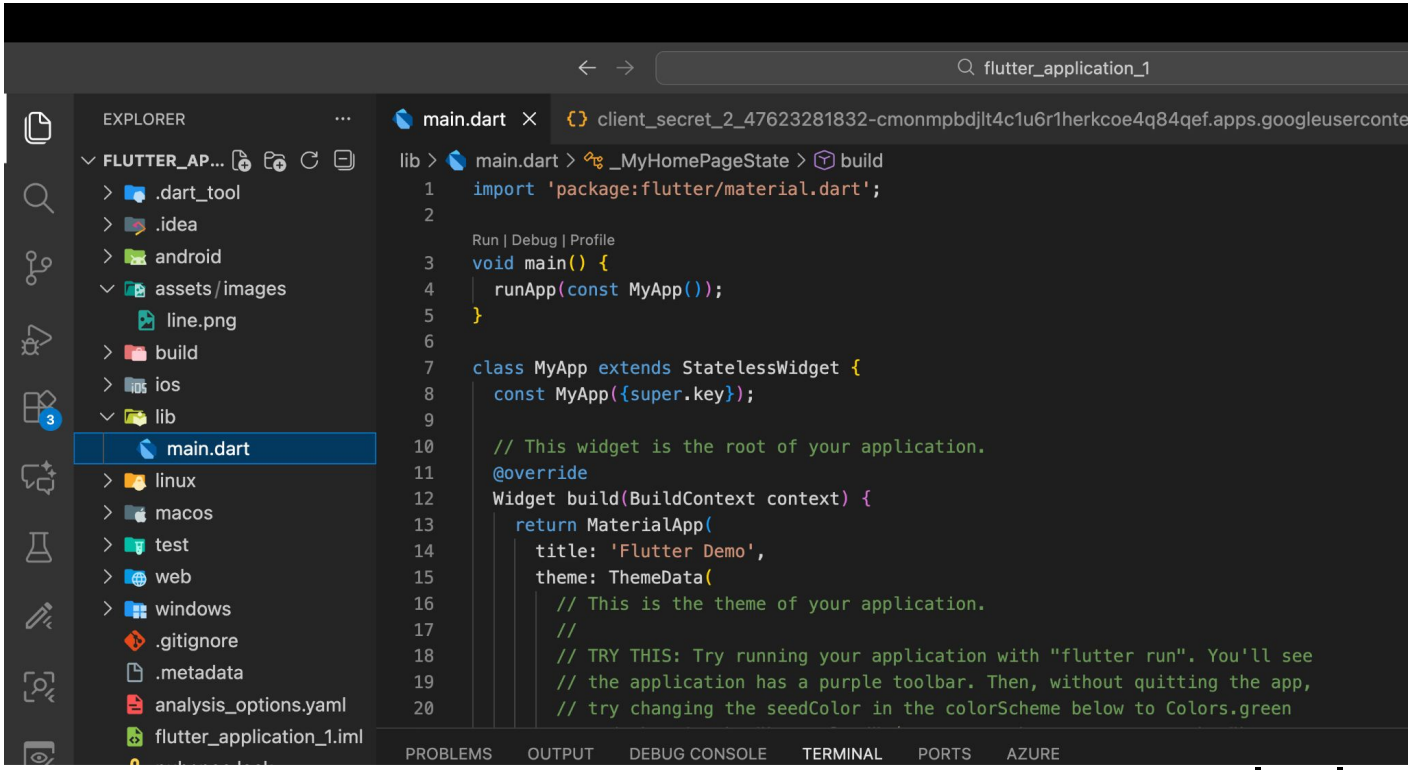

MOBILE PROGRAMMING
WITH FLUTTER - DAY3
เริ่มพัฒนาแอปพลิเคชันบนมือถือด้วย Flutter

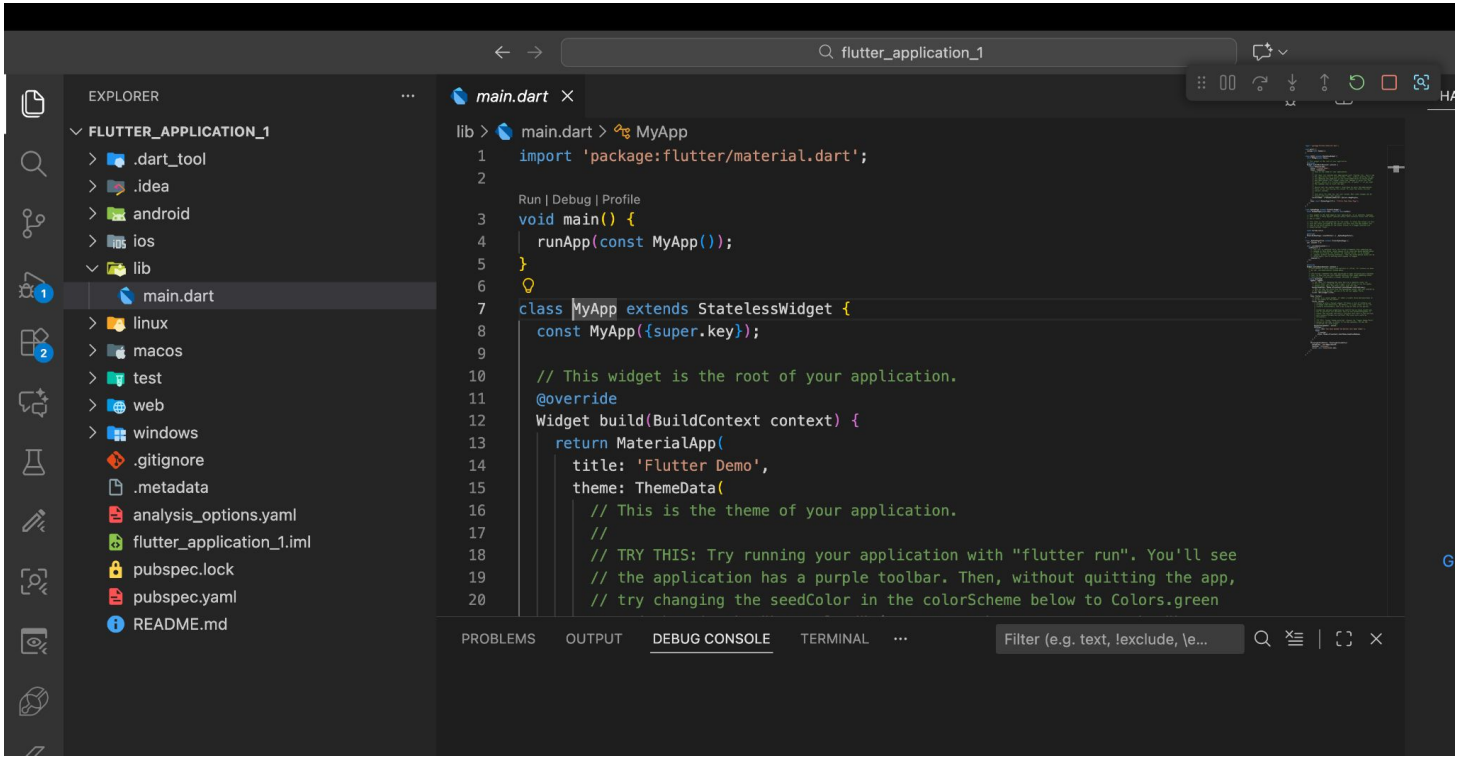
เกณฑ์คะแนน

ส่วนประกอบ (Component)	สัดส่วน (%)	วิธีการวัดผล	CLO
1. Class Engagement / Micro-Challenges & Home work	**20%**	ถาม-ตอบในคลาส / Show & Tell งานในคาบ / เช็คงานท้ายคาบ (Pass/Fail) / การบ้าน	CLO-1, CLO-5
2. Term Project (Group Work)	**40%**	Proposal / Progress Check / Final Demo & Pitching	CLO-2, 3, 4, 5
3. Midterm Exam (Practical Lab)	**20%**	โจทย์สร้าง UI/Logic/คุณภาพ Code ในห้องปฏิบัติการคอมพิวเตอร์	CLO-1, CLO-2
4. Final Exam (Scenario/Case Study)	**20%**	โจทย์ออกแบบ Architecture / แก้ Bug / เขียน Flow (ไม่เน้น Code ยาว)	CLO-1, CLO-3

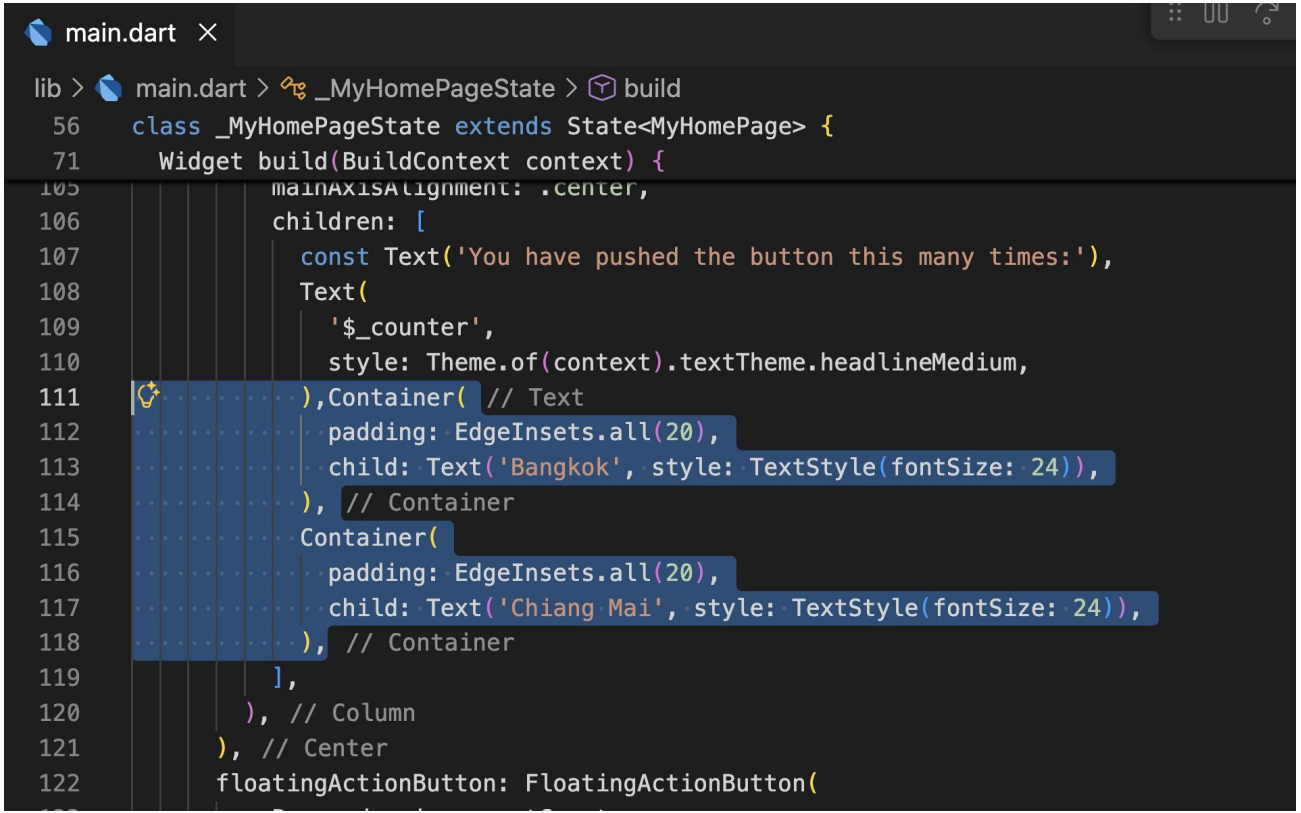
App Structure & Refactoring



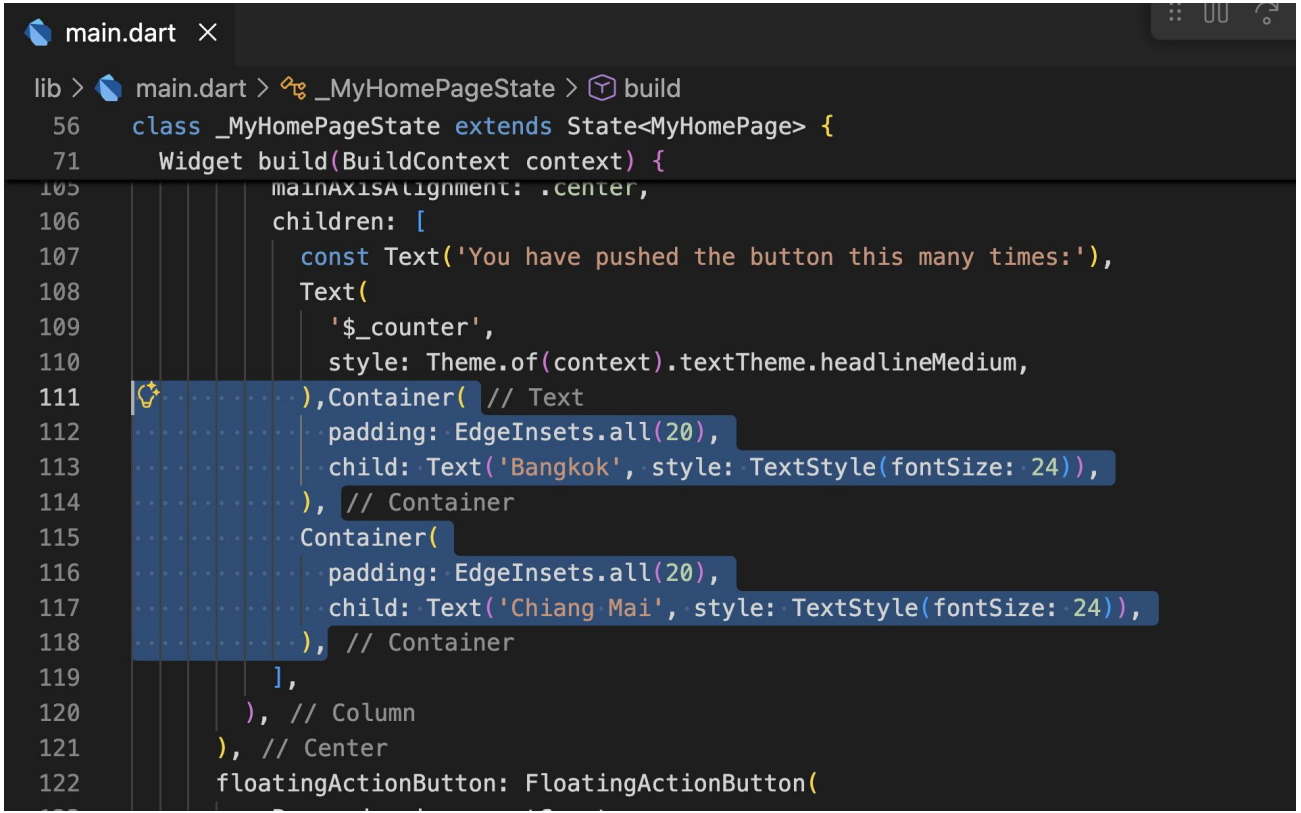
สมมติว่าเรากำลังจะทำแอปแนะนำสถานที่เที่ยว ใน Week ที่แล้ว



