Cách sử dụng các tệp

1. Tạo dự án Flutter mới: flutter create lab4_1
2. Thay thế thư mục lib / và tệp pubspec.yaml bằng các tệp đính kèm
3. Có thể them hình ảnh, font chữ
4. Chạy lệnh: flutter packages get
5. flutter run (với trình giả lập đang chạy hoặc thiết bị thực được kết nối với máy) để xem ứng dụng

Nội dung:

I. Điều hướng đến một màn hình mới và quay lại

- Hầu hết các ứng dụng chứa một số màn hình để hiển thị các loại thông tin khác nhau. Ví dụ: một ứng dụng có thể có màn hình hiển thị các sản phẩm. Khi người dùng chạm vào hình ảnh của sản phẩm, một màn hình mới sẽ hiển thị chi tiết về sản phẩm.
- Điều hướng đến một tuyến đường mới bằng cách sử dụng Navigator. Công thức này sử dụng các bước sau:
- Một số phần tiếp theo cho thấy cách điều hướng giữa hai tuyến đường, sử dụng các bước sau:
  + Tạo hai tuyến đường.
  + Điều hướng đến tuyến đường thứ hai bằng Navigator.push ().
  + Quay trở lại tuyến đường đầu tiên bằng Navigator.pop ().

1. Tạo hai tuyến đường

- Đầu tiên, tạo hai tuyến đường để làm việc với. Vì đây là một ví dụ cơ bản, mỗi tuyến chỉ chứa một nút duy nhất. Nhấn vào nút trên tuyến đầu tiên điều hướng đến tuyến thứ hai. Nhấn vào nút trên tuyến thứ hai sẽ trở lại tuyến đầu tiên.

```
import 'package:flutter/material.dart';
void main() {
  runApp(MaterialApp(
    title: 'Navigation Basics',
    home: FirstRoute(),
  ));
}
```

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

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

2. Điều hướng đến tuyến đường thứ hai bằng Navigator.push()

- Để chuyển sang một tuyến mới, sử dụng Navigator.push() phương pháp. Các push()phương pháp bổ sung thêm một Routevào stack của các tuyến đường do quản lý Navigator. Trong build() phương thức của FirstRoutewidget, cập nhật cuộc onPressed() gọi lại

```
class FirstRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('First Route'),
      ),
      body: Center(
        child: RaisedButton(
          child: Text('Open route'),
          onPressed: () {
            // Navigate to second route when tapped.
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => SecondRoute()),
            );
          },
        ),
      ),
    );
  }
}
```

3. Quay trở lại tuyến đường đầu tiên bằng Navigator.pop ()

- Bằng cách sử dụng Navigator.pop()phương pháp. Các pop() phương pháp loại bỏ các hiện Routetừ ngăn xếp của các tuyến đường do quản lý Navigator.
- Để thực hiện quay lại tuyến đường ban đầu, hãy cập nhật cuộc onPressed() gọi lại trong SecondRoute tiện ích:

```
class SecondRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
      appBar: AppBar(
        title: Text("Second Route"),
      ),
      body: Center(
        child: RaisedButton(
          onPressed: () {
            // Navigate back to first route when tapped.
            Navigator.pop(context);
          },
          child: Text('Go back!'),
        ),
      ),
    );
  }
}
```

II. Điều hướng với các tuyến đường được đặt tên

Trong điều hướng đến một màn hình mới và công thức quay lại, nếu cần điều hướng đến cùng một màn hình trong nhiều phần của ứng dụng, cách tiếp cận này có thể dẫn đến sao chép mã. Giải pháp là xác định tuyến đường được đặt tên và sử dụng tuyến đường được đặt tên để điều hướng.

Để làm việc với các tuyến được đặt tên, sử dụng Navigator.pushNamed() chức năng.

+ Xác định các tuyến đường.

+ Điều hướng đến màn hình thứ hai bằng cách sử dụng Navigator.pushNamed().

+ Quay trở lại màn hình đầu tiên bằng cách sử dụng Navigator.pop().

