

# MOBILE PROGRAMMING WITH FLUTTER - DAY3

เริ่มพัฒนาแอปพลิเคชันบนมือถือด้วย Flutter

## เกณฑ์คะแนน

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

# App Structure & Refactoring

The screenshot shows the VS Code interface with a Flutter project open. The Explorer sidebar on the left displays the project structure, including `lib` (which contains `main.dart`, `linux`, `macos`, `test`, `web`, `windows`, and several configuration files like `.gitignore`, `.metadata`, and `analysis\_options.yaml`). The main editor area shows the `main.dart` file with the following code:

```
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(const MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   const MyApp({super.key});
9
10 // This widget is the root of your application.
11 @override
12 Widget build(BuildContext context) {
13   return MaterialApp(
14     title: 'Flutter Demo',
15     theme: ThemeData(
16       // This is the theme of your application.
17       //
18       // TRY THIS: Try running your application with "flutter run". You'll see
19       // the application has a purple toolbar. Then, without quitting the app,
20       // try changing the seedColor in the colorScheme below to Colors.green
21   );
22 }
```

สมมติว่าเรากำลังจะทำแอปแนะนำสถานที่เที่ยว ใน Week ที่แล้ว

flutter\_application\_1

EXPLORER

main.dart X

lib > main.dart > MyApp

```
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(const MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   const MyApp({super.key});
9
10 // This widget is the root of your application.
11 @override
12 Widget build(BuildContext context) {
13   return MaterialApp(
14     title: 'Flutter Demo',
15     theme: ThemeData(
16       // This is the theme of your application.
17       //
18       // TRY THIS: Try running your application with "flutter run". You'll see
19       // the application has a purple toolbar. Then, without quitting the app,
20       // try changing the seedColor in the colorScheme below to Colors.green
21   );
22 }
```

PROBLEMS    OUTPUT    **DEBUG CONSOLE**    TERMINAL    ...

Filter (e.g. text, | exclude, | e...)

# สร้าง Project ใหม่ และ ลอง Run ใหม่

main.dart X

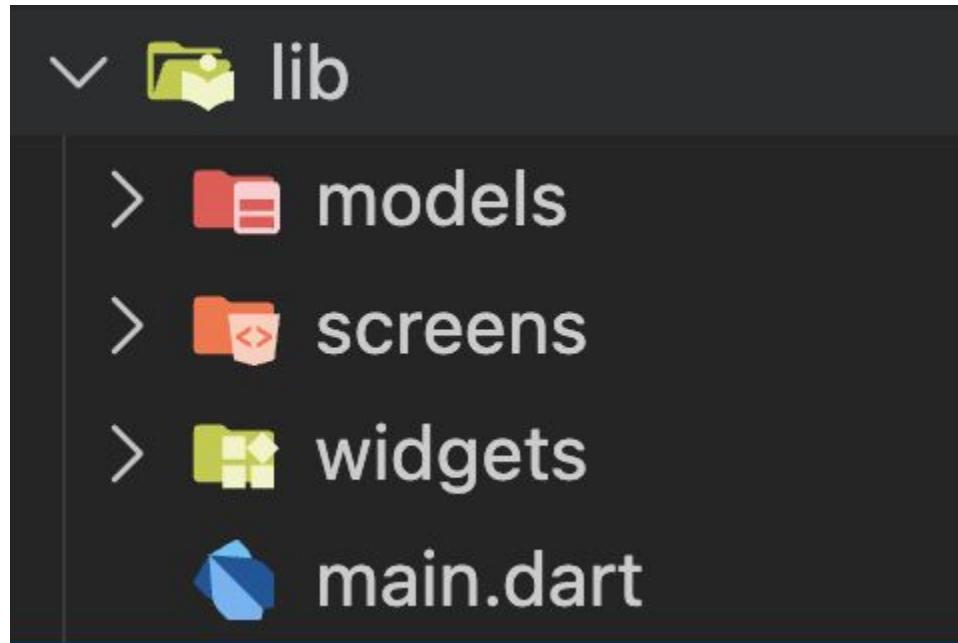
```
lib > main.dart > _MyHomePageState > build
56   class _MyHomePageState extends State<MyHomePage> {
71     Widget build(BuildContext context) {
105       mainAxisAlignment: MainAxisAlignment.center,
106       children: [
107         const Text('You have pushed the button this many times:'),
108         Text(
109           '_counter',
110           style: Theme.of(context).textTheme.headlineMedium,
111         ), Container( // Text
112           padding: EdgeInsets.all(20),
113           child: Text('Bangkok', style: TextStyle(fontSize: 24)),
114         ), // Container
115         Container(
116           padding: EdgeInsets.all(20),
117           child: Text('Chiang Mai', style: TextStyle(fontSize: 24)),
118         ), // Container
119       ],
120     ), // Column
121   ), // Center
122   floatingActionButton: FloatingActionButton(
123
```

# สร้