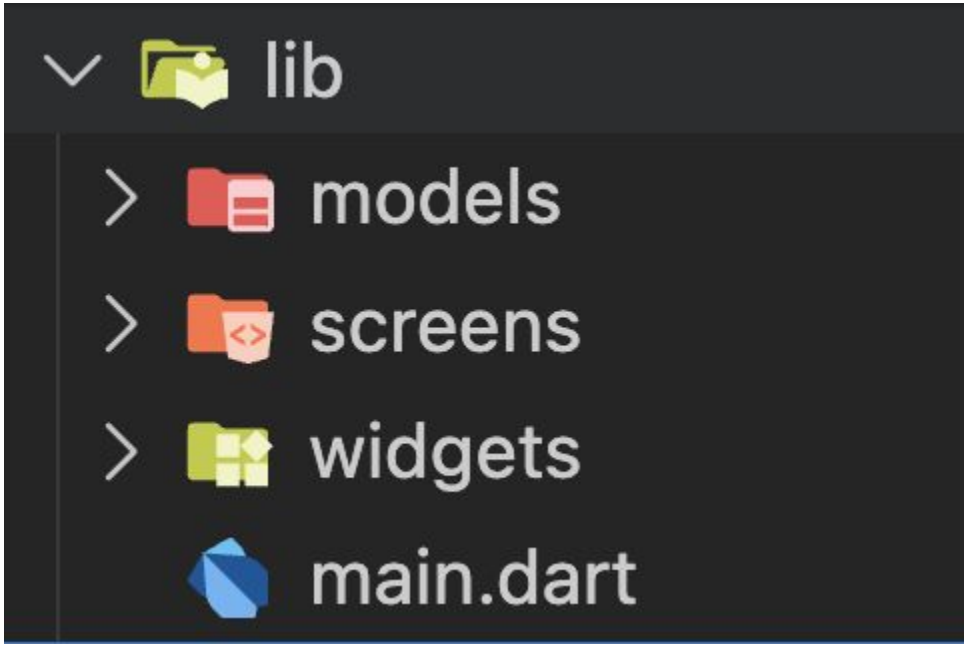
สร้าง Project ใหม่ และ ลอง Run ใหม่



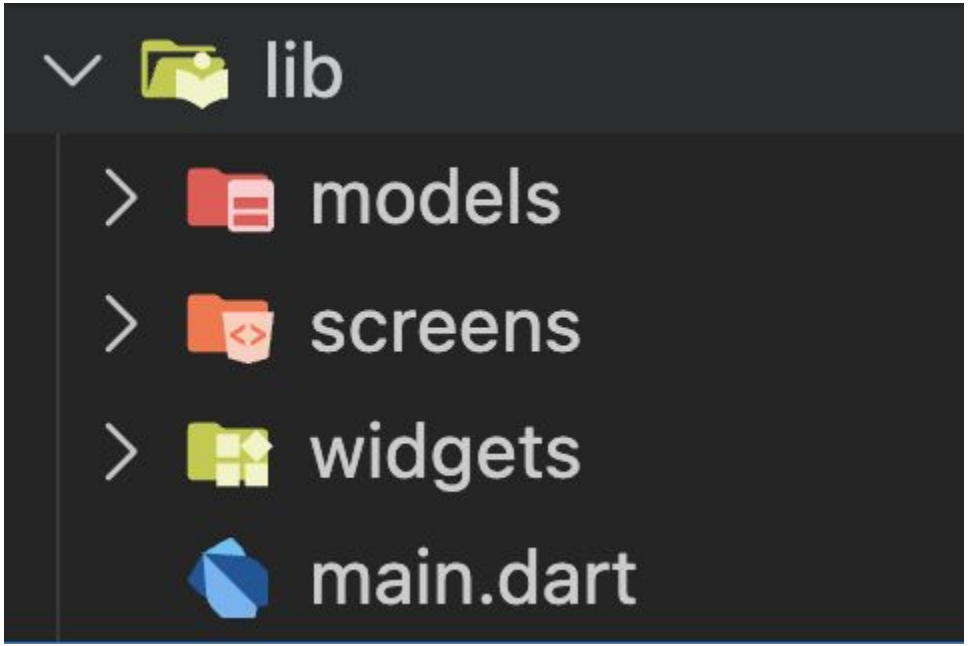
สร้าง Container ที่มี text



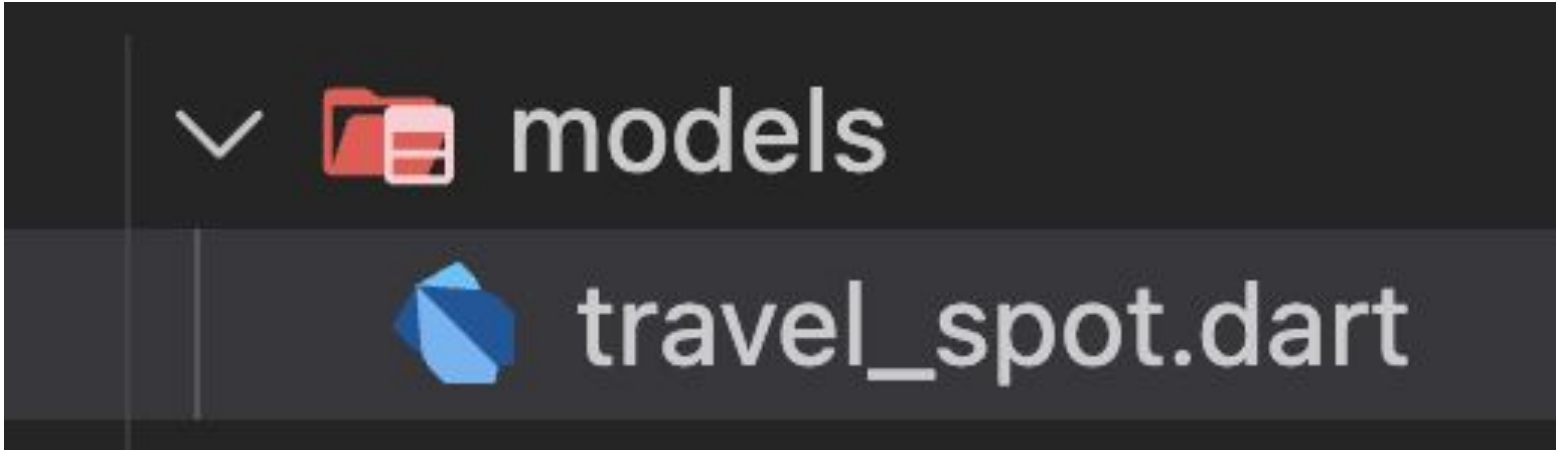
เป็นการ Hardcode ลงไปในโปรแกรม



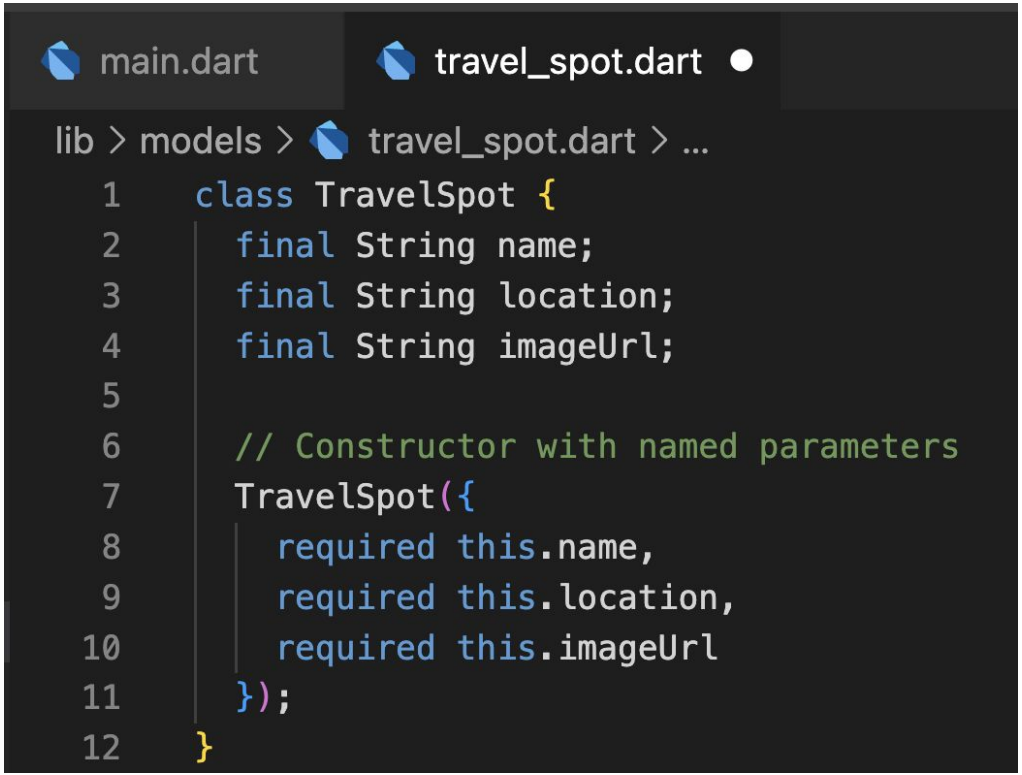
สร้าง Folder ทั้งหมด 3 ตัว



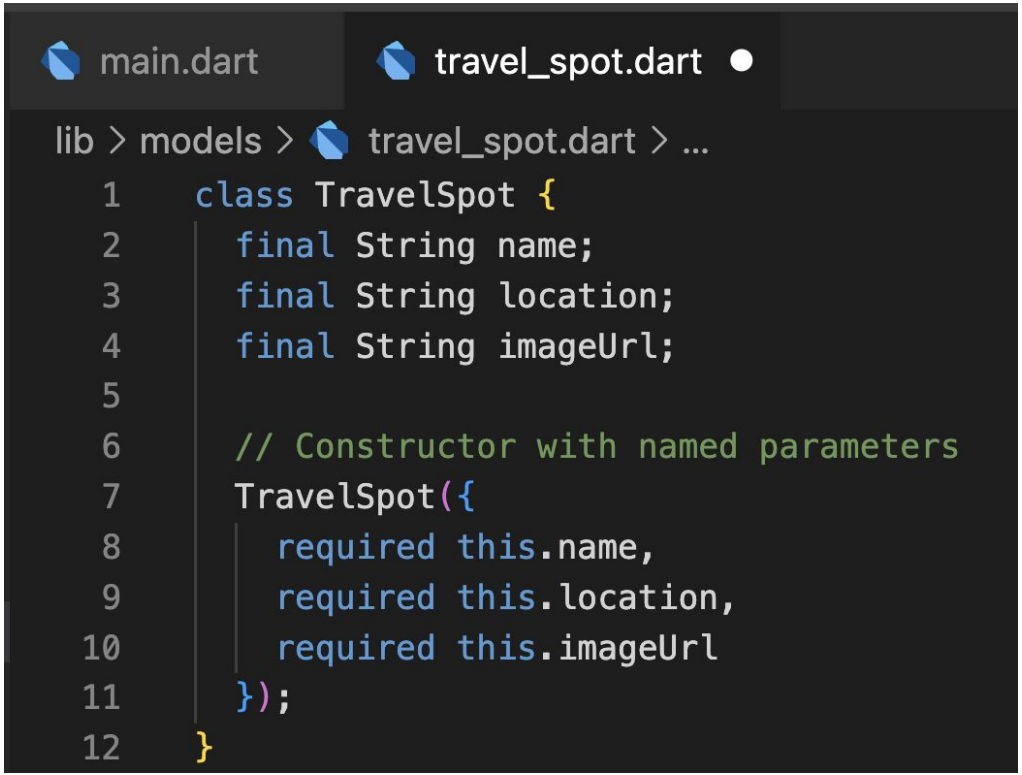
models/ (เก็บพิมพ์เขียวของข้อมูล - *ฝึกเขียน Dart Class*)
screens/ (เก็บหน้าจอหลักทั้งหน้า - *Scaffolds*)
widgets/ (เก็บชิ้นส่วน UI ที่ใช้ซ้ำ - *Reusable Components*)



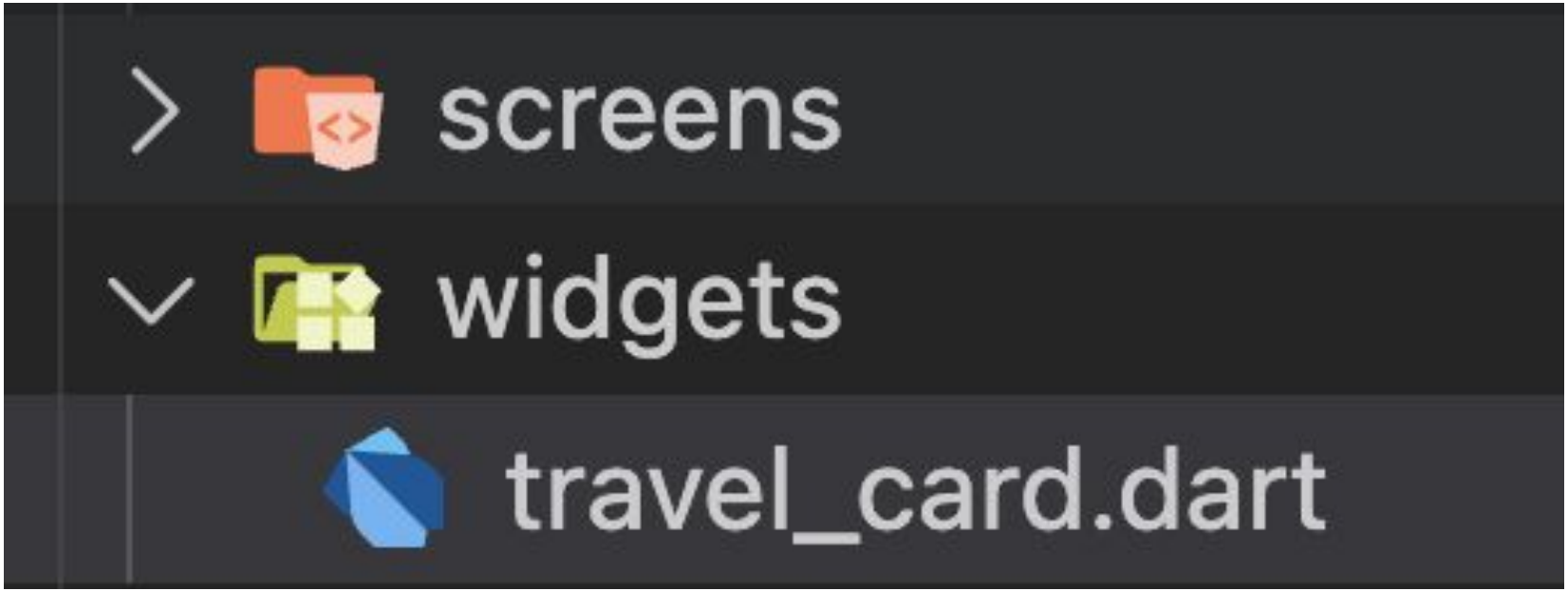
สร้างไฟล์ lib/models/travel_spot.dart



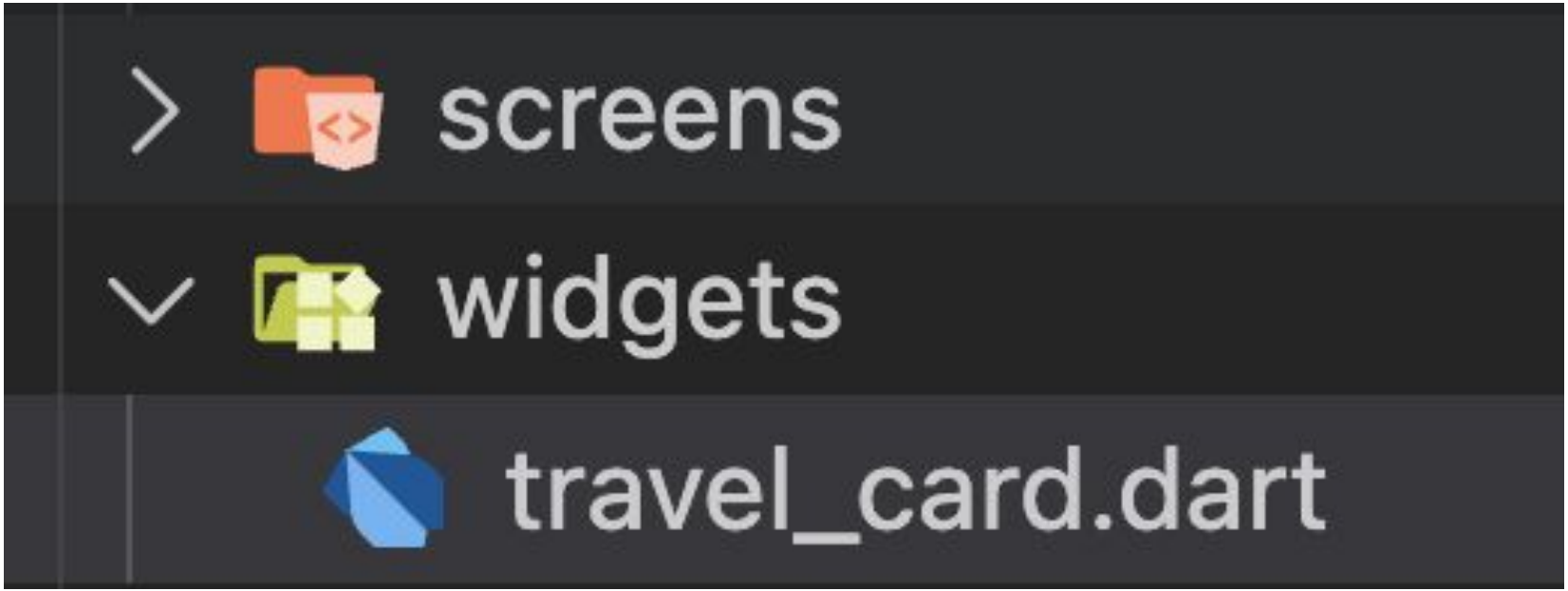
สร้างไฟล์ lib/models/travel_spot.dart



required คือการบังคับว่าต้องส่งค่ามา ห้ามเป็น null



สร้างไฟล์ lib/widgets/travel_card.dart



Quiz : สร้างเพื่ออะไร ?

travel_spot.dart | travel_card.dart | home_screen.dart | main.dart

lib > widgets > travel_card.dart > TravelCard

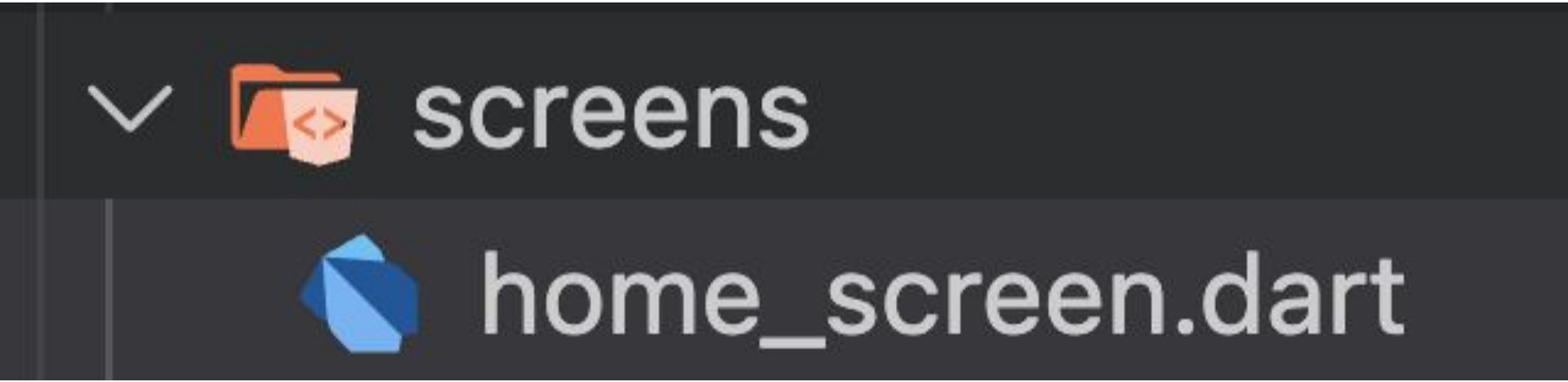
```dart
import 'package:flutter/material.dart';
import '../models/travel_spot.dart'; // อย่าลืม Import Model ที่เราเพิ่งสร้าง

class TravelCard extends StatelessWidget {
  // รับ Data Model เข้ามา แทนที่จะรับ String ทีละตัว
  final TravelSpot spot;

const TravelCard({super.key, required this.spot});

  @override
  Widget build(BuildContext context) {
    return Card(
      margin: EdgeInsets.all(10),
      child: Column(
        children: [
          // แสดงรูปภาพ (สมมติใช้ NetworkImage)
          Image.network(spot.imageUrl, height: 150, fit: BoxFit.cover),
          ListTile(
            title: Text(spot.name, style: TextStyle(fontWeight: FontWeight.bold)),
            subtitle: Text(spot.location),
            trailing: Icon(Icons.arrow_forward),
          ), // ListTile
        ],
      ), // Column
    ); // Card
  }
}
```

travel_spot.dart | travel_card.dart | home_screen.dart | main.dart

lib > widgets > travel_card.dart > TravelCard

```dart
import 'package:flutter/material.dart';
import '../models/travel_spot.dart'; // อย่าลืม Import Model ที่เราเพิ่งสร้าง

class TravelCard extends StatelessWidget {
  // รับ Data Model เข้ามา แทนที่จะรับ String ทีละตัว
  final TravelSpot spot;

const TravelCard({super.key, required this.spot});

```

```dart
const TravelCard({super.key, required this.spot});

  @override
  Widget build(BuildContext context) {
    return Card(
      margin: EdgeInsets.all(10),
      child: Column(
        children: [
          // แสดงรูปภาพ (สมมติใช้ NetworkImage)
          Image.network(spot.imageUrl, height: 150, fit: BoxFit.cover),
          ListTile(
            title: Text(spot.name, style: TextStyle(fontWeight: FontWeight.bold)),
            subtitle: Text(spot.location),
            trailing: Icon(Icons.arrow_forward),
          ), // ListTile
        ],
      ), // Column
    ); // Card
  }
}
```



สร้างไฟล์ lib/screens/home_screen.dart

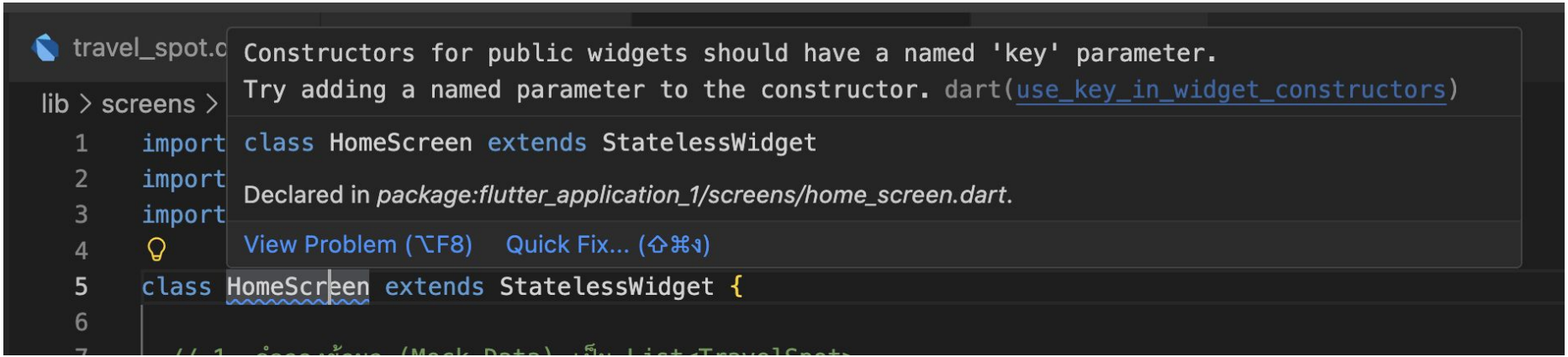
lib > screens > home_screen.dart > ...

```dart
import 'package:flutter/material.dart';
import '../models/travel_spot.dart';
import '../widgets/travel_card.dart';

class HomeScreen extends StatelessWidget {

  // 1. จำลองข้อมูล (Mock Data) เป็น List<TravelSpot>
  final List<TravelSpot> spots = [
    TravelSpot(
      name: 'Grand Palace',
      location: 'Bangkok, Thailand',
      imageUrl: 'https://link_to_image_1.jpg', // หาลกรูปจริงใส่
    ),
    TravelSpot(
      name: 'Doi Inthanon',
      location: 'Chiang Mai, Thailand',
      imageUrl: 'https://link_to_image_2.jpg',
    ),
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Thailand Travel')),
      // 2. ใช้ ListView.builder เพื่อ Loop ข้อมูลที่มีประสิทธิภาพ
      body: ListView.builder(
        itemCount: spots.length,
        itemBuilder: (context, index) {
          // ส่ง Object แต่ละตัวเข้าไปใน Widget แยก
          return TravelCard(spot: spots[index]);
        },
      ), // ListView.builder
    ); // Scaffold
  }
}
```

```dart
import 'package:flutter/material.dart';
import '../models/travel_spot.dart';
import '../widgets/travel_card.dart';

class HomeScreen extends StatelessWidget {
```

```dart
// 1. จำลองข้อมูล (Mock Data) เป็น List<TravelSpot>
final List<TravelSpot> spots = [
  TravelSpot(
    name: 'Grand Palace',
    location: 'Bangkok, Thailand',
    imageUrl: 'https://link_to_image_1.jpg', // หาลกรูปจริงใส่
  ),
  TravelSpot(
    name: 'Doi Inthanon',
    location: 'Chiang Mai, Thailand',
    imageUrl: 'https://link_to_image_2.jpg',
  ),
];
```

```dart
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Thailand Travel')),
      // 2. ใช้ ListView.builder เพื่อ Loop ข้อมูลที่มีประสิทธิภาพ
      body: ListView.builder(
        itemCount: spots.length,
        itemBuilder: (context, index) {
          // ส่ง Object แต่ละตัวเข้าไปใน Widget แยก
          return TravelCard(spot: spots[index]);
        },
      ), // ListView.builder
    ); // Scaffold
  }
}

```



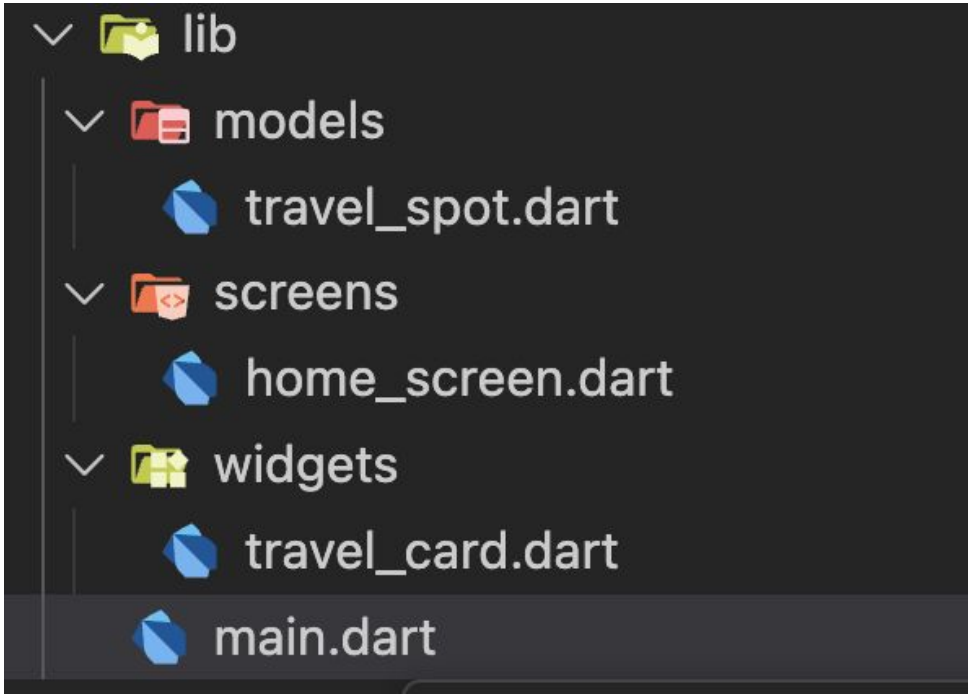
สังเกต Warning : Constructors for public widgets should have a named 'key' parameter

```
21    HomeScreen({super.key});
22    @override
23    Widget build(BuildContext context) {
24      return Scaffold(
25        appBar: AppBar(title: Text('Thailand Tr
```

ใส่ const HomeScreen({super.key});

```
21    HomeScreen({super.key});
22    @override
23    Widget build(BuildContext context) {
24      return Scaffold(
25        appBar: AppBar(title: Text('Thailand Tr
```

เปิดช่องให้ส่งค่า Key เข้ามาได้เสมอ เผื่อกรณีที่ต้องมีการสลับ
ตำแหน่ง Widget หรือจัดการ State
ถึงแม้ตอนนี้เราจะยังไม่ได้ใช้ก็ตาม แต่เป็นมาตรฐานที่ต้องมี



กลับมาที่ main.dart ลบขยะทิ้งให้หมด ให้เหลือแค่ทางเข้า (Entry Point)

main.dart | travel_spot.dart | travel_card.dart | home_screen.dart

lib > main.dart > ...