1. Xác định các tuyến đường

- Xác định các tuyến bằng cách cung cấp các thuộc tính bổ sung cho hàm MaterialApptạo: initialRoute và routes chính chúng.
- Các initialRouteđịnh nghĩa tài sản mà định tuyến các ứng dụng nên bắt đầu với. Các routes bất động sản xác định tên các tuyến đường có sẵn và widget để xây dựng khi điều hướng đến những tuyến đường.

```
void main() {
  runApp(MaterialApp(
    title: 'Navigation Basics',
    // 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 FirstRoute widget.
      '/': (context) => FirstRoute(),
      // When navigating to the "/second" route, build the FirstRoute widget.
      '/second': (context) => FirstRoute(),
    },
  ));
}
```

2. Điều hướng đến màn hình thứ hai

- Với các vật dụng và tuyến đường tại chỗ, kích hoạt điều hướng bằng cách sử dụng Navigator.pushNamed() phương thức. Điều này báo cho Flutter xây dựng widget được xác định trong routes bảng và khởi chạy màn hình.
- Trong build()phương thức của FirstRoute widget, cập nhật cuộc onPressed() gọi lại

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

3. Quay trở lại màn hình đầu tiên
   Để điều hướng trở lại màn hình đầu tiên, hãy sử dụng Navigator.pop()chức năng.

```
class SecondRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Second Route"),
      ),
      body: Center(
        child: RaisedButton(
          onPressed: () {
            // Navigate back to first route when tapped.
            Navigator.pop(context);
          },
          child: Text('Go back!'),
        ),
      ),
    );
  }
}
```

III. Truyền đối số cho một tuyến đường được đặt tên

- Các Navigator cung cấp khả năng điều hướng đến một con đường được đặt tên từ bất kỳ phần nào của một ứng dụng bằng cách sử dụng định danh chung. Trong một số trường hợp, cũng có thể cần truyền đối số cho một tuyến đã đặt tên.
- Có thể hoàn thành nhiệm vụ này bằng cách sử dụng argumentstham số của Navigator.pushNamed() phương thức. Trích xuất các đối số bằng ModalRoute.of() phương thức hoặc bên trong một onGenerateRoute() hàm được cung cấp cho MaterialApphoặc hàm CupertinoApp tạo.
- Công thức này trình bày cách truyền đối số vào một tuyến đã đặt tên và đọc các đối số bằng cách sử dụng ModalRoute.of() và onGenerateRoute()sử dụng các bước sau:

  + Xác định các đối số

  + Tạo một widget trích xuất các đối số.

  + Đăng ký các widget trong routes.

  + Điều hướng đến các widget.

1. Xác định các đối số.

- Đầu tiên, xác định các đối số cần chuyển sang tuyến mới. Trong ví dụ này, truyền hai phần dữ liệu: title Màn hình và a messages
- Để truyền cả hai phần dữ liệu, hãy tạo một lớp lưu trữ thông tin này.

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Navigation with Arguments',
      home: HomeScreen(),
    );
  }
}

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Home Screen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            // A button that navigates to a named route that. The named route
            // extracts the arguments by itself.
            RaisedButton(
              child: Text("Navigate to screen that extracts arguments"),
              onPressed: () {},
            ),
            RaisedButton(
              child: Text("Navigate to a named that accepts arguments"),
              onPressed: () {},
            ),
          ],
        ),
      ),
```

```
    );
  }
}

// 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);
}
```

2. Tạo một tiện ích trích xuất các đối số

- Tiếp theo, tạo một widget trích xuất và hiển thị title và message từ ScreenArguments. Để truy cập ScreenArguments, sử dụng ModalRoute.of() phương pháp. Phương thức này trả về tuyến đường hiện tại với các đối số.

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

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Navigation with Arguments',
      home: HomeScreen(),
    );
  }
}

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Home Screen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
```

```
          children: <Widget>[
            // A button that navigates to a named route that. The named route
            // extracts the arguments by itself.
            RaisedButton(
              child: Text("Navigate to screen that extracts arguments"),
              onPressed: () {},
            ),
            RaisedButton(
              child: Text("Navigate to a named that accepts arguments"),
              onPressed: () {},
            ),
          ],
        ),
      ),
    );
  }
}
// A widget that extracts the necessary arguments from the ModalRoute.
class ExtractArgumentsScreen extends StatelessWidget {
  static const routeName = '/extractArguments';

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

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

3. Đăng ký widget trong routes

Tiếp theo, thêm một mục vào routes cung cấp cho MaterialAppwidget. Việc routes xác định widget nào sẽ được tạo dựa trên tên của tuyến đường.

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Navigation with Arguments',
      home: HomeScreen(),
      routes: {
        ExtractArgumentsScreen.routeName: (context) => ExtractArgumentsScreen(),
      },
    );
  }
}
```

