement ${settings.name}');
    return null;
  },
)
```

Navigator API

https://api.flutter.dev/flutter/widgets/Navigator-class.html

Một số phương thức điều hướng khác của Navigator:

- **pushReplacement** / **pushReplacementNamed**: thay thế trang hiện thời với trang mới
- **popUtil(context, predicate)**: gọi pop liên tiếp cho đến khi predicate trả về true
  - ModalRoute.withName(): predicate dựa trên route
- **pushAndRemoveUntil** / **pushNamedAndRemoveUntil (context, predicate)**: thêm trang mới và xóa các trang trước đó cho đến khi predicate trả về true

Tạo hiệu ứng cho một widget khi chuyển trang

Hai trang cùng hiển thị một ảnh

```dart
class MainScreen extends StatelessWidget {
  const MainScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Main Screen'),
      ),
      body: GestureDetector(
        onTap: () {
          Navigator.push(context, MaterialPageRoute(builder: (context) {
            return const DetailScreen();
          }));
        },
        child: Image.network(
          'https://picsum.photos/250?image=9',
        ),
      ),
    );
  }
}
```

```dart
class DetailScreen extends StatelessWidget {
  const DetailScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: GestureDetector(
        onTap: () {
          Navigator.pop(context);
        },
        child: Center(
          child: Image.network(
            'https://picsum.photos/250?image=9',
          ),
        ),
      ),
    );
  }
}
```

Tạo hiệu ứng cho một widget khi chuyển trang

Kết nối hai trang với hiệu ứng: bao widget cần tạo hiệu ứng với widget **Hero**. Widget **Hero** yêu cầu hai tham số:

- **tag**: đối tượng xác định Hero. Phải giống nhau ở cả hai trang
- **child**: widget cần tạo hiệu hứng cho giữa các trang

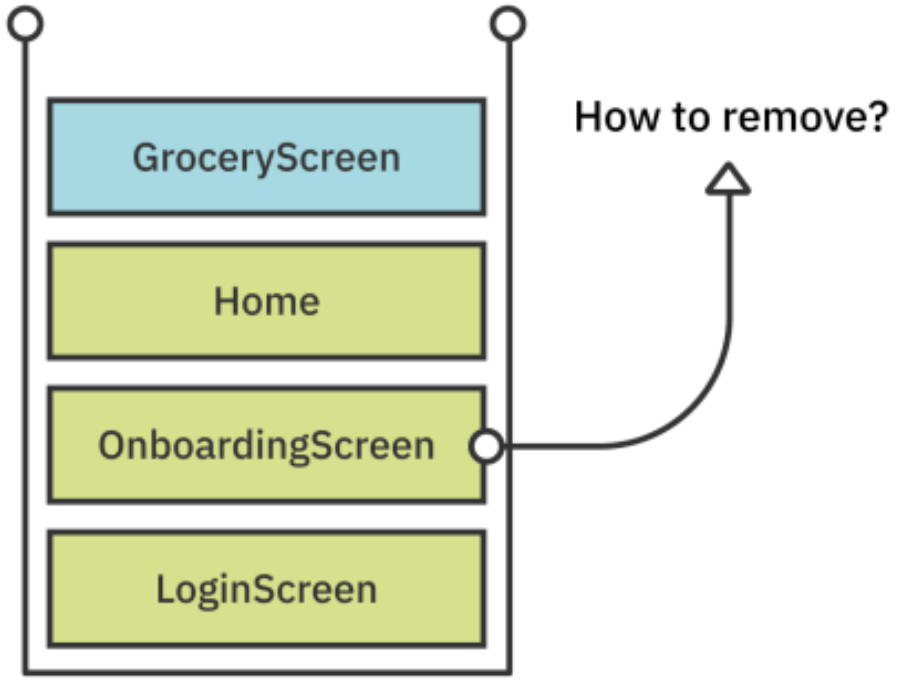
```
Hero(
  tag: 'imageHero',
  child: Image.network(
    'https://picsum.photos/250?image=9',
  ),
)
```

Tạo hiệu ứng tùy biến khi chuyển trang

Xem tham khảo tại:
https://docs.flutter.dev/cookbook/animation/page-route-animation

Nhược điểm của Navigator API

- Navigator 1.0 không thể hiện ngăn xếp route ra cho nhà phát triển
  - Phải tự ghi nhớ ngăn xếp route
  - Khó xử lý những yêu cầu định tuyến phức tạp (thêm, xóa các trang trong ngăn xếp route)
- Trong một số trường hợp như ứng dụng có các navigator lồng nhau, nút Back trên các thiết bị Android có thể không hoạt động đúng



Navigator 2.0 (Router API)

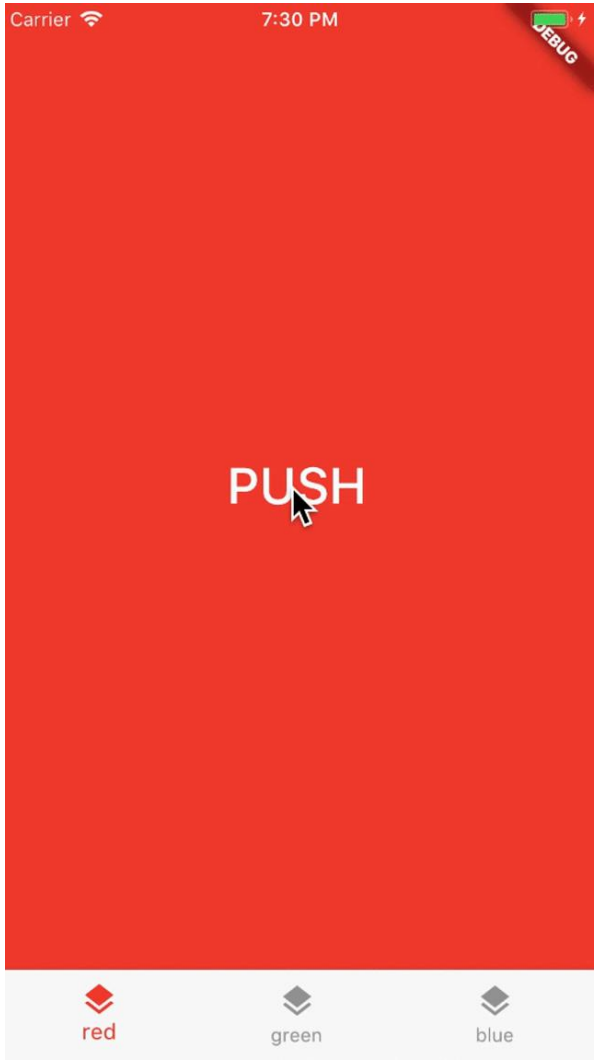
- API dạng khai báo (declarative), khắc phục được hầu hết các nhược điểm của Navigator API