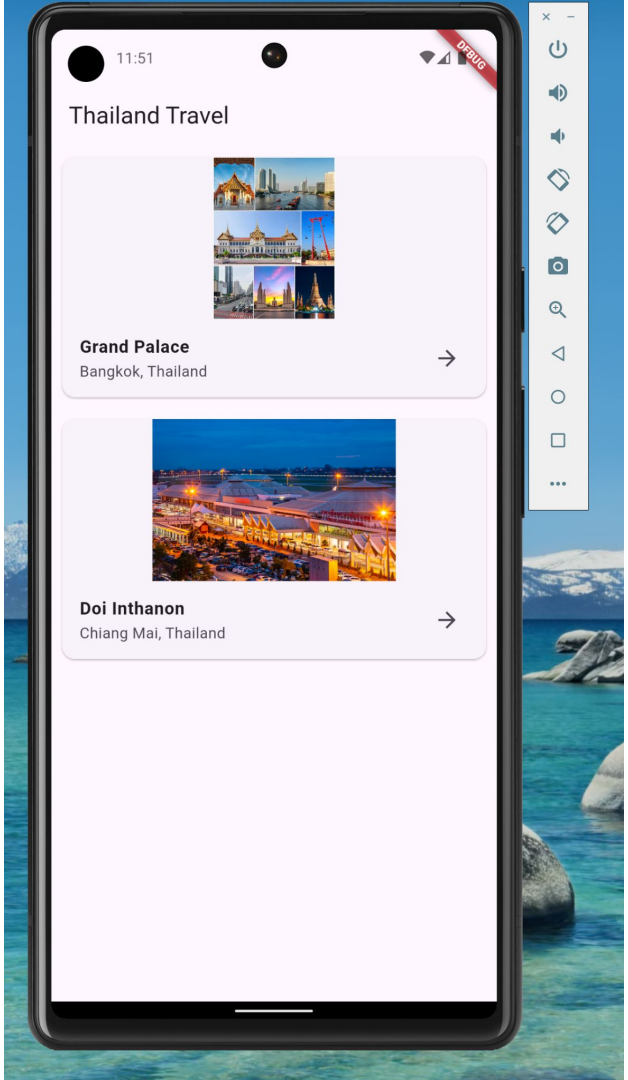
```dart
import 'package:flutter/material.dart';
import 'screens/home_screen.dart'; // Import หน้า Home

Run | Debug | Profile
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Travel App',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: HomeScreen(), // เรียกใช้ HomeScreen ที่เราแยกไปแล้ว
    ); // MaterialApp
  }
}

```

travel_spot.dart | travel_card.dart | home_screen.dart | main.dart ●

lib > main.dart > ...

```dart
import 'package:flutter/material.dart';
import 'screens/home_screen.dart'; // Import หน้า Home

Run | Debug | Profile
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Travel App',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: HomeScreen(), // เรียกใช้ HomeScreen ที่เราแยกไปแล้ว
    ); // MaterialApp
  }
}
```

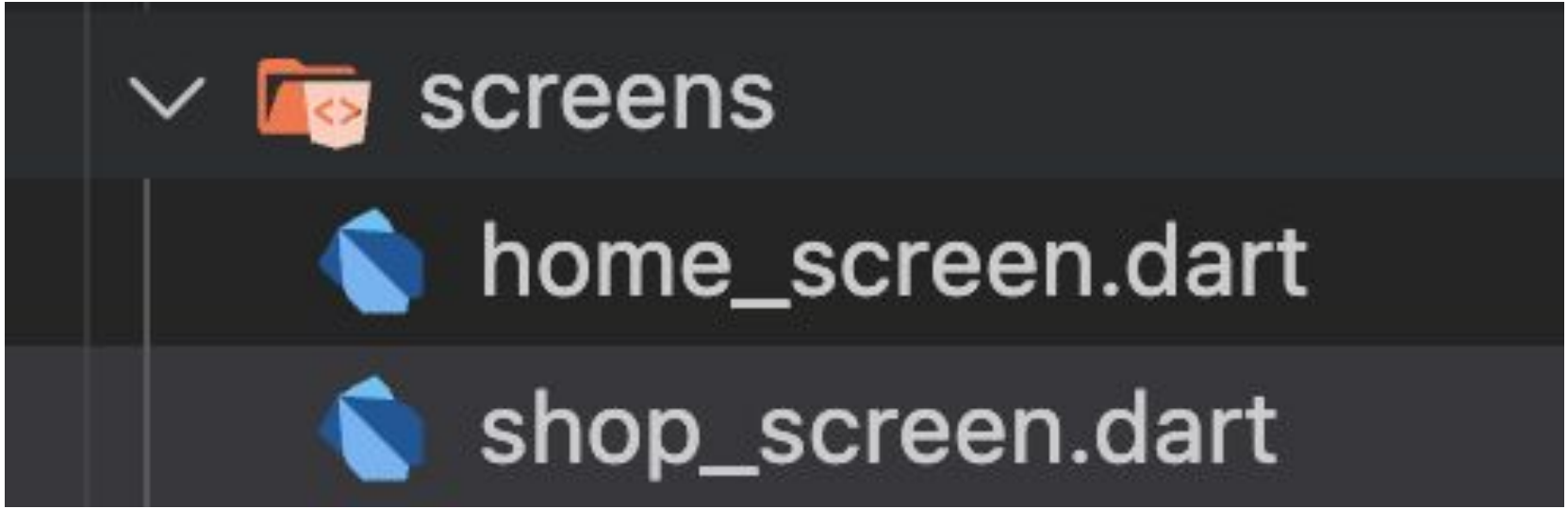
BORNTO DEV


11:51
DEBUG
Thailand Travel
Grand Palace
Bangkok, Thailand
Doi Inthanon
Chiang Mai, Thailand

Refactoring UI

https://bornto.dev/FlutterKMITL68

🛑 Step 0: The "Messy" Code



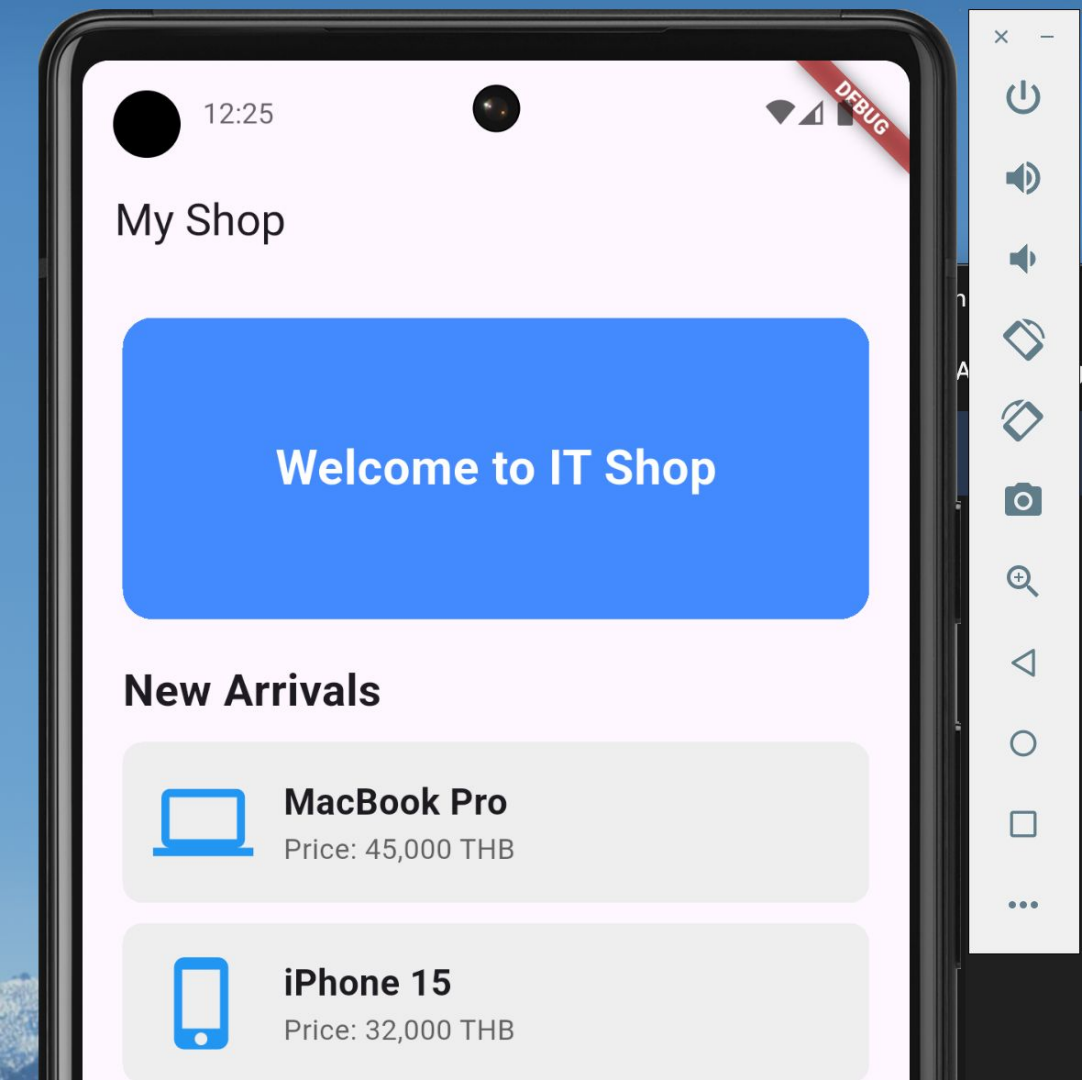
สร้างไฟล์ : lib/screens/shop_screen.dart

shop_screen.dart | travel_spot.dart | travel_card.dart | home_screen.dart | main.dart

lib > screens > shop_screen.dart > ShopScreen > build

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

class ShopScreen extends StatelessWidget {
  const ShopScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('My Shop')),
      body: ListView(
        padding: EdgeInsets.all(20),
        children: [
          // --- ส่วนที่ 1: Header (Banner) ---
          Container(
            height: 150,
            width: double.infinity,
            decoration: BoxDecoration(
              color: Colors.blueAccent,
              borderRadius: BorderRadius.circular(15),
            ), // BoxDecoration
            child: Center(
              child: Text(
                'Welcome to IT Shop',
                style: TextStyle(color: Colors.white, fontSize: 24, fontWeight: FontWeight.bold),
              ), // Text
            ), // Center
          ), // Container
          SizedBox(height: 20),

          // --- ส่วนที่ 2: Section Title ---
          Text('New Arrivals', style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold)),
          SizedBox(height: 10),

          // --- ส่วนที่ 3: Product Items (Card) ---
```

BORNTO DEV



Quiz : ทำให้หน้าต่างแสดงออกมา

shop_screen.dart | travel_spot.dart | travel_card.dart | home_screen.dart | main.dart 1 ●

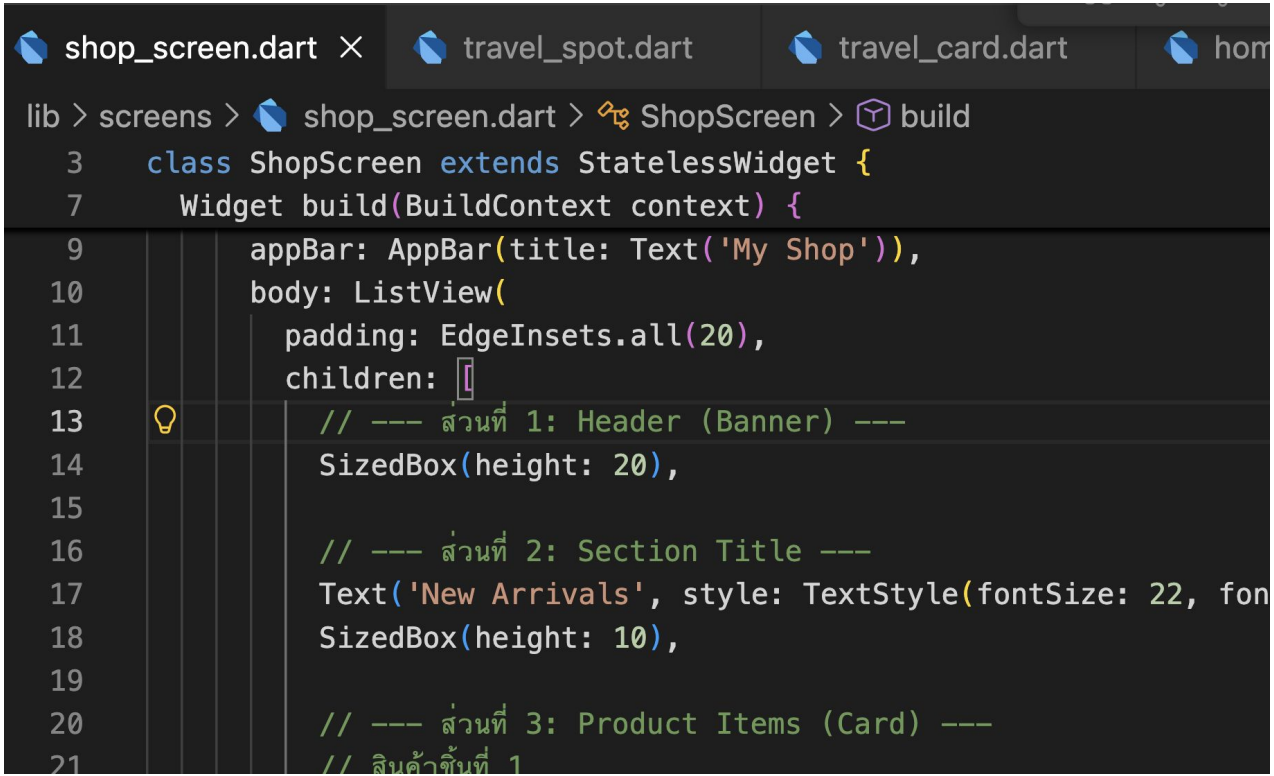
lib > main.dart > MyApp > build

```dart
import 'package:flutter/material.dart';
import 'screens/home_screen.dart'; // Import หน้า Home
import 'screens/shop_screen.dart';

Run | Debug | Profile
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Travel App',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: ShopScreen(), // เรียกใช้ HomeScreen ที่เราแยกไปแล้ว
    ); // MaterialApp
  }
}
```

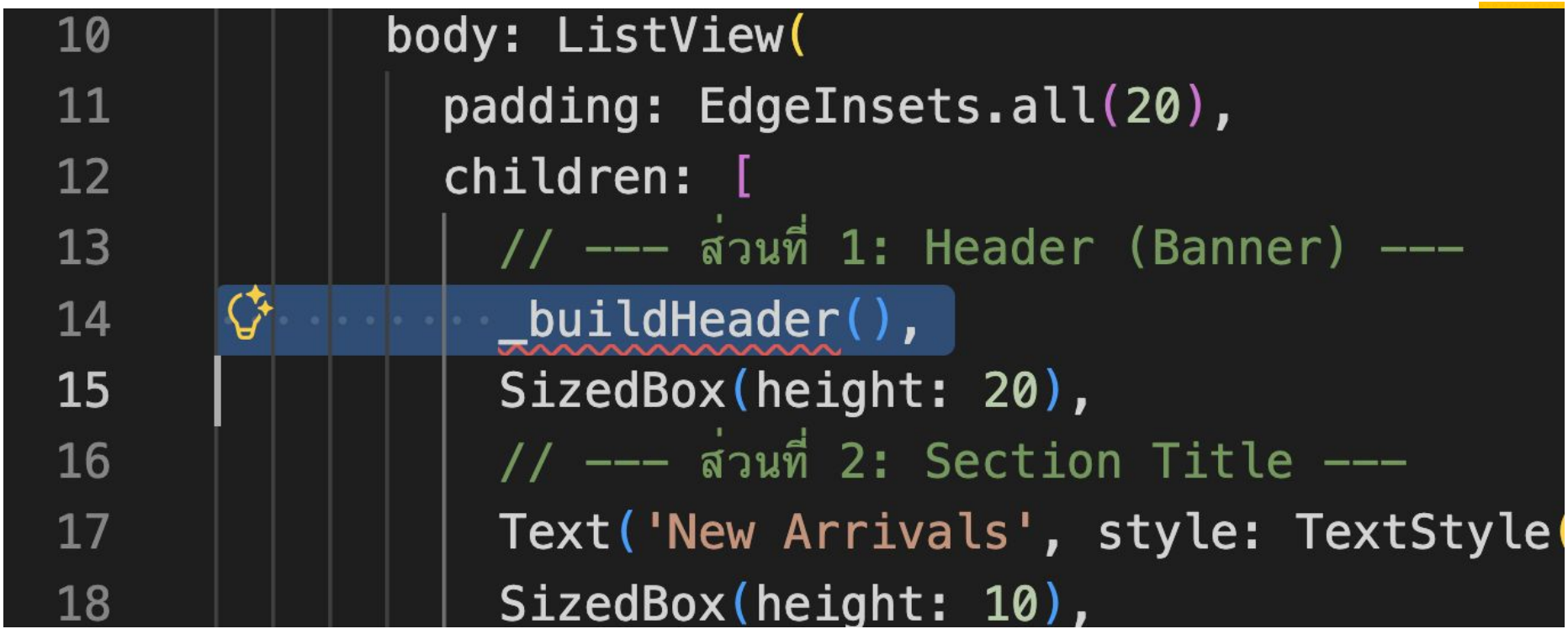
✂️ Step 1: Extract Method