4. Điều hướng đến widget

- Cuối cùng, điều hướng đến ExtractArgumentsScreen khi người dùng chạm một nút bằng cách sử dụng Navigator.pushNamed().
- Cung cấp các đối số cho các tuyến thông qua arguments tài sản. Các ExtractArgumentsScreen trích xuất title và message từ các đối số.

```
class HomeScreen extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text('Home Screen'),

      ),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: <Widget>[
```

```
            // A button that navigates to a named route. The named route
            // extracts the arguments by itself.
            RaisedButton(
              child: Text("Navigate to screen that extracts arguments"),
              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.',
                  ),
                );
              },
            ),
            RaisedButton(
              child: Text("Navigate to a named that accepts arguments"),
              onPressed: () {
                Navigator.pushNamed(
                  context,
                  ExtractArgumentsScreen.routeName,
                  arguments: ScreenArguments(
                    'Accept Arguments Screen',
                    'This message is extracted in the onGenerateRoute function.',
                  ),
                );
              },
```

```
            ),
          ],
        ),
      ),
    );
  }
}
}
```

Hoặc, trích xuất các đối số bằng cách sử dụng onGenerateRoute

+ Thay vì trích xuất các đối số trực tiếp bên trong widget, cũng có thể trích xuất các đối số bên trong một onGenerateRoute() hàm và chuyển chúng vào một widget.

+ Các onGenerateRoute()chức năng tạo ra con đường đúng đắn dựa trên RouteSettings.

a. Tạo PassArgumentsScreen

```
class PassArgumentsScreen extends StatelessWidget {
  static const routeName = '/passArguments';

  final String title;
  final String message;

  // This Widget accepts the arguments as constructor parameters. It does not
  // extract the arguments from the ModalRoute.
  //
  // The arguments are extracted by the onGenerateRoute function provided to the
  // MaterialApp widget.
  const PassArgumentsScreen({
    Key key,
    @required this.title,
    @required this.message,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
      appBar: AppBar(
        title: Text(title),
      ),
      body: Center(
        child: Text(message),
      ),
    );
  }
}
```

b. Sử dụng onGenerateRoute()

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return 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 ScreenArguments args = settings.arguments;

          // 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,
              );
            },
          );
        }
      },
      title: 'Navigation with Arguments',
      home: HomeScreen(),
      routes: {
        ExtractArgumentsScreen.routeName: (context) => ExtractArgumentsScreen(),
      },
    );
```

```
  }
}
```

IV. Trả lại dữ liệu từ màn hình

Trong một số trường hợp, có thể muốn trả lại dữ liệu từ một màn hình mới.

Có thể làm điều này với Navigator.pop() phương pháp bằng các bước sau:

+ Xác định màn hình chính

+ Thêm một nút khởi chạy màn hình lựa chọn

+ Hiển thị màn hình lựa chọn với hai nút

+ Khi nhấn nút, đóng màn hình lựa chọn

+ Hiển thị một thanh đồ ăn nhẹ trên màn hình chính với lựa chọn

1. Xác định màn hình chính

Màn hình chính hiển thị một nút. Khi gõ, nó sẽ khởi chạy màn hình lựa chọn.

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

void main() {
  runApp(MaterialApp(
    title: 'Returning Data',
    home: HomeScreen(),
  ));
}

class HomeScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Returning Data Demo'),
      ),
      body: Center(child: SelectionButton()),
    );
  }
}

class SelectionButton extends StatelessWidget {
  @override
```

```
  Widget build(BuildContext context) {
    return RaisedButton(
      onPressed: () {},
      child: Text('Pick an option, any option!'),
    );
  }
}
```

2. Thêm một nút khởi chạy màn hình lựa chọn

Bây giờ, tạo nút chọn, thực hiện như sau:

+ Khởi chạy SelectionScreen khi nó khai thác.