- Mạnh mẽ và mềm dẻo hơn so với Navigator API tuy nhiên cài đặt phức tạp/dài dòng hơn
- Thường không sử dụng trực tiếp Router API mà dùng các thư viện wrapper với API đơn giản hơn như **go_router** hoặc **beamer**

Điều hướng lồng nhau (nested navigation)

- Một ứng dụng có thể có nhiều hơn một Navigator
- Một trường hợp thường gặp là điều hướng lồng bên trong một BottomNavigationBar
- Tham khảo ví dụ cài đặt điều hướng lồng nhau:
  - Navigator 1.0: https://codewithandrea.com/articles/multiple-navigators-bottom-navigation-bar/
  - Navigator 2.0: https://codewithandrea.com/articles/flutter-bottom-navigation-bar-nested-routes-gorouter-beamer/

Điều hướng lồng nhau (nested navigation)

Dùng Navigator mặc định của ứng dụng



Dùng Navigator riêng cho từng thẻ

Điều hướng với Drawer và Tab

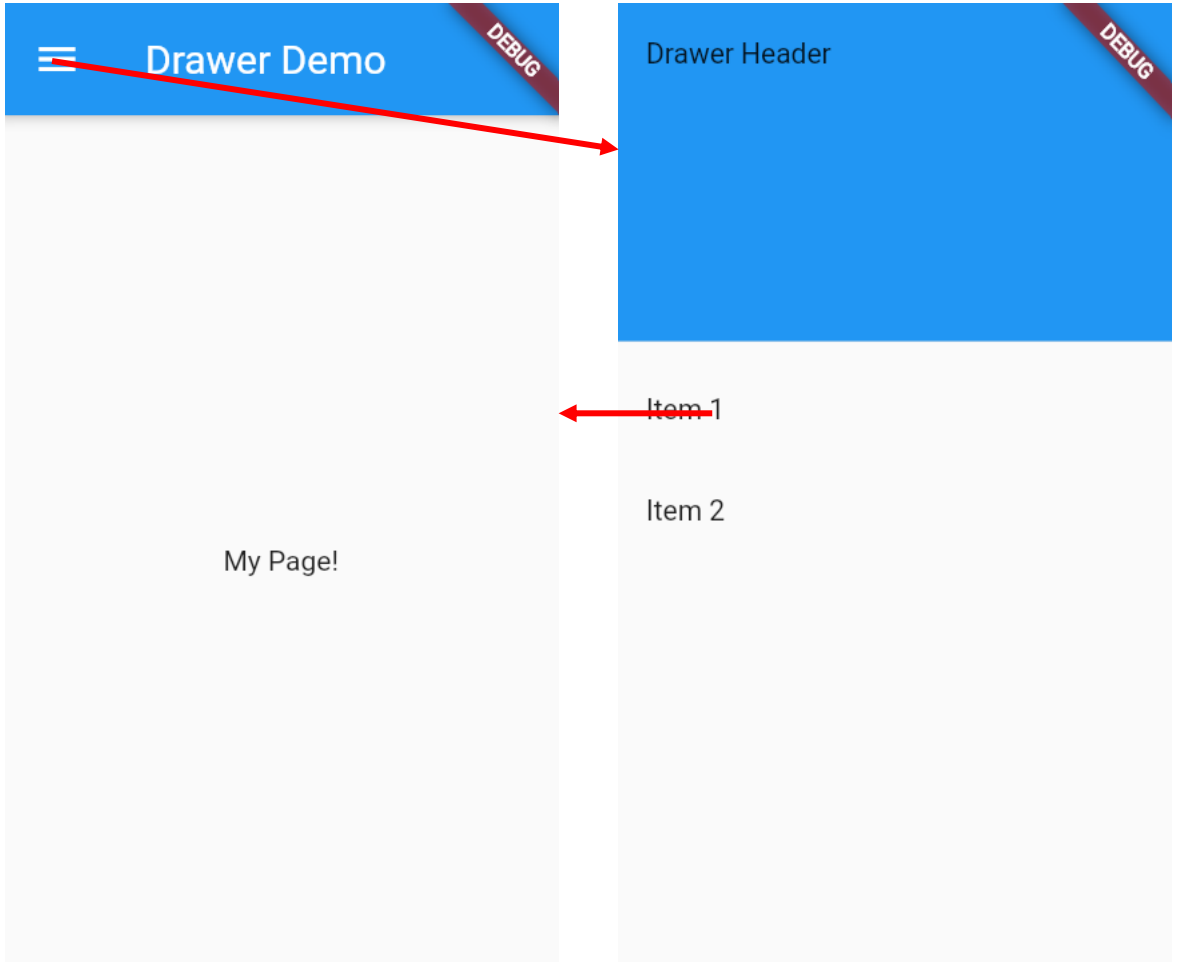
Thêm Drawer

- Trong hệ thống *thiết kế Material*, hai tùy chọn chính cho điều hướng là Tab và Drawer
- Widget **Scaffold** cung cấp một cấu trúc trực quan, nhất quán cho các ứng dụng tuân thủ theo thiết kế Material
  - Hỗ trợ nhiều thành phần thiết kế Material: Drawer, AppBar, SnackBar
- Thêm Drawer

```
Scaffold(
  drawer: Drawer(
    child: // Populate the Drawer in the next step.