```
13          // --- ส่วนที่ 1: Header (Banner) ---
14          Container(
15            height: 150,
16            width: double.infinity,
17            decoration: BoxDecoration(
18              color: Colors.blueAccent,
19              borderRadius: BorderRadius.circular(15),
20            ), // BoxDecoration
21            child: Center(
22              child: Text(
23                'Welcome to IT Shop',
24                style: TextStyle(color: Colors.white, fontSize: 24, fontWeight: FontWeight.bol
25              ), // Text
26            ), // Center
27          ), // Container
```

Ctrl + x


shop_screen.dart
travel_spot.dart
travel_card.dart
lib > screens > shop_screen.dart > ShopScreen > build
class ShopScreen extends StatelessWidget {
Widget build(BuildContext context) {
appBar: AppBar(title: Text('My Shop')),
body: ListView(
padding: EdgeInsets.all(20),
children: [
// --- ส่วนที่ 1: Header (Banner) ---
SizedBox(height: 20),
// --- ส่วนที่ 2: Section Title ---
Text('New Arrivals', style: TextStyle(fontSize: 22, fon
SizedBox(height: 10),
// --- ส่วนที่ 3: Product Items (Card) ---

ตัดทิ้ง

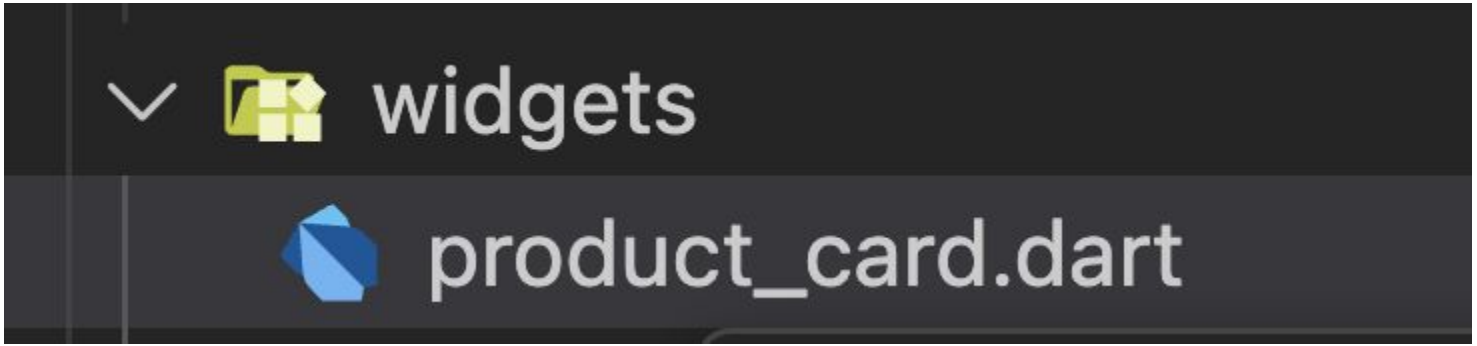


Call Function

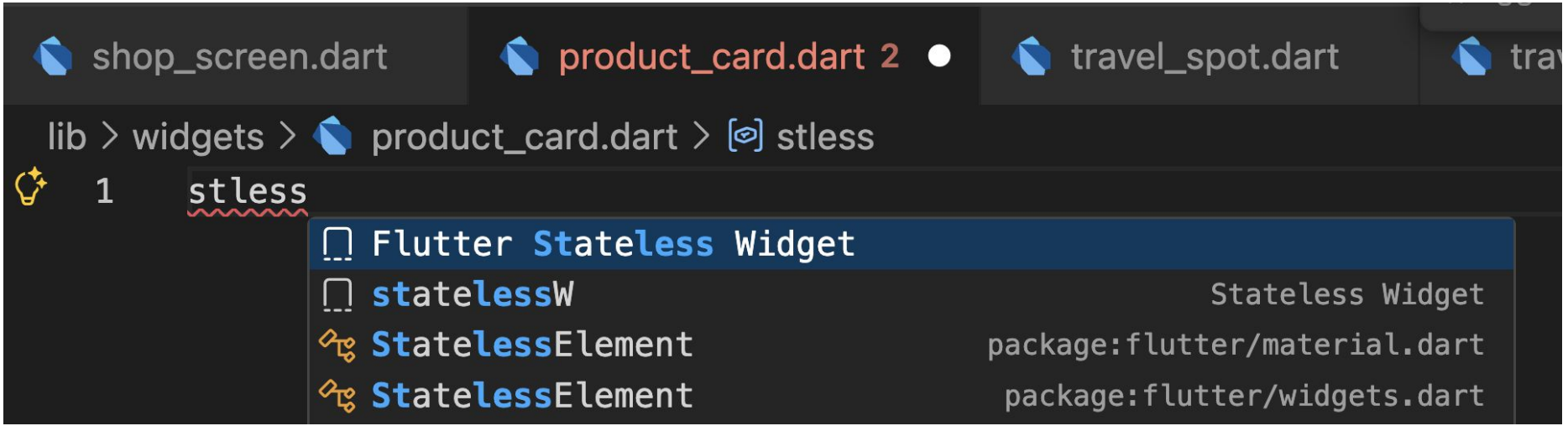
```dart
Widget _buildHeader() {
  return Container(
    height: 150,
    decoration: BoxDecoration(
      color: const Color.fromARGB(255, 33, 149, 243),
      borderRadius: BorderRadius.circular(10),
    ), // BoxDecoration
    alignment: Alignment.center,
    child: Text(
      'Welcome to My Shop',
      style: TextStyle(color: Colors.white, fontSize: 24, fontWeight: FontWeight.bold),
    ), // Text
  ); // Container
}
}
```

สร้าง Widget

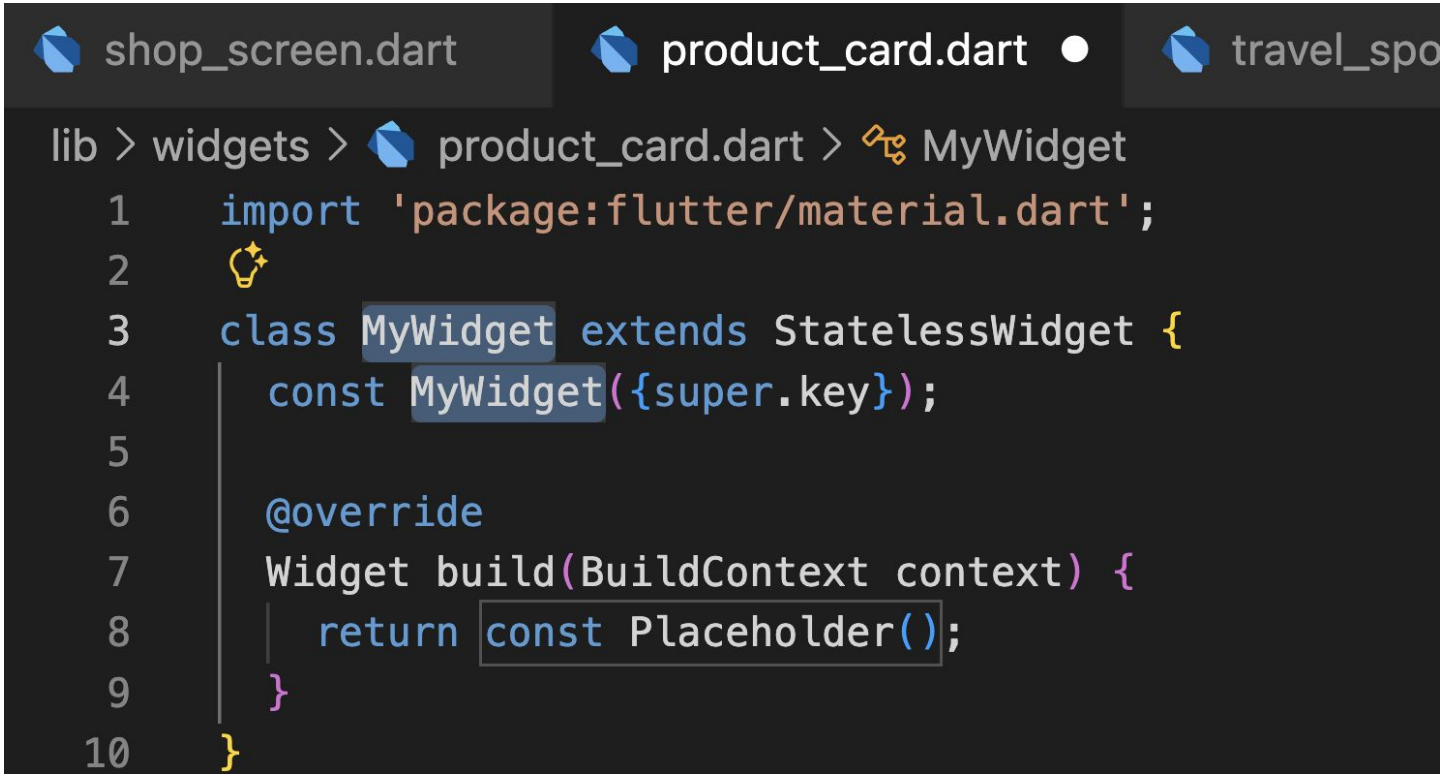
📦 Step 2: Extract Widget (สร้าง Class ใหม่เพื่อใช้ซ้ำ)



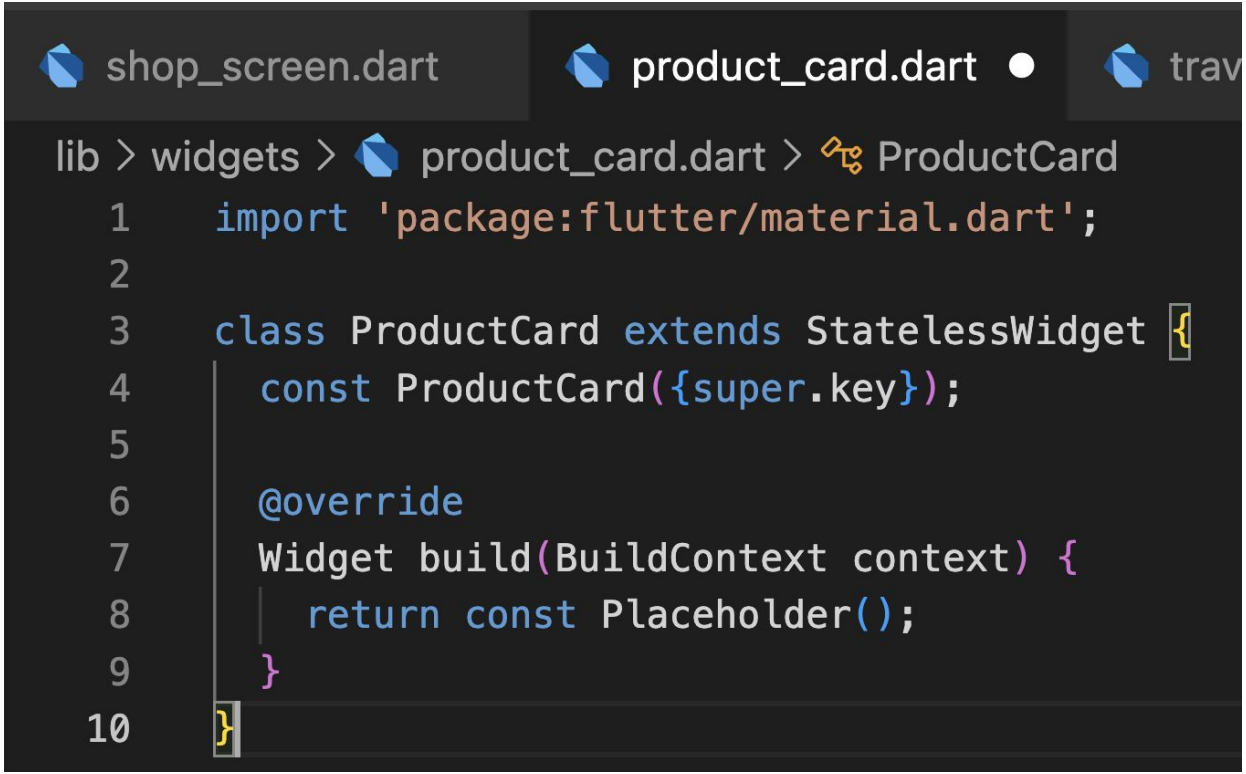
สร้างไฟล์ใหม่ lib/widgets/product_card.dart



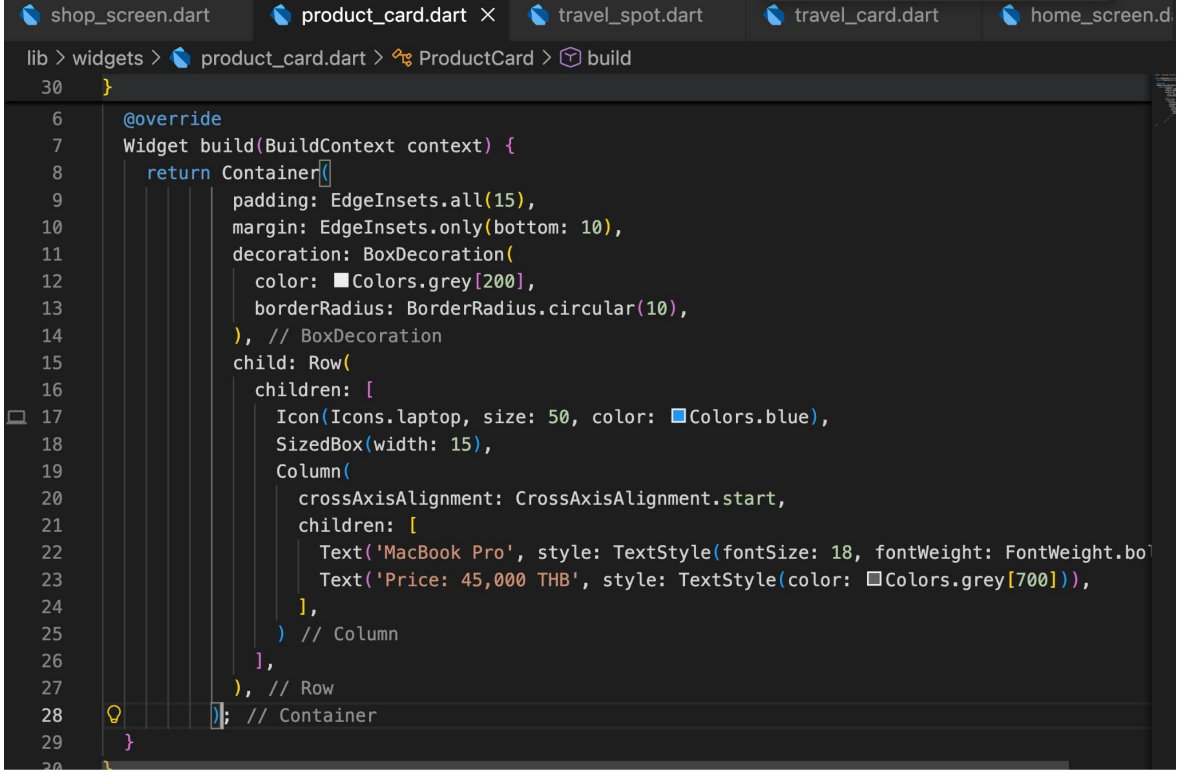
พิมพ์ stless แล้ว Enter เพื่อสร้าง Class เปล่า
ชื่อ ProductCard



พิมพ์ stless แล้ว Enter เพื่อสร้าง Class เปล่า ชื่อ ProductCard



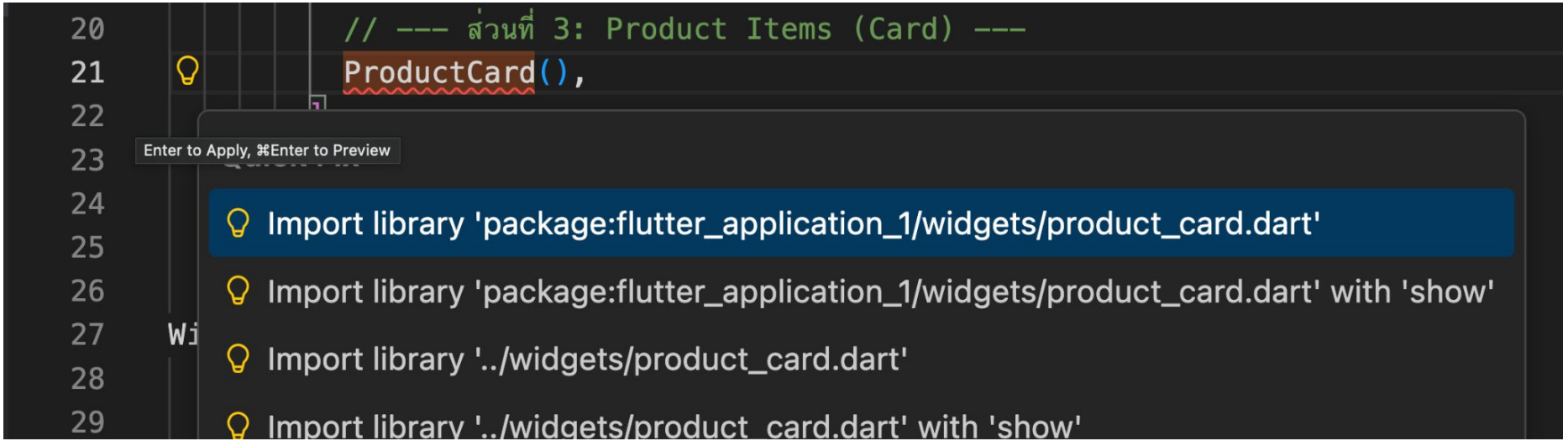
พิมพ์ stless แล้ว Enter เพื่อสร้าง Class เปล่า ชื่อ ProductCard



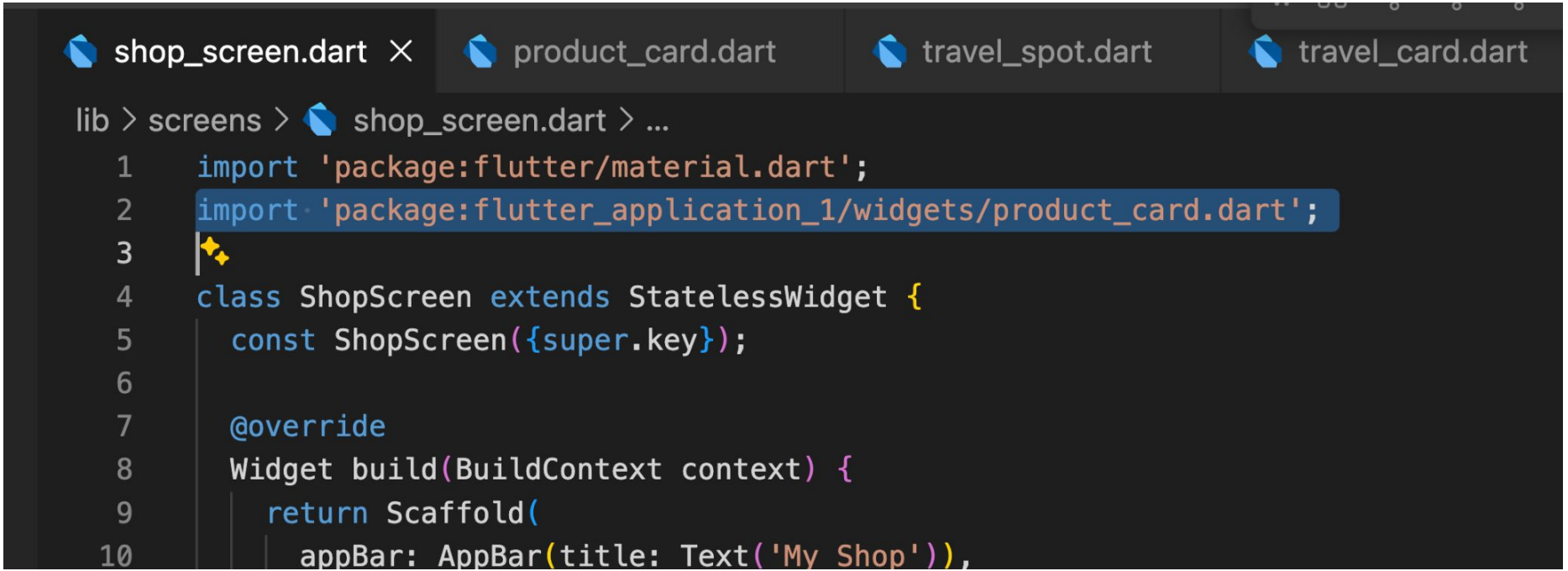
Copy ส่วน Code Container สินค้าจากหน้าเดิม มาวางใน build ของไฟล์ใหม่