+ Chờ cho SelectionScreen trả về kết quả.

```
class SelectionButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return RaisedButton(
      onPressed: () {
        _navigateAndDisplaySelection(context);
      },
      child: Text('Pick an option, any option!'),
    );
  }
   // A method that launches the SelectionScreen and awaits the
  // result from Navigator.pop.
  _navigateAndDisplaySelection(BuildContext context) async {
    // Navigator.push returns a Future that completes after calling
    // Navigator.pop on the Selection Screen.
    final result = await Navigator.push(
      context,
      // Create the SelectionScreen in the next step.
      MaterialPageRoute(builder: (context) => null),
    );
  }
}
```

3. Hiển thị màn hình lựa chọn với hai nút

- Xây dựng một màn hình lựa chọn có chứa hai nút. Khi người dùng chạm vào một nút, ứng dụng đó sẽ đóng màn hình lựa chọn và cho phép màn hình chính biết nút nào được nhấn.

- Bước này xác định UI. Bước tiếp theo thêm mã để trả về dữ liệu.

```
class SelectionButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return RaisedButton(
      onPressed: () {
        _navigateAndDisplaySelection(context);
      },
      child: Text('Pick an option, any option!'),
    );
  }
   // A method that launches the SelectionScreen and awaits the
  // result from Navigator.pop.
  _navigateAndDisplaySelection(BuildContext context) async {
    // Navigator.push returns a Future that completes after calling
    // Navigator.pop on the Selection Screen.
    final result = await Navigator.push(
      context,
      // Create the SelectionScreen in the next step.
      MaterialPageRoute(builder: (context) => SelectionScreen()),
    );
  }
}

class SelectionScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Pick an option'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: RaisedButton(
                onPressed: () {
                  // Pop here with "Yep"...
                },
                child: Text('Yep!'),
              ),
```

```
            ),
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: RaisedButton(
                onPressed: () {
                  // Pop here with "Nope"
                },
                child: Text('Nope.'),
              ),
            )
          ],
        ),
      ),
    );
  }
}
```

4. Khi nhấn nút, đóng màn hình lựa chọn

Bây giờ, cập nhật cuộc onPressed() gọi lại cho cả hai nút. Để trả dữ liệu về màn hình đầu tiên, hãy sử dụng Navigator.pop()phương thức chấp nhận đối số thứ hai tùy chọn được gọi result. Bất kỳ kết quả nào được trả về Future nút chọn.

```
class SelectionScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Pick an option'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: RaisedButton(
                onPressed: () {
                  Navigator.pop(context, 'Yep!');
                },
                child: Text('Yep!'),
```

```
            ),
          ),
          Padding(
            padding: const EdgeInsets.all(8.0),
            child: RaisedButton(
              onPressed: () {
                Navigator.pop(context, 'Nop!');
              },
              child: Text('Nope.'),
            ),
          )
        ],
      ),
    ),
  );
 }
}
```

5. Hiển thị một thanh đồ ăn nhẹ trên màn hình chính với lựa chọn

Trong trường hợp này, hiển thị kết quả bằng cách sử dụng _navigateAndDisplaySelection() phương pháp trong SelectionButton

```
class SelectionButton extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return RaisedButton(
      onPressed: () {
        _navigateAndDisplaySelection(context);
      },
      child: Text('Pick an option, any option!'),
    );
  }

  // A method that launches the SelectionScreen and awaits the result from
  // Navigator.pop.
  _navigateAndDisplaySelection(BuildContext context) async {
    // Navigator.push returns a Future that completes after calling
    // Navigator.pop on the Selection Screen.
    final result = await Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => SelectionScreen()),
    );

    // After the Selection Screen returns a result, hide any previous snackbars
```

```
    // and show the new result.
    Scaffold.of(context)
      ..removeCurrentSnackBar()
      ..showSnackBar(SnackBar(content: Text("$result")));
  }
}
```