  ),
);
```

Thêm Drawer

```
return Scaffold(
  ...
  drawer: Drawer(
    // Add a ListView to the drawer. This ensures the user can scroll
    // through the options in the drawer if there isn't enough vertical
    // space to fit everything.
    child: ListView(
      padding: EdgeInsets.zero,
      children: [
        const DrawerHeader(
          decoration: BoxDecoration(color: Colors.blue),
          child: Text('Drawer Header'),
        ),
        ListTile(
          title: const Text('Item 1'),
          onTap: () {
            //...
            Navigator.pop(context);
          },
        ),
        ListTile(
          title: const Text('Item 2'),
          onTap: () {
            //...
            Navigator.pop(context);
          },
        ),
      ],
    ),
  ),
);
```



Làm việc với Tab

- Các tab được tạo với widget **TabBar**
- Nội dung các tab được tạo với widget **TabBarView**
- **TabController** được sử dụng để đồng bộ hóa tab được chọn với phần nội dung
  - TabBar và TabBarView phải cùng chia sẻ một TabController
  - Widget **DefaultTabController**: chia sẻ một TabController với các widget hậu duệ

Làm việc với Tab

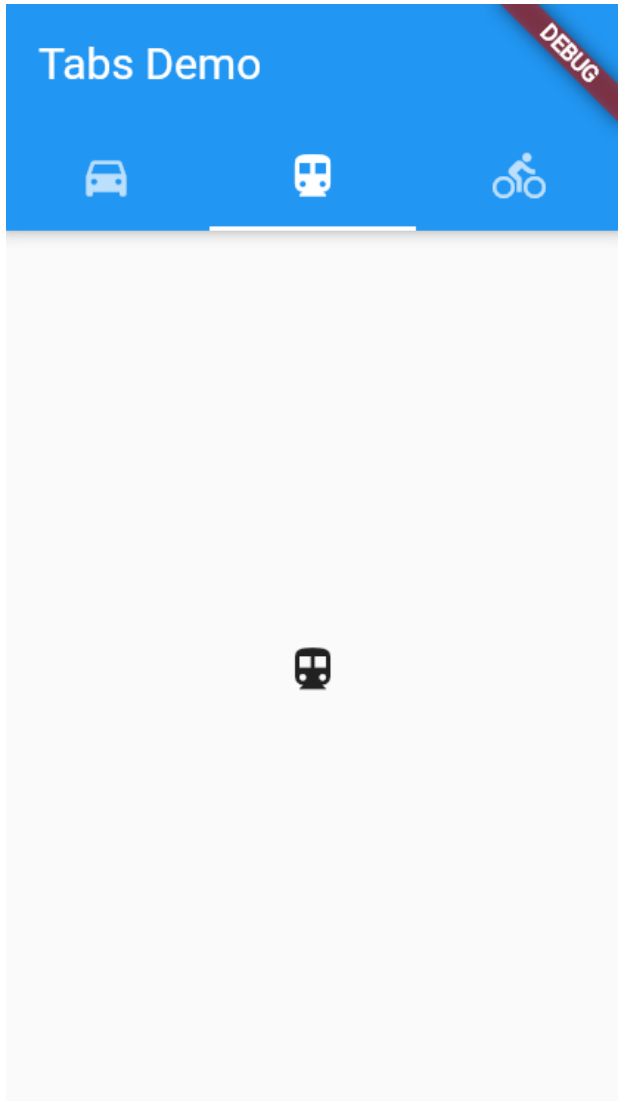
Tạo các tab

```
return MaterialApp(
  home: DefaultTabController(
    length: 3,
    child: Scaffold(
      appBar: AppBar(
        bottom: const TabBar(
          tabs: [
            Tab(icon: Icon(Icons.directions_car)),
            Tab(icon: Icon(Icons.directions_transit)),
            Tab(icon: Icon(Icons.directions_bike)),
          ],
        ),
      ),
    ),
  ),
);
```

Làm việc với Tab

Tạo nội dung các tab

```
body: const TabBarView(
  children: [
    Icon(Icons.directions_car),
    Icon(Icons.directions_transit),
    Icon(Icons.directions_bike),
  ],
),
```



Câu hỏi?