```
19
20          // --- ส่วนที่ 3: Product Items (Card) ---
21          ProductCard(),
22        ],
```

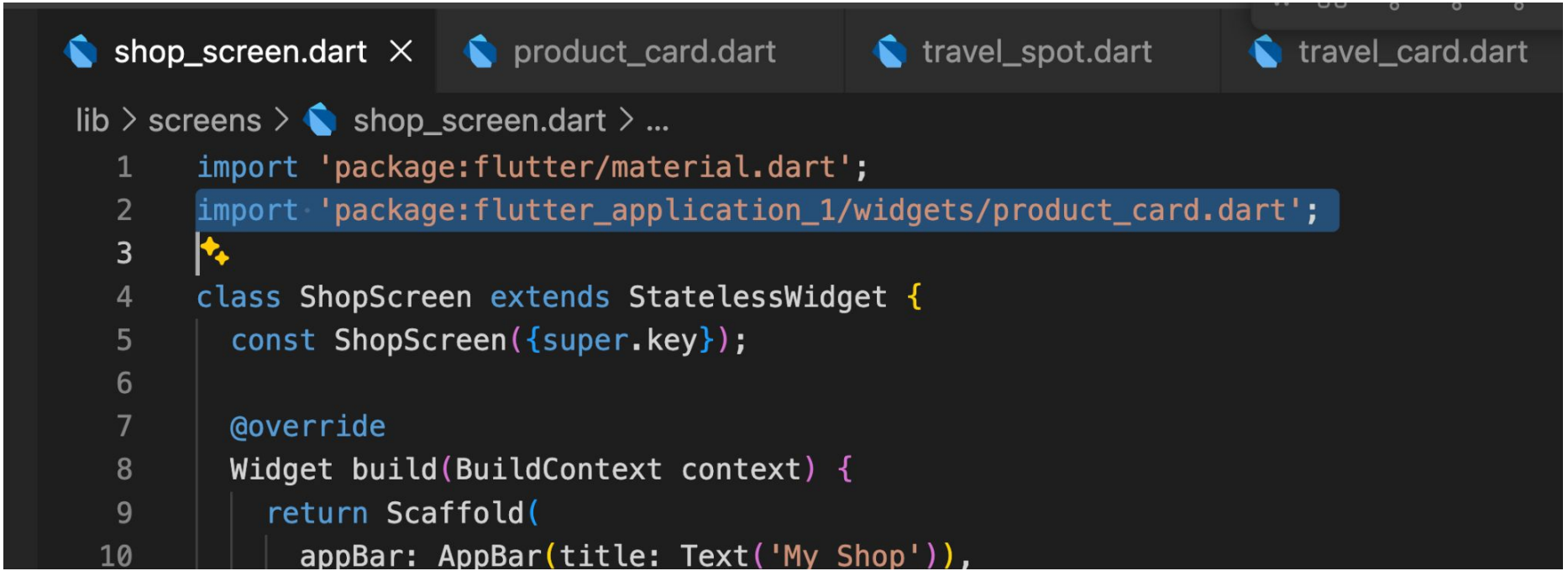
เอามาโชว์ให้มีกำลังใจ



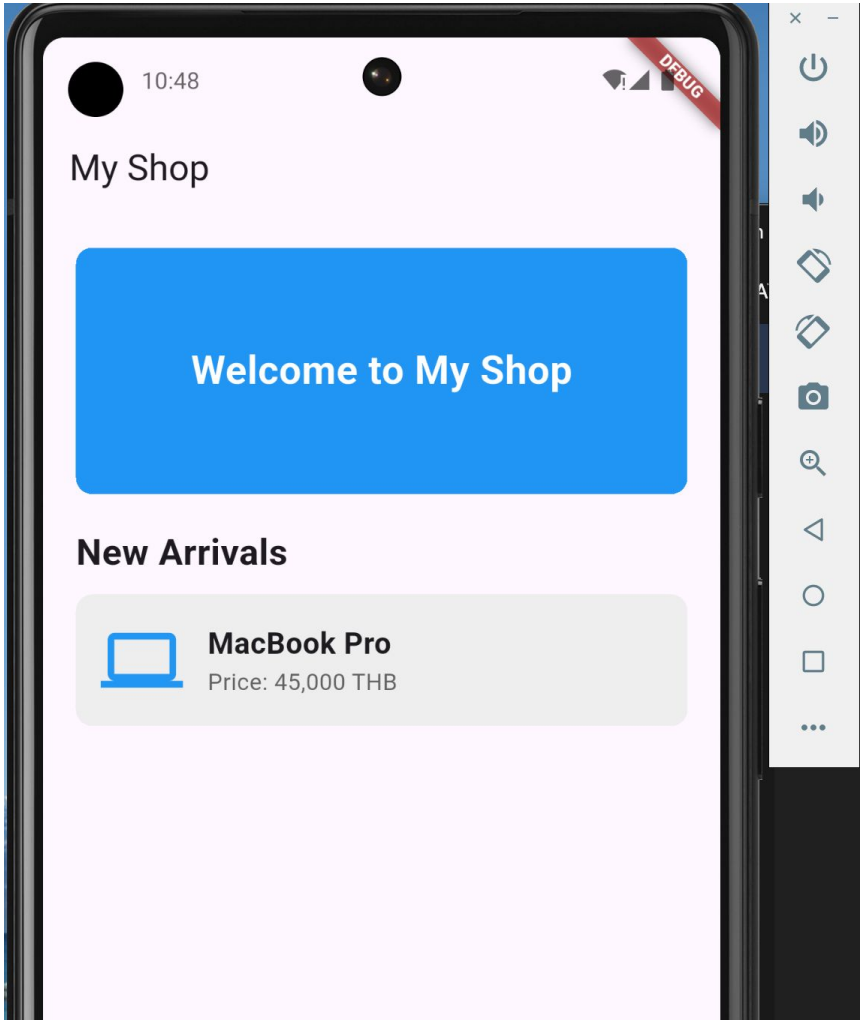
เอามาโชว์ให้มีกำลังใจ



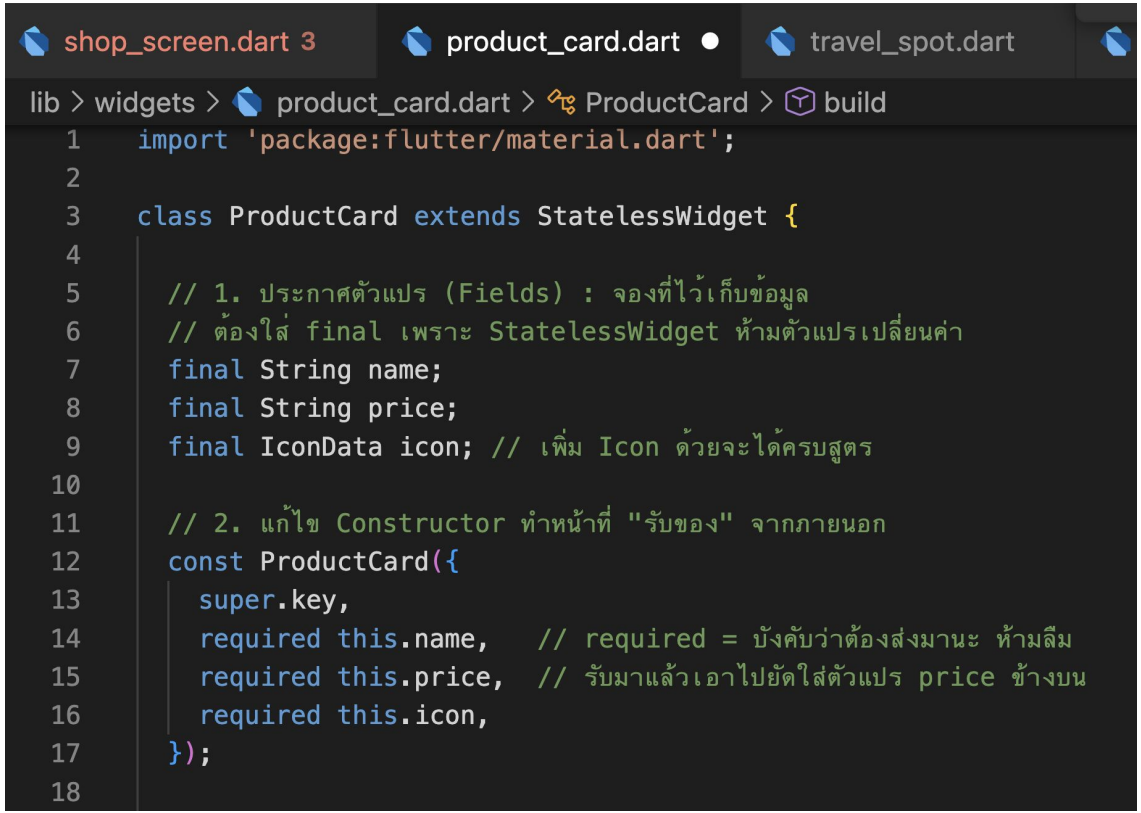
เอามาโชว์ให้มีกำลังใจ



เอามาโชว์ให้มีกำลังใจ


10:48
DEBUG
My Shop
Welcome to My Shop
New Arrivals
MacBook Pro
Price: 45,000 THB

🧠 Step 3: Parameter & Constructor (หัวใจสำคัญ)



ประกาศตัวแปรรับค่า ใช้ final เพราะ Widget เป็น immutable

shop_screen.dart 3 | product_card.dart ● | travel_spot.dart

lib > widgets > product_card.dart > ProductCard > build

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

class ProductCard extends StatelessWidget {

  // 1. ประกาศตัวแปร (Fields) : จองที่ไว้เก็บข้อมูล
  // ต้องใส่ final เพราะ StatelessWidget ห้ามตัวแปรเปลี่ยนค่า
  final String name;
  final String price;
  final IconData icon; // เพิ่ม Icon ด้วยจะได้ครบสูตร

  // 2. แก้ไข Constructor ทำหน้าที่ "รับของ" จากภายนอก
  const ProductCard({
    super.key,
    required this.name,   // required = บังคับว่าต้องส่งมานะ ห้ามลืม
    required this.price,  // รับมาแล้วเอาไปยัดใส่ตัวแปร price ข้างบน
    required this.icon,
  });

```

```dart
@override
Widget build(BuildContext context) {
  return Container(
    padding: EdgeInsets.all(15),
    margin: EdgeInsets.only(bottom: 10),
    decoration: BoxDecoration(
      color: Colors.grey[200],
      borderRadius: BorderRadius.circular(10),
    ), // BoxDecoration
    child: Row(
      children: [
        Icon(icon, size: 50, color: Colors.blue), // ใช้ Icon ที่รับมา
        SizedBox(width: 15),
        Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(name, style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold)),// ใช้ name ที่รับมา
            Text('Price: $price', style: TextStyle(color: Colors.grey[700])),// ใช้ price ที่รับมา
          ],
        ) // Column
      ],
    ), // Row
  ); // Container
}
```

ใช้ของที่รับมาแทนการ Hardcode

Step 4: Combine Everything

```
// --- ส่วนที่ 3: Product Items (Card) ---
ProductCard(
  name: 'MacBook Pro',
  price: '450,000 THB',
  icon: Icons.laptop,
), // ProductCard
]
```

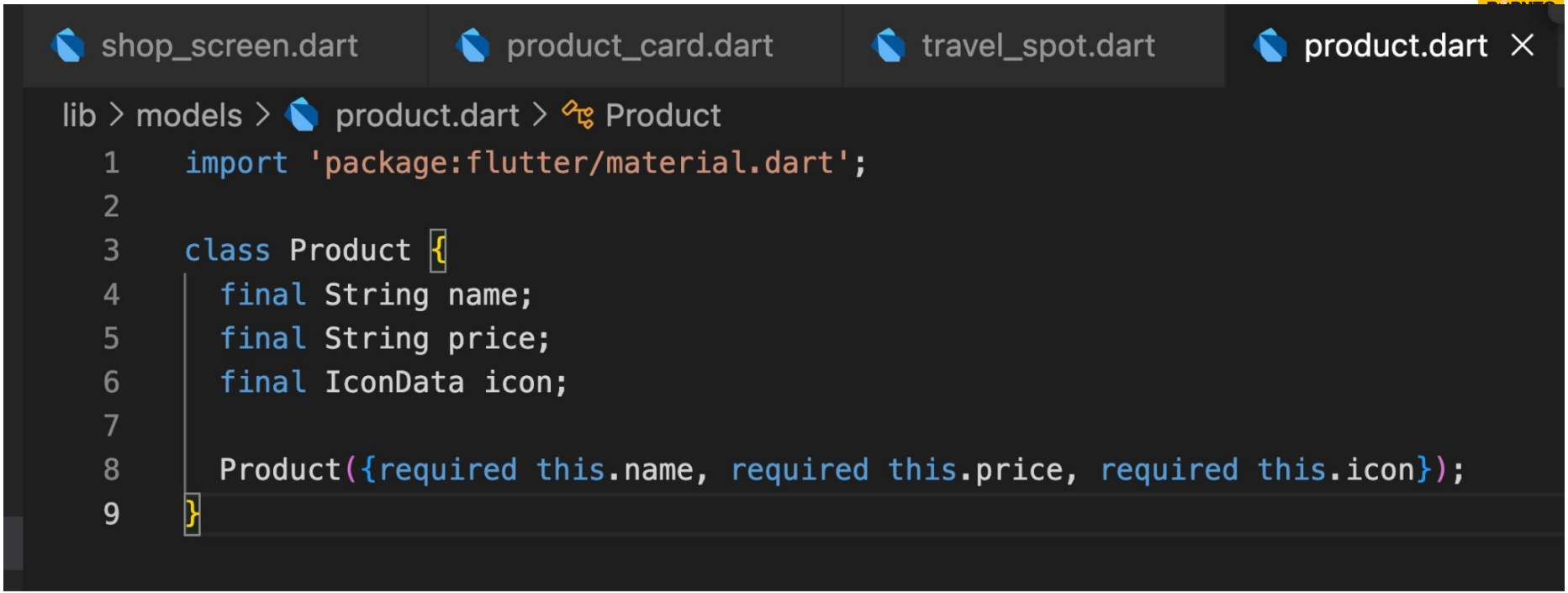
ใช้ของที่รับมาแทนการ Hardcode

```
// --- ส่วนที่ 3: Product Items (Card) ---
ProductCard(
  name: 'MacBook Pro',
  price: '450,000 THB',
  icon: Icons.laptop,
), // ProductCard
]
```

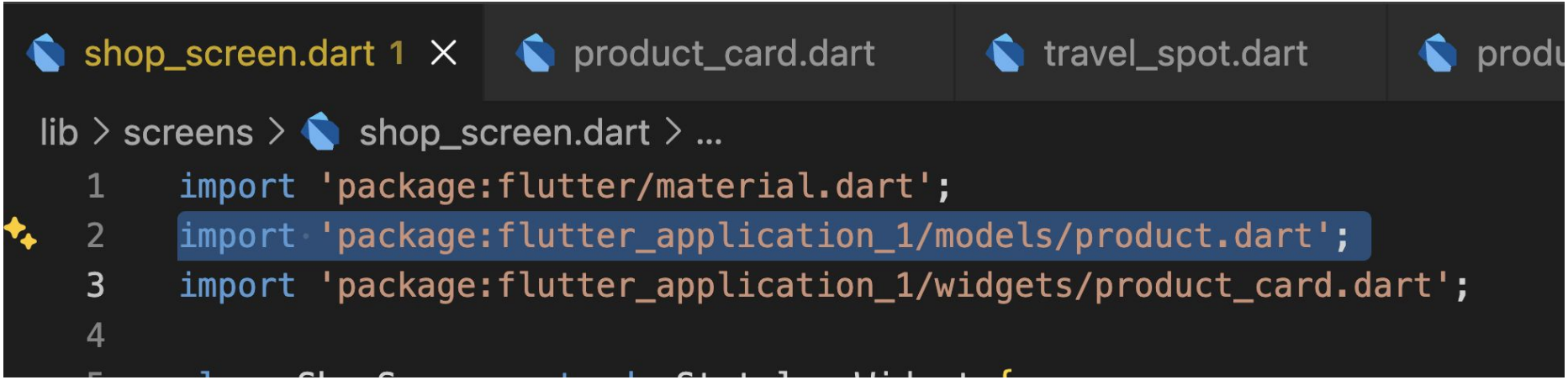
Next Challenge: "ลองเพิ่มสินค้าอีก 2 ชิ้น และลองเปลี่ยนสี ProductCard โดยแก้ที่ไฟล์เดียวดูสิ

```
// --- ส่วนที่ 3: Product Items (Card) ---
ProductCard(
  name: 'MacBook Pro',
  price: '450,000 THB',
  icon: Icons.laptop,
), // ProductCard
]
```

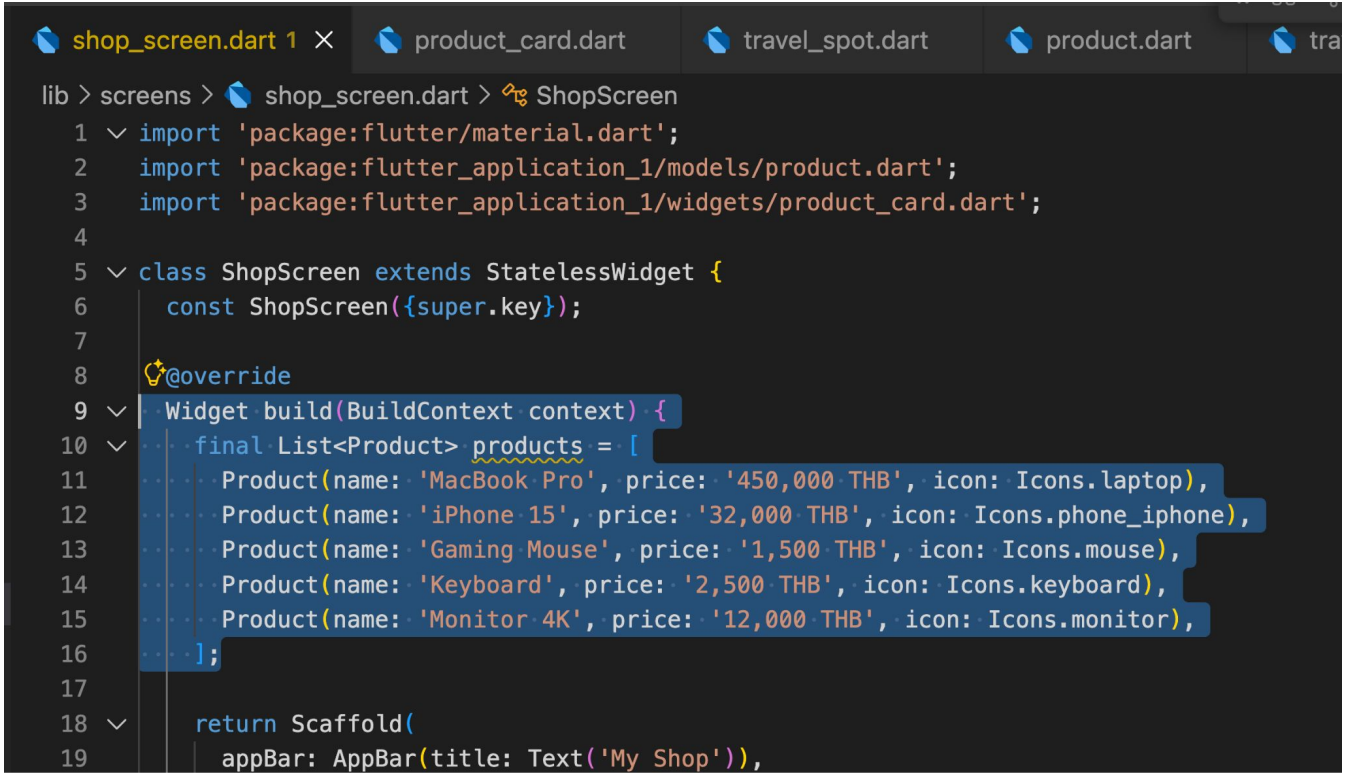
Next Challenge: "ลองเพิ่มสินค้าอีก N ชิ้น ใน List แล้วแสดงออกมา”



แยกไฟล์ models/product.dart



ใน ShopScreen ให้สร้างตัวแปร List ขึ้นมาเพื่อเก็บข้อมูลสินค้า N ชิ้นที่เราต้องการ



ใน ShopScreen ให้สร้างตัวแปร List ขึ้นมาเพื่อเก็บข้อมูลสินค้า N ชิ้นที่เราต้องการ

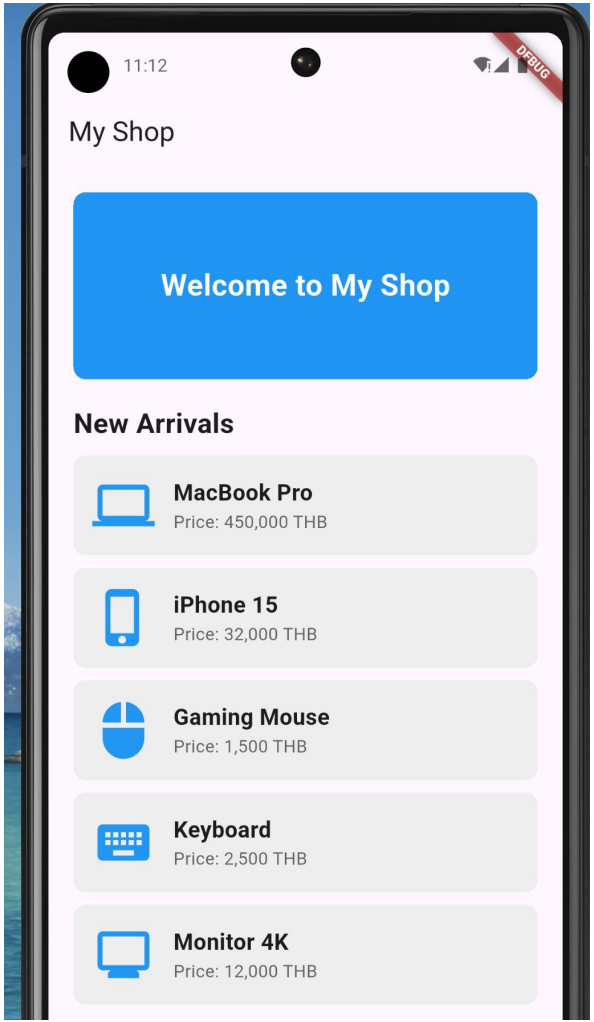
shop_screen.dart 1 | product_card.dart | travel_spot.dart | product.dart | tra

lib > screens > shop_screen.dart > ShopScreen

```dart
import 'package:flutter/material.dart';
import 'package:flutter_application_1/models/product.dart';
import 'package:flutter_application_1/widgets/product_card.dart';

class ShopScreen extends StatelessWidget {
  const ShopScreen({super.key});

  @override
  Widget build(BuildContext context) {
    final List<Product> products = [
      Product(name: 'MacBook Pro', price: '450,000 THB', icon: Icons.laptop),
      Product(name: 'iPhone 15', price: '32,000 THB', icon: Icons.phone_iphone),
      Product(name: 'Gaming Mouse', price: '1,500 THB', icon: Icons.mouse),
      Product(name: 'Keyboard', price: '2,500 THB', icon: Icons.keyboard),
      Product(name: 'Monitor 4K', price: '12,000 THB', icon: Icons.monitor),
    ];

    return Scaffold(
      appBar: AppBar(title: Text('My Shop')),
```

```
// --- ส่วนที่ 3: Product Items (Card) ---
for (var product in products)
  ProductCard(
    name: product.name,
    price: product.price,
    icon: product.icon,
  ), // ProductCard
],
```

ใน children ของ ListView แทนที่เราจะวาง ProductCard ทีละอัน เราจะใช้ Collection For (การวนลูปใน List ของ Widget) แทน


11:12
DEBUG
My Shop
Welcome to My Shop
New Arrivals
MacBook Pro
Price: 450,000 THB
iPhone 15
Price: 32,000 THB
Gaming Mouse
Price: 1,500 THB
Keyboard
Price: 2,500 THB
Monitor 4K
Price: 12,000 THB

อยากย้ายหน้าไปมา ทำอย่างไร?

AI solutions

User interface

Introduction

Widget catalog

Layout

Adaptive & responsive design

Design & theming

Interactivity

Assets & media

Navigation & routing

Overview

Add tabs to your app

Navigate to a new screen and back

Send data to a new screen

Return data from a screen

Add a drawer to a screen

Set up deep linking

Set up app links for Android

Flutter 3.38 and Dart 3.10 are here! Learn more

Navigate to a new screen and back

How to navigate between routes.

Most apps contain several screens for displaying different types of information. For example, an app might have a screen that displays products. When the user taps the image of a product, a new screen displays details about the product.

> **Terminology**
>
> In Flutter, *screens* and *pages* are called *routes*. The remainder of this recipe refers to routes.

In Android, a route is equivalent to an `Activity`. In iOS, a route is equivalent to a `ViewController`. In Flutter, a route is just a widget.

This recipe uses the `Navigator` to navigate to a new route.

The next few sections show how to navigate between two routes, using these steps:

1. Create two routes.
2. Navigate to the second route using `Navigator.push()`.
3. Return to the first route using `Navigator.pop()`.

Navigate to a new screen and back

How to navigate between routes.

Most apps contain several screens for displaying different types of information. For example, an app might have a screen that displays products. When the user taps the image of a product, a new screen displays details about the product.

> **Terminology**
>
> In Flutter, *screens* and *pages* are called *routes*. The remainder of this recipe refers to routes.

In Android, a route is equivalent to an `Activity`. In iOS, a route is equivalent to a `ViewController`. In Flutter, a route is just a widget.

This recipe uses the `Navigator` to navigate to a new route.

The next few sections show how to navigate between two routes, using these steps:

1. Create two routes.
2. Navigate to the second route using `Navigator.push()`.
3. Return to the first route using `Navigator.pop()`.

1. Create two routes

First, create two routes to work with. Since this is a basic example, each route contains only a single button. Tapping the button on the first route navigates to the second route. Tapping the button on the second route returns to the first route.

First, set up the visual structure:

เพิ่มหน้าที่สองลงไป

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

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

2. Navigate to the second route using Navigator.push()

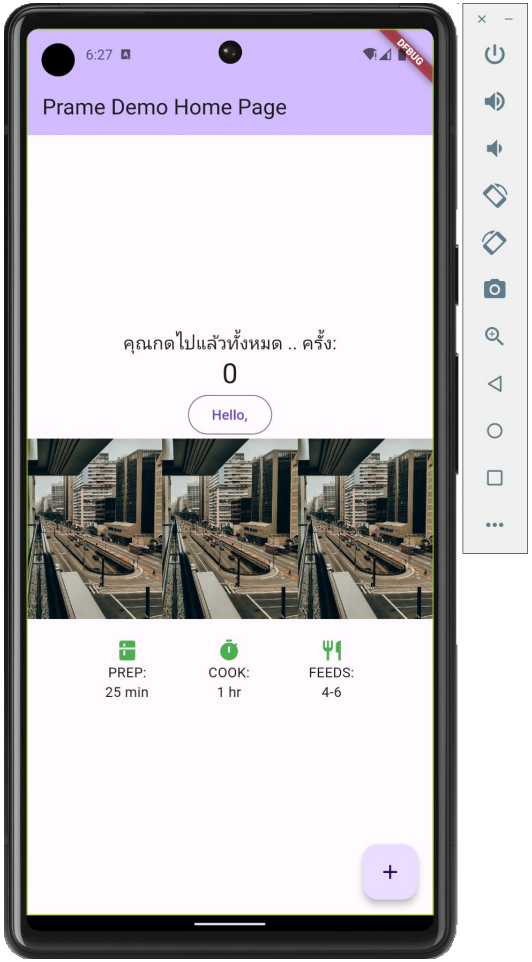
To switch to a new route, use the `Navigator.push()` method. The `push()` method adds a `Route` to the stack of routes managed by the `Navigator`. Where does the `Route` come from? You can create your own, or use a platform-specific route such as `MaterialPageRoute` or `CupertinoPageRoute`. A platform-specific route is useful because it transitions to the new route using a platform-specific animation.

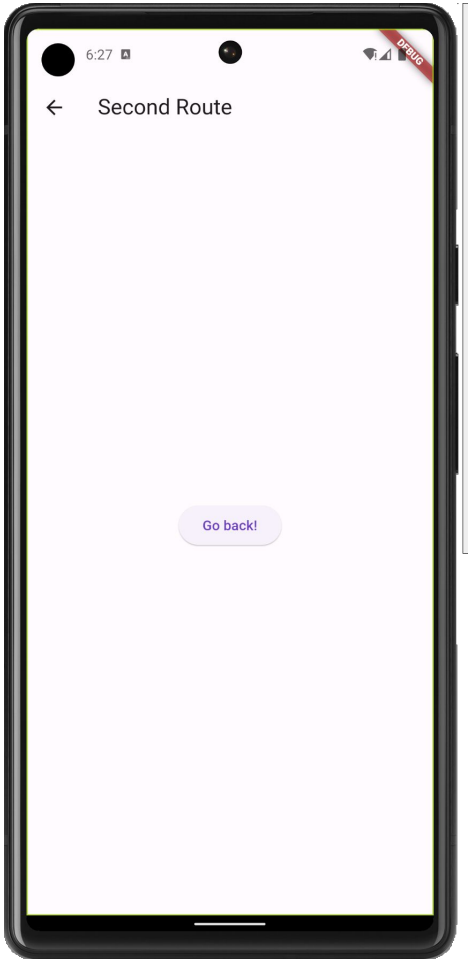
In the `build()` method of the `FirstRoute` widget, update the `onPressed()` callback:

สร้าง Navigator ไว้ใน OutlinedButton ที่เคยสร้างตอนแรก

```
106            mainAxisAlignment: MainAxisAlignment.center,
107            children: <Widget>[
108              const Text(
109                'คุณกดไปแล้วทั้งหมด .. ครั้ง:',
110                style: TextStyle(
111                  fontSize: 18,
112                ), // TextStyle
113              ), // Text
114              Text(
115                '$_counter',
116                style: Theme.of(context).textTheme.headlineMedium,
117              ), // Text
118              OutlinedButton(
119                  onPressed: () {
120                    Navigator.push(
121                      context,
122                      MaterialPageRoute(
123                          builder: (context) => const SecondRoute()), // MaterialPageRoute
124                    );
125                  },
126                  child: Text("Hello,")), // OutlinedButton
127              Row(
```

เพิ่มหน้าที่สองลงไป





3. Return to the first route using Navigator.pop()

How do you close the second route and return to the first? By using the `Navigator.pop()` method. The `pop()` method removes the current `Route` from the stack of routes managed by the `Navigator`.

To implement a return to the original route, update the `onPressed()` callback in the `SecondRoute` widget:

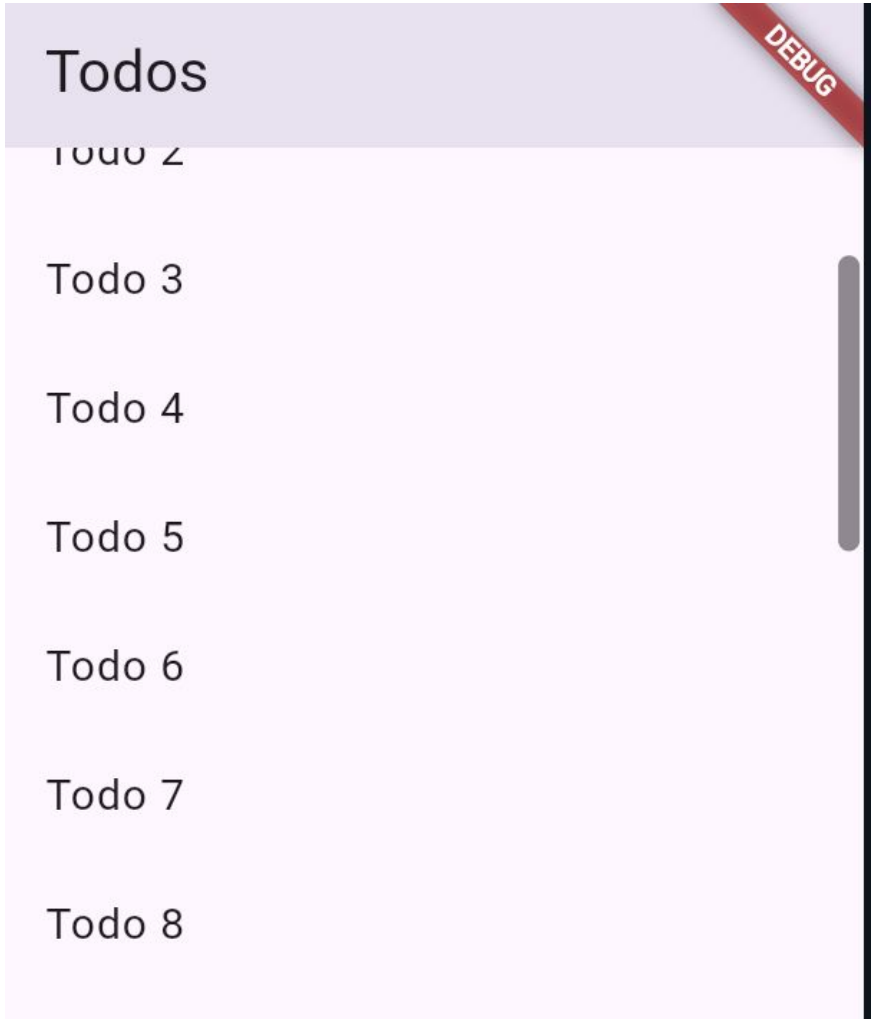
```
// Within the SecondRoute widget
onPressed: () {
  Navigator.pop(context);
}
```

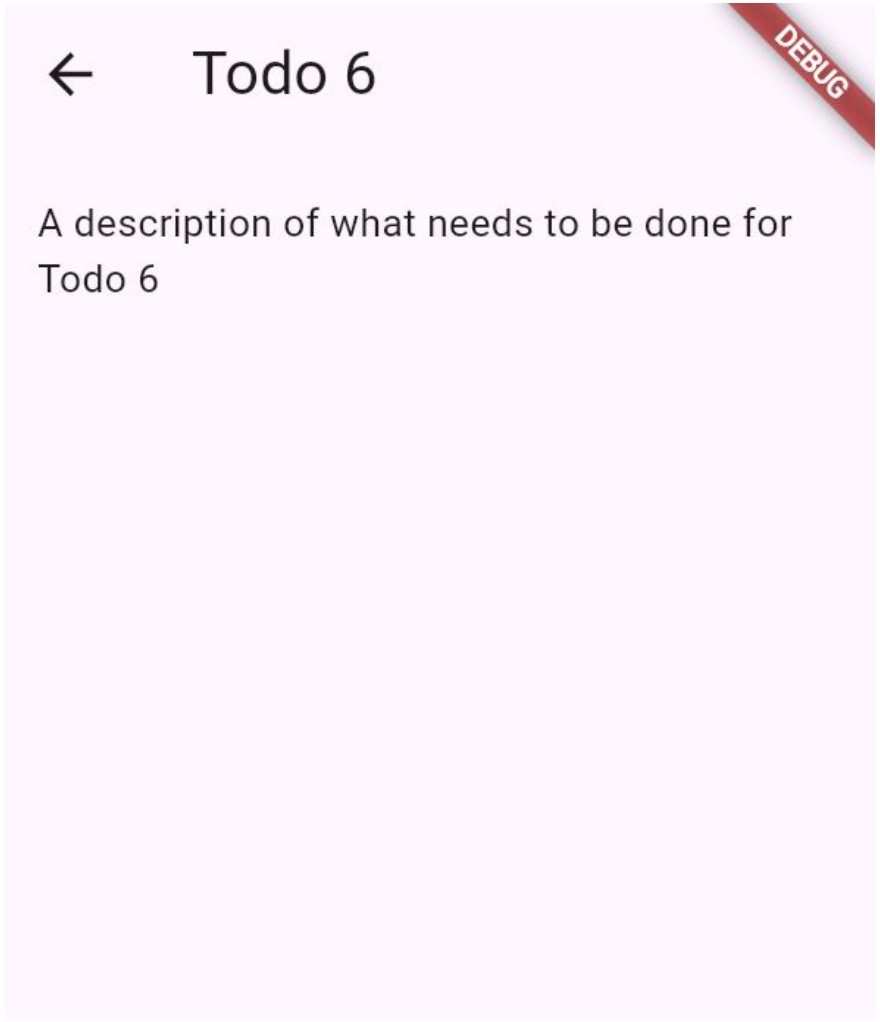
Additional navigation methods

The recipe in this topic shows you one way to navigate to a new screen and back to the previous scene, using the `push` and `pop` methods in the `Navigator` class, but there are several other `Navigator` static methods that you can use. Here are a few of them:

- `pushAndRemoveUntil`: Adds a navigation route to the stack and then removes the most recent routes from the stack until a condition is met.
- `pushReplacement`: Replaces the current route on the top of the stack with a new one.
- `replace`: Replace a route on the stack with another route.
- `replaceRouteBelow`: Replace the route below a specific route on the stack.
- `popUntil`: Removes the most recent routes that were added to the stack of navigation routes until a condition is met.
- `removeRoute`: Remove a specific route from the stack.
- `removeRouteBelow`: Remove the route below a specific route on the stack.
- `restorablePush`: Restore a route that was removed from the stack.

ส่งค่าข้ามหน้า ทำอย่างไร ?


Todos
DEBUG
Todo 2
Todo 3
Todo 4
Todo 5
Todo 6
Todo 7
Todo 8


Todo 6
DEBUG
A description of what needs to be done for Todo 6

Send data to a new screen

How to pass data to a new route.

Often, you not only want to navigate to a new screen, but also pass data to the screen as well. For example, you might want to pass information about the item that's been tapped.

Remember: Screens are just widgets. In this example, create a list of todos. When a todo is tapped, navigate to a new screen (widget) that displays information about the todo. This recipe uses the following steps:

1. Define a todo class.
2. Display a list of todos.
3. Create a detail screen that can display information about a todo.
4. Navigate and pass data to the detail screen.

1. Define a todo class

First, you need a simple way to represent todos. For this example, create a class that contains two pieces of data: the title and description.

```
class Todo {
  final String title;
  final String description;

  const Todo(this.title, this.description);
}
```

2. Create a list of todos

Second, display a list of todos. In this example, generate 20 todos and show them using a ListView. For more information on working with lists, see the Use lists recipe.

Generate the list of todos

```dart
final todos = List.generate(
  20,
  (i) => Todo(
    'Todo $i',
    'A description of what needs to be done for Todo $i',
  ),
);
```

Display the list of todos using a ListView

```dart
ListView.builder(
  itemCount: todos.length,
  itemBuilder: (context, index) {
    return ListTile(title: Text(todos[index].title));
  },
)
```

So far, so good. This generates 20 todos and displays them in a ListView.

3. Create a Todo screen to display the list

For this, we create a `StatelessWidget`. We call it `TodosScreen`. Since the contents of this page won't change during runtime, we'll have to require the list of todos within the scope of this widget.

We pass in our `ListView.builder` as body of the widget we're returning to `build()`. This'll render the list on to the screen for you to get going!

```dart
class TodosScreen extends StatelessWidget {
  // Requiring the list of todos.
  const TodosScreen({super.key, required this.todos});

  final List<Todo> todos;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Todos')),
      //passing in the ListView.builder
      body: ListView.builder(
        itemCount: todos.length,
        itemBuilder: (context, index) {
          return ListTile(title: Text(todos[index].title));
        },
      ),
    );
  }
}
```

With Flutter's default styling, you're good to go without sweating about things that you'd like to do later on!

4. Create a detail screen to display information about a todo

Now, create the second screen. The title of the screen contains the title of the todo, and the body of the screen shows the description.

Since the detail screen is a normal `StatelessWidget`, require the user to enter a `Todo` in the UI. Then, build the UI using the given todo.

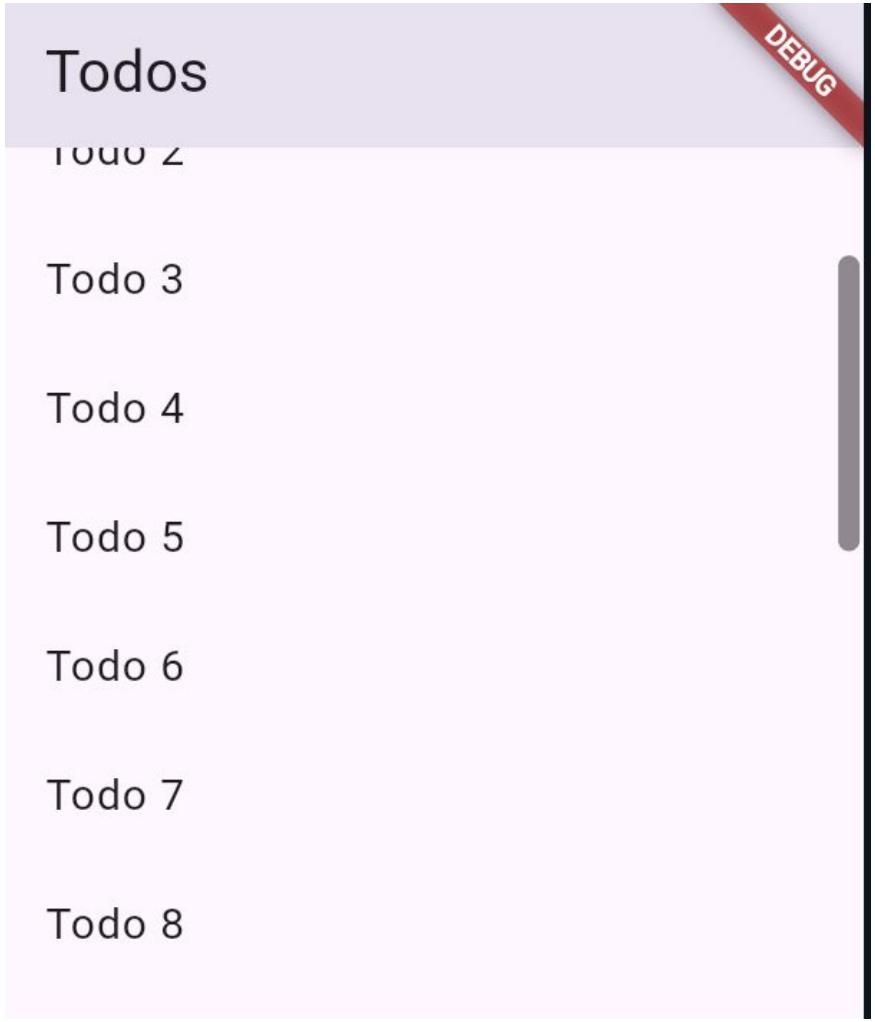
```dart
class DetailScreen extends StatelessWidget {
  // In the constructor, require a Todo.
  const DetailScreen({super.key, required this.todo});

  // Declare a field that holds the Todo.
  final Todo todo;

  @override
  Widget build(BuildContext context) {
    // Use the Todo to create the UI.
    return Scaffold(
      appBar: AppBar(title: Text(todo.title)),
      body: Padding(
        padding: const EdgeInsets.all(16),
        child: Text(todo.description),
      ),
    );
  }
}
```

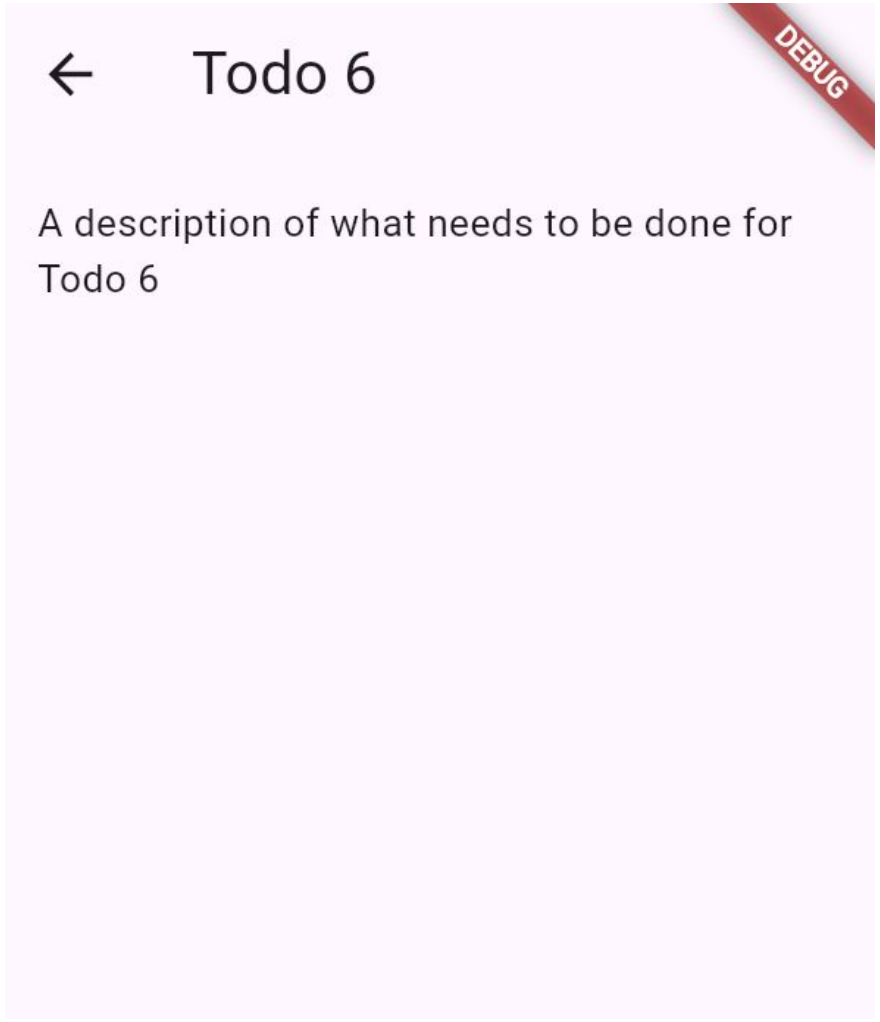
5. Navigate and pass data to the detail screen

With a `DetailScreen` in place, you're ready to perform the Navigation. In this example, navigate to the `DetailScreen` when a user taps a todo in the list. Pass the todo to the `DetailScreen`.

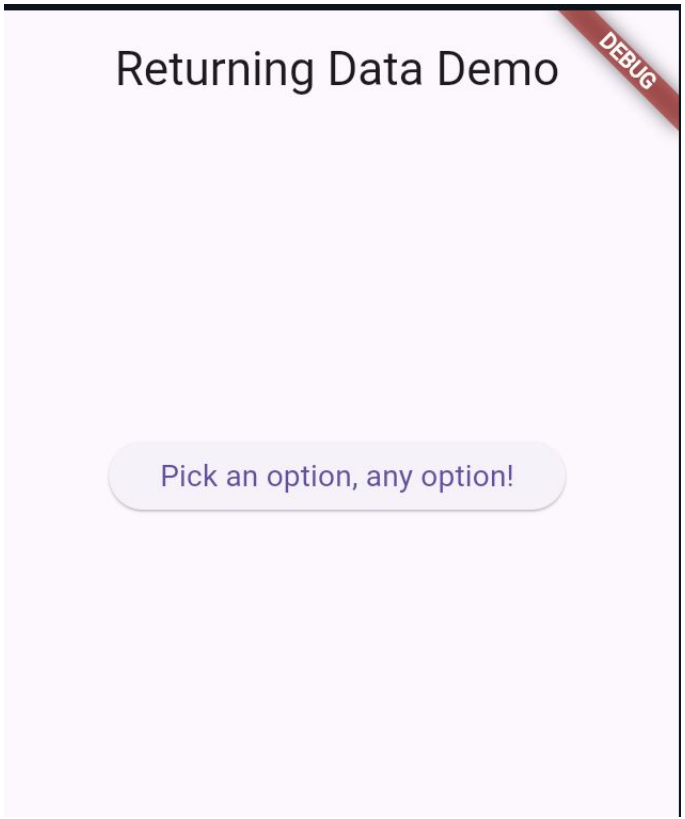
To capture the user's tap in the `TodosScreen`, write an `onTap()` callback for the `ListTile` widget. Within the `onTap()` callback, use the `Navigator.push()` method.

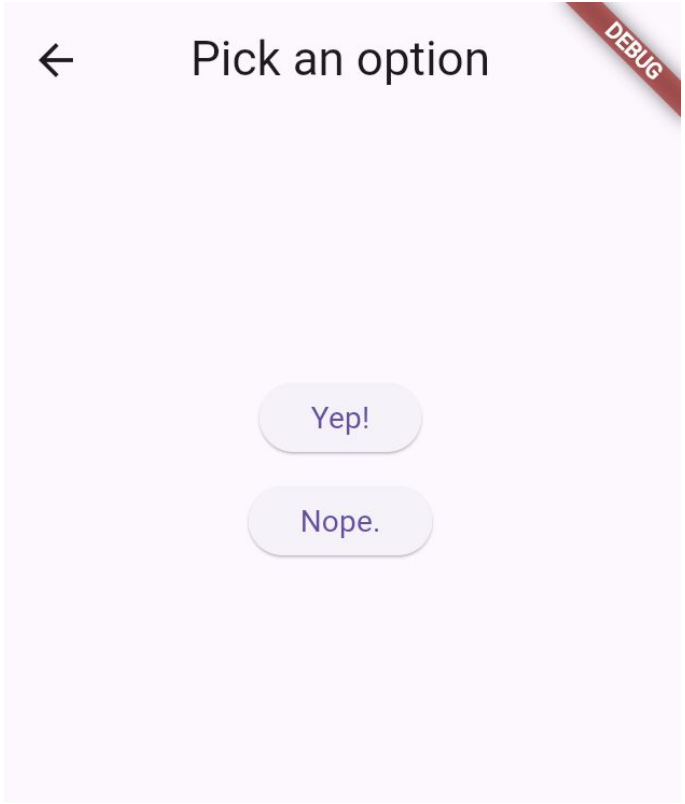
```
body: ListView.builder(
  itemCount: todos.length,
  itemBuilder: (context, index) {
    return ListTile(
      title: Text(todos[index].title),
      // When a user taps the ListTile, navigate to the DetailScreen.
      // Notice that you're not only creating a DetailScreen, you're
      // also passing the current todo through to it.
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute<void>(
            builder: (context) => DetailScreen(todo: todos[index]),
          ),
        );
      },
    );
  },
),
```

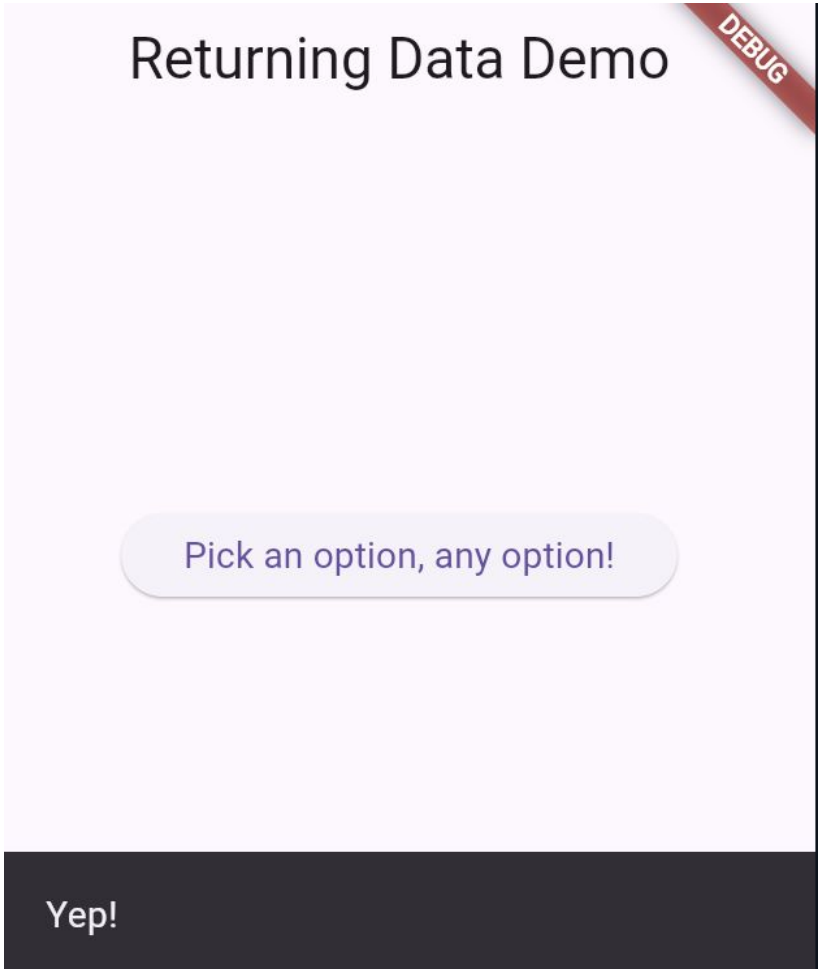

Todos
DEBUG
Todo 2
Todo 3
Todo 4
Todo 5
Todo 6
Todo 7
Todo 8


Todo 6
DEBUG
A description of what needs to be done for Todo 6

ส่งค่ากลับจากหน้าอื่น


Returning Data Demo
DEBUG
Pick an option, any option!


Pick an option
DEBUG
Yep!
Nope.


Returning Data Demo
DEBUG
Pick an option, any option!
Yep!

Return data from a screen

How to return data from a new screen.

In some cases, you might want to return data from a new screen. For example, say you push a new screen that presents two options to a user. When the user taps an option, you want to inform the first screen of the user's selection so that it can act on that information.

You can do this with the `Navigator.pop()` method using the following steps:

1. Define the home screen
2. Add a button that launches the selection screen
3. Show the selection screen with two buttons
4. When a button is tapped, close the selection screen
5. Show a snackbar on the home screen with the selection

1. Define the home screen

The home screen displays a button. When tapped, it launches the selection screen.

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Returning Data Demo')),
      // Create the SelectionButton widget in the next step.
      body: const Center(child: SelectionButton()),
    );
  }
}
```

2. Add a button that launches the selection screen

Now, create the SelectionButton, which does the following:

- Launches the SelectionScreen when it's tapped.
- Waits for the SelectionScreen to return a result.

```dart
class SelectionButton extends StatefulWidget {
  const SelectionButton({super.key});

  @override
  State<SelectionButton> createState() => _SelectionButtonState();
}

class _SelectionButtonState extends State<SelectionButton> {
  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: () {
        _navigateAndDisplaySelection(context);
      },
      child: const Text('Pick an option, any option!'),
    );
  }

  Future<void> _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<String>(builder: (context) => const SelectionScreen()),
    );
  }
}
```

3. Show the selection screen with two buttons

Now, build a selection screen that contains two buttons. When a user taps a button, that app closes the selection screen and lets the home screen know which button was tapped.

This step defines the UI. The next step adds code to return data.

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Pick an option')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Padding(
              padding: const EdgeInsets.all(8),
              child: ElevatedButton(
                onPressed: () {
                  // Pop here with "Yep"...
                },
                child: const Text('Yep!'),
              ),
            ),
            Padding(
              padding: const EdgeInsets.all(8),
              child: ElevatedButton(
                onPressed: () {
                  // Pop here with "Nope"...
                },
                child: const Text('Nope.'),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

4. When a button is tapped, close the selection screen

Now, update the `onPressed()` callback for both of the buttons. To return data to the first screen, use the `Navigator.pop()` method, which accepts an optional second argument called `result`. Any result is returned to the `Future` in the SelectionButton.

Yep button

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

Nope button

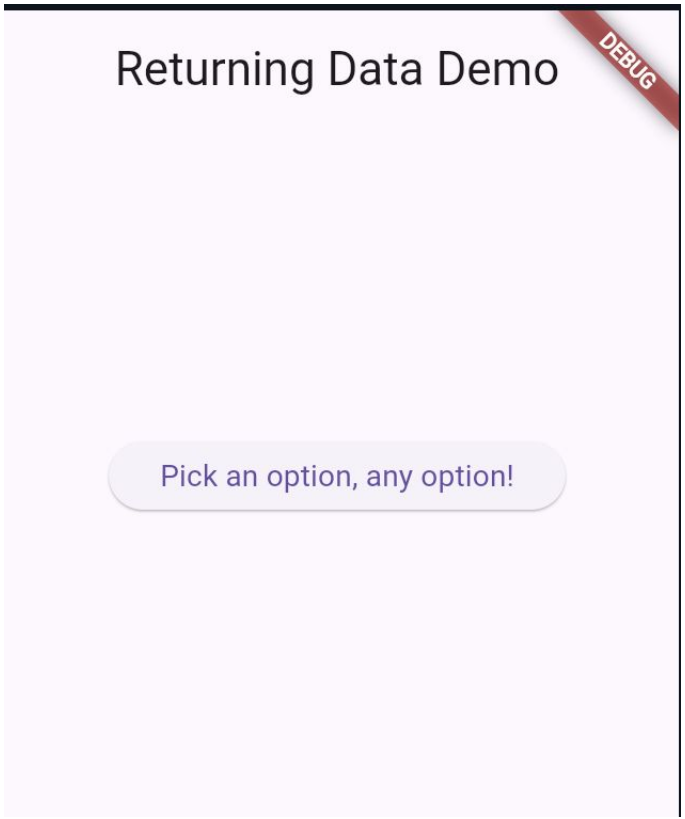
```dart
ElevatedButton(
  onPressed: () {
    // Close the screen and return "Nope." as the result.
    Navigator.pop(context, 'Nope.');
  },
  child: const Text('Nope.'),
)
```

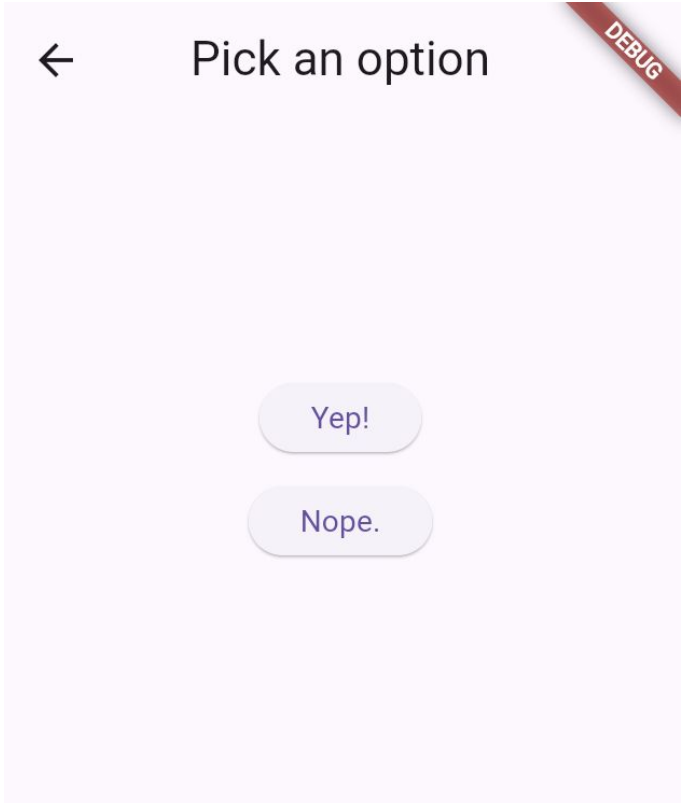
5. Show a snackbar on the home screen with the selection

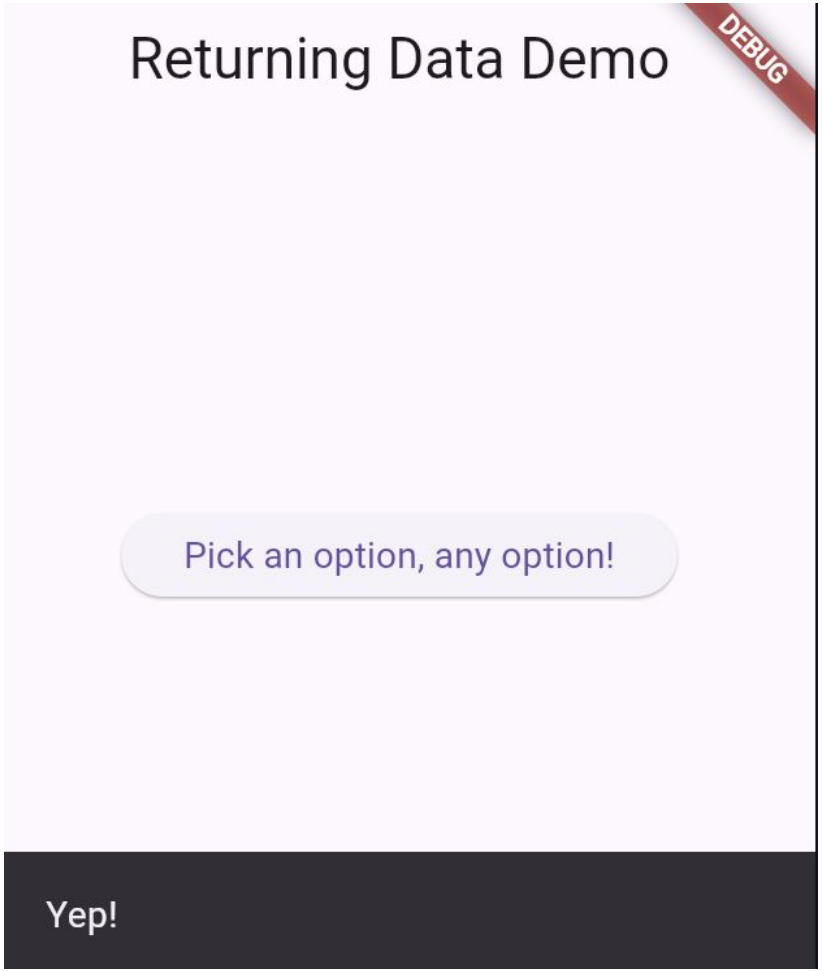
Now that you're launching a selection screen and awaiting the result, you'll want to do something with the information that's returned.

In this case, show a snackbar displaying the result by using the `_navigateAndDisplaySelection()` method in `SelectionButton`:

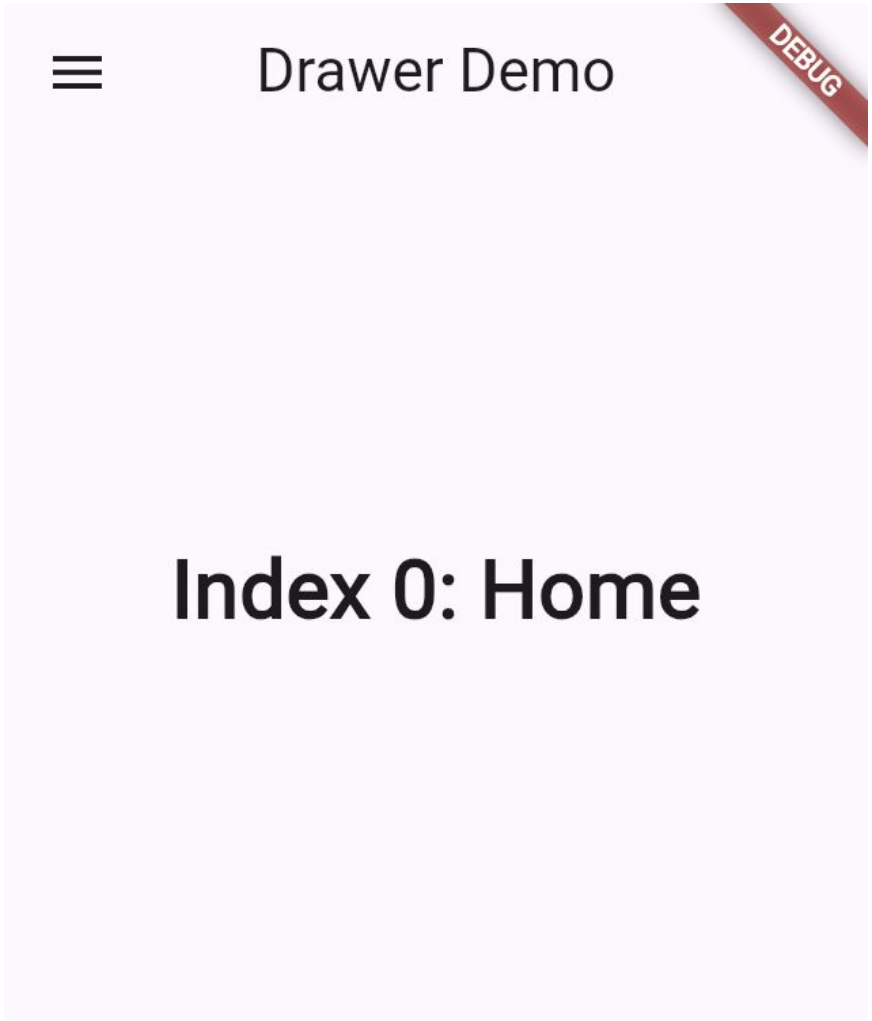
```
// A method that launches the SelectionScreen and awaits the result from
// Navigator.pop.
Future<void> _navigateAndDisplaySelection(BuildContext context) async {
  // Navigator.push returns a Future that completes after calling
  // Navigator.pop on the Selection Screen.
  final result = await Navigator.push(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

  // When a BuildContext is used from a StatefulWidget, the mounted property
  // must be checked after an asynchronous gap.
  if (!context.mounted) return;

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

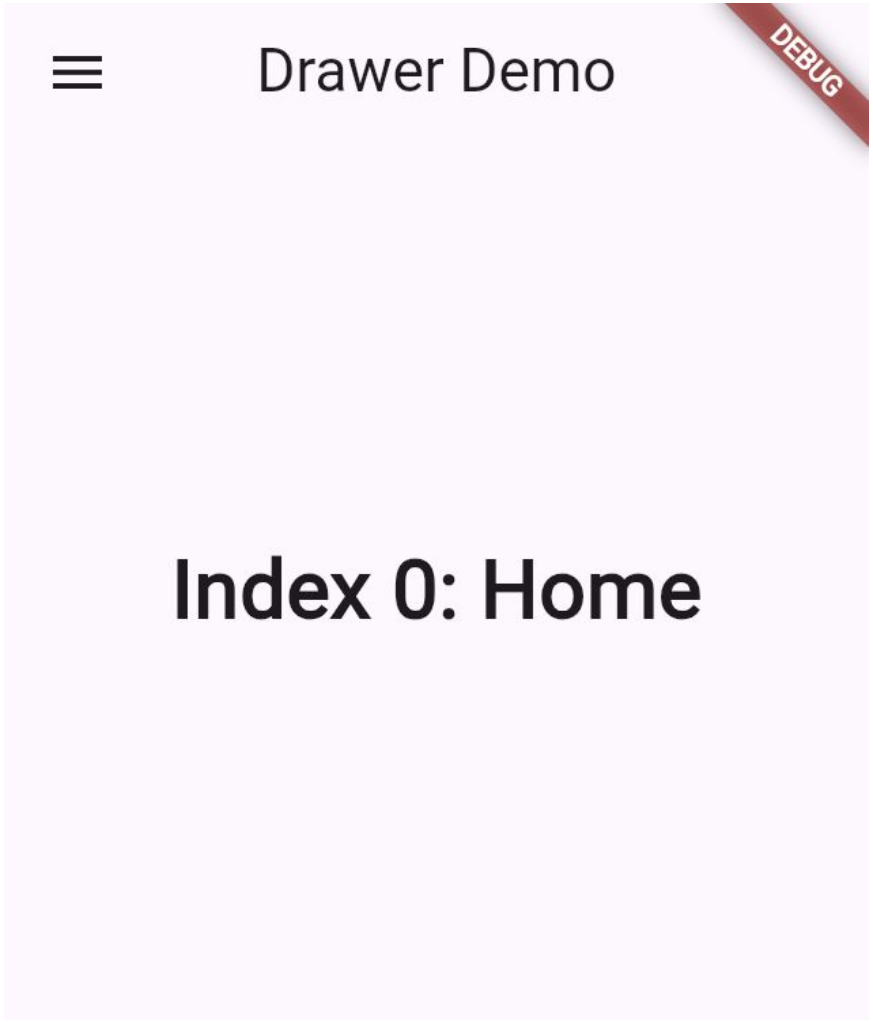

Returning Data Demo
DEBUG
Pick an option, any option!

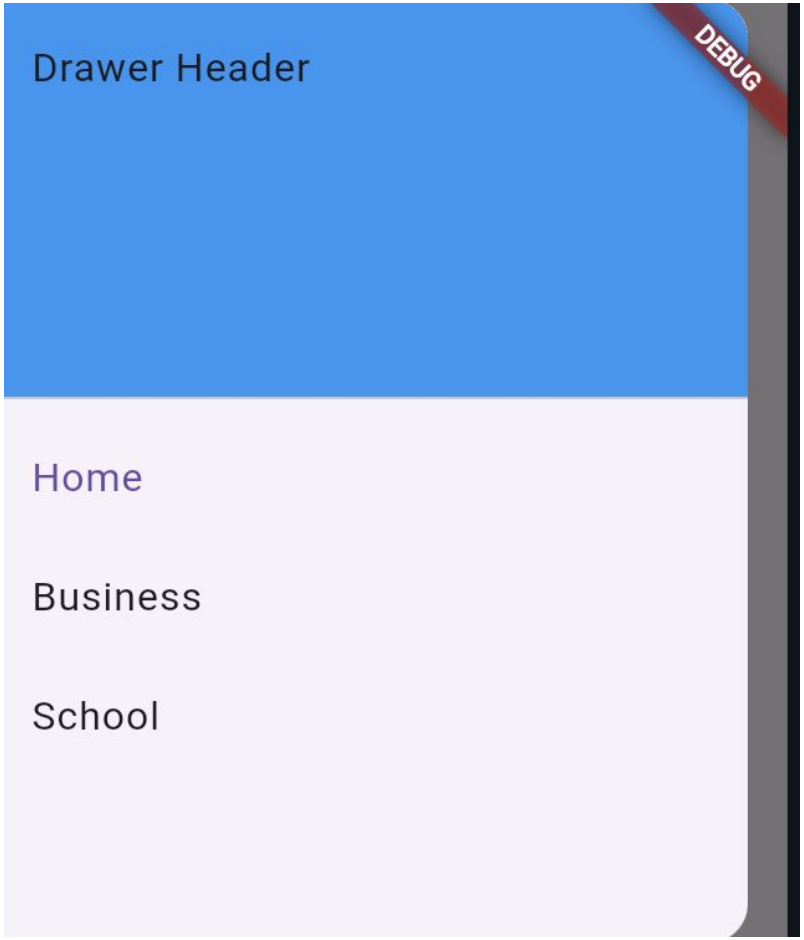

Pick an option
DEBUG
Yep!
Nope.

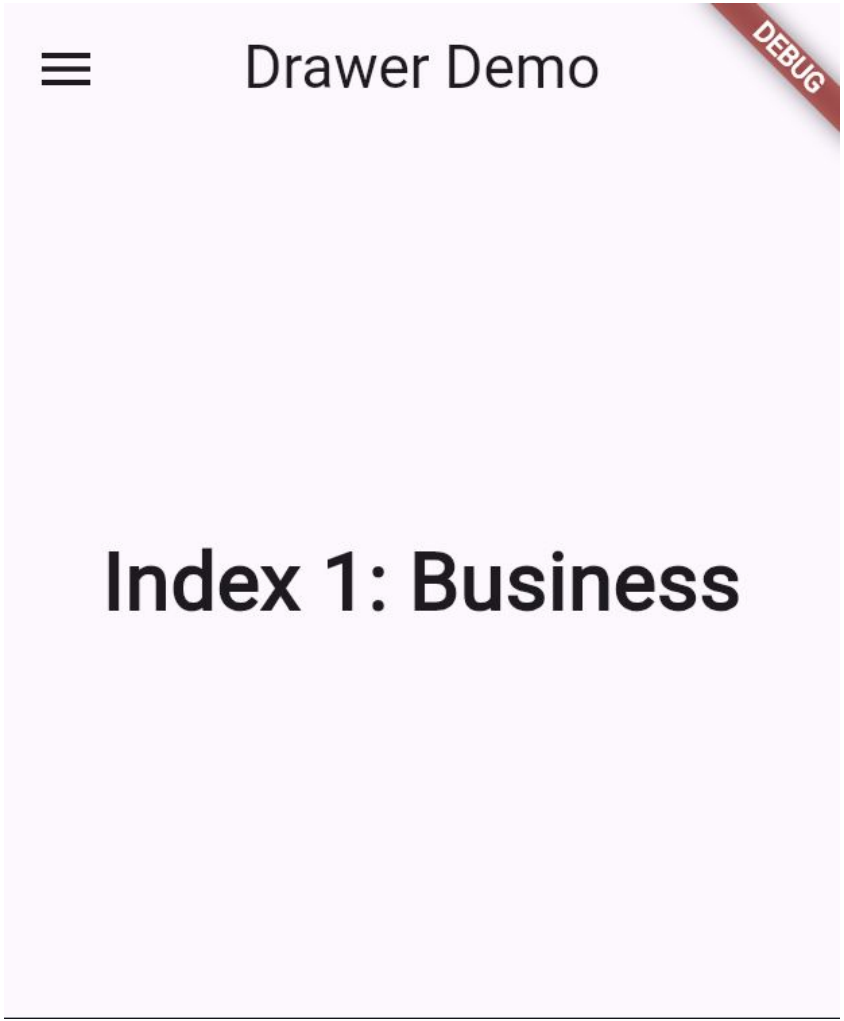

Returning Data Demo
DEBUG
Pick an option, any option!
Yep!

Add a drawer to a screen


Drawer Demo
DEBUG
Index 0: Home


Drawer Demo
DEBUG
Index 0: Home


Drawer Header
DEBUG
Home
Business
School


Drawer Demo
DEBUG
Index 1: Business

Add a drawer to a screen

How to implement a Material Drawer.

In apps that use Material Design, there are two primary options for navigation: tabs and drawers. When there is insufficient space to support tabs, drawers provide a handy alternative.

In Flutter, use the `Drawer` widget in combination with a `Scaffold` to create a layout with a Material Design drawer. This recipe uses the following steps:

1. Create a `Scaffold`.
2. Add a drawer.
3. Populate the drawer with items.
4. Close the drawer programmatically.

1. Create a `Scaffold`

To add a drawer to the app, wrap it in a `Scaffold` widget. The `Scaffold` widget provides a consistent visual structure to apps that follow the Material Design Guidelines. It also supports special Material Design components, such as Drawers, AppBars, and SnackBars.

In this example, create a `Scaffold` with a `drawer`:

```dart
Scaffold(
  appBar: AppBar(title: const Text('AppBar without hamburger button')),
  drawer: // Add a Drawer here in the next step.
);
```

2. Add a drawer

Now add a drawer to the `Scaffold`. A drawer can be any widget, but it's often best to use the `Drawer` widget from the material library, which adheres to the Material Design spec.

```dart
Scaffold(
  appBar: AppBar(title: const Text('AppBar with hamburger button')),
  drawer: Drawer(
    child: // Populate the Drawer in the next step.
  ),
);
```

BORNTO DEV

3. Populate the drawer with items

Now that you have a `Drawer` in place, add content to it. For this example, use a `ListView`. While you could use a `Column` widget, `ListView` is handy because it allows users to scroll through the drawer if the content takes more space than the screen supports.

Populate the `ListView` with a `DrawerHeader` and two `ListTile` widgets. For more information on working with Lists, see the list recipes.

```
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(
    // Important: Remove any padding from the ListView.
    padding: EdgeInsets.zero,
    children: [
      const DrawerHeader(
        decoration: BoxDecoration(color: Colors.blue),
        child: Text('Drawer Header'),
      ),
      ListTile(
        title: const Text('Item 1'),
        onTap: () {
          // Update the state of the app.
          // ...
        },
      ),
      ListTile(
        title: const Text('Item 2'),
        onTap: () {
          // Update the state of the app.
          // ...
        },
      ),
    ],
  ),
);
```

4. Open the drawer programmatically

Typically, you don't need to write any code to open a `drawer`, Because when the `leading` widget is null, the default implementation in `AppBar` is `DrawerButton`.

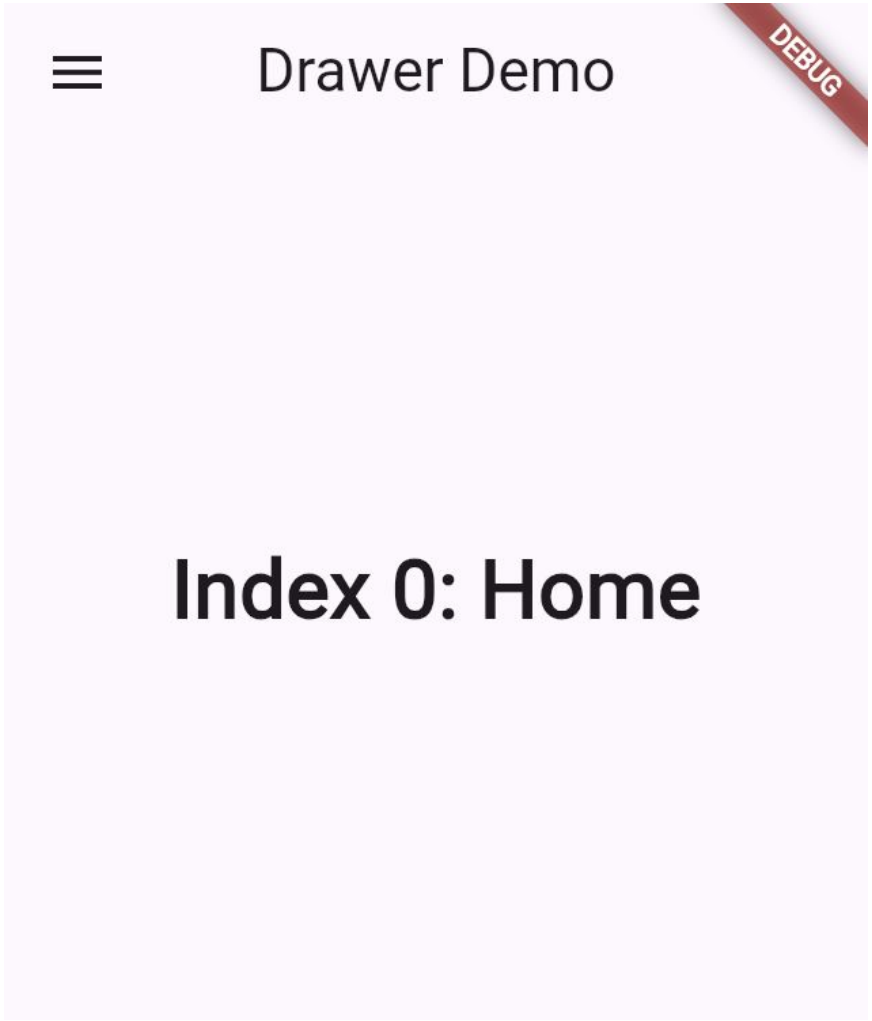
But if you want to have free control of the `drawer`. You can do this by using the `Builder` call `Scaffold.of(context).openDrawer()`.

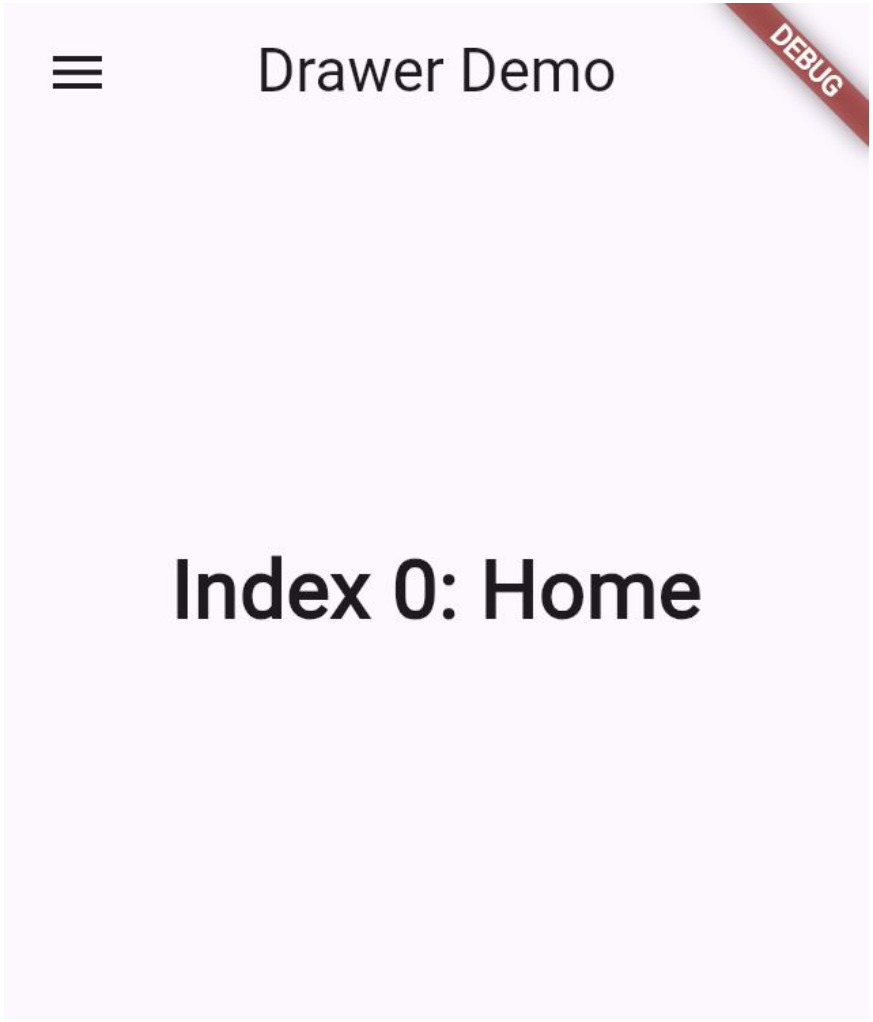
5. Close the drawer programmatically

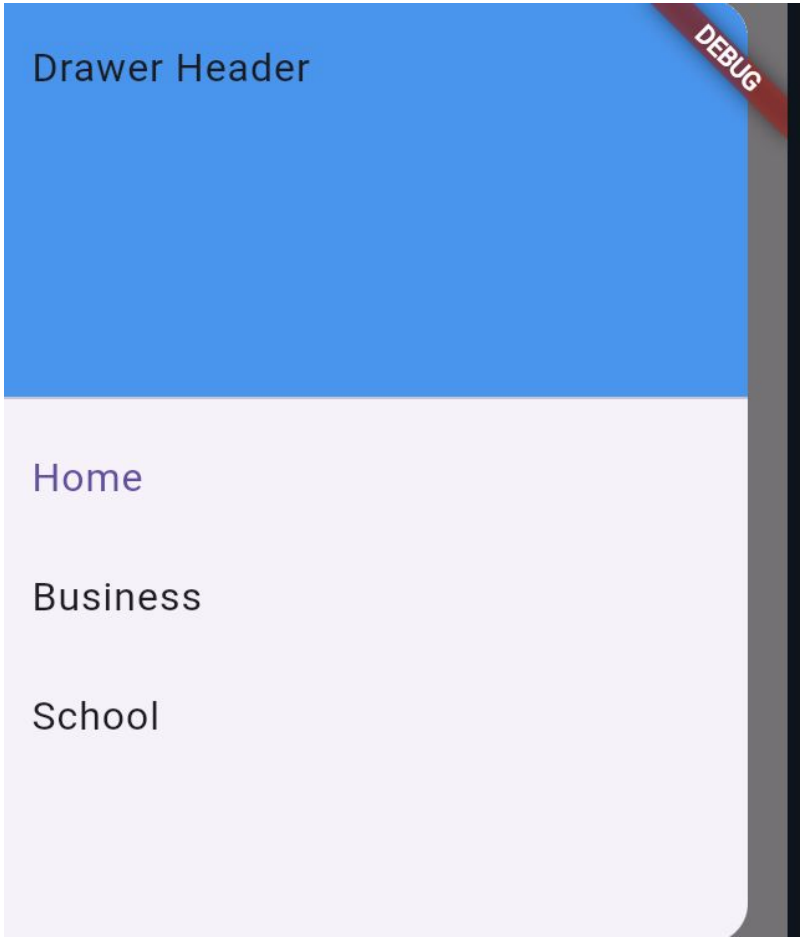
After a user taps an item, you might want to close the drawer. You can do this by using the `Navigator`.

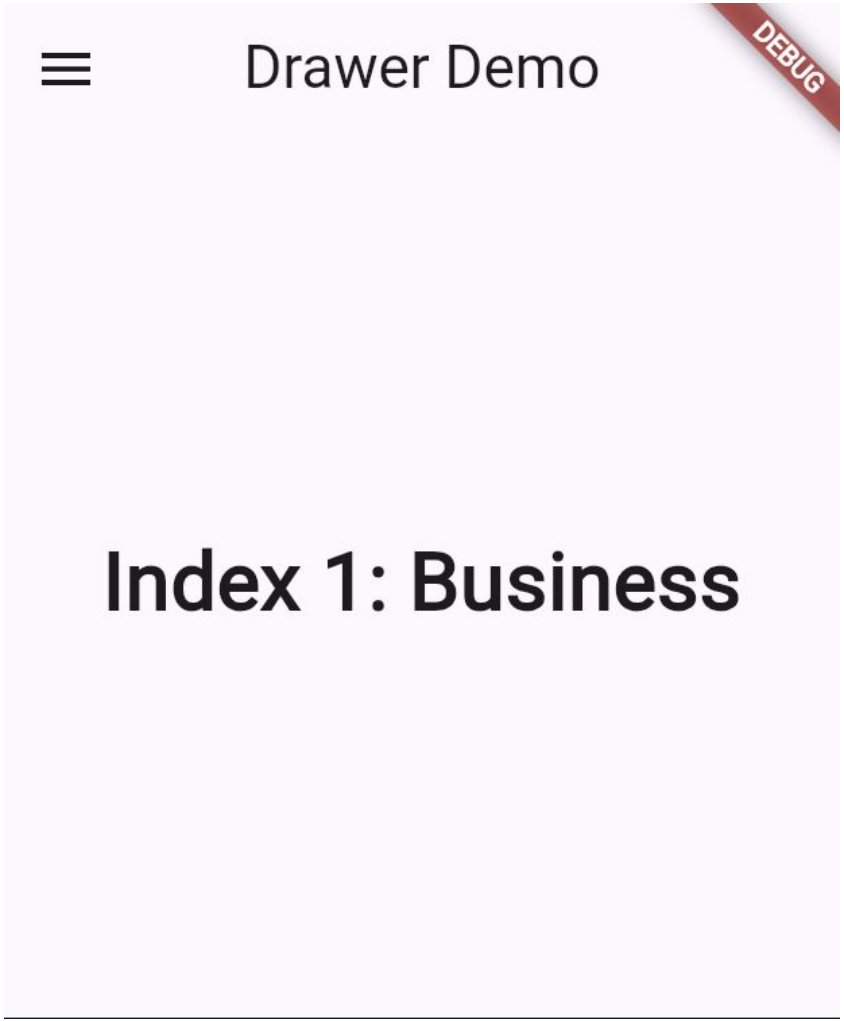
When a user opens the drawer, Flutter adds the drawer to the navigation stack. Therefore, to close the drawer, call `Navigator.pop(context)`.

```
ListTile(
  title: const Text('Item 1'),
  onTap: () {
    // Update the state of the app
    // ...
    // Then close the drawer
    Navigator.pop(context);
  },
),
```


Drawer Demo
DEBUG
Index 0: Home


Drawer Demo
DEBUG
Index 0: Home


Drawer Header
DEBUG
Home
Business
School


Drawer Demo
DEBUG
Index 1: Business

การบ้านสำหรับเช็คชื่อรอบถัดไป ..

- ✅ สร้างแอปพลิเคชันที่มีการ Navigate
- ✅ มีหน้าอย่างน้อย 3 หน้า
- ✅ มีการส่งข้อมูลข้ามหน้ากัน
- ✅ เป็นแอปพลิเคชันที่มีใช้ได้จริง

PROJECT รวมกลุ่ม 3 คน

Technical 25% Idea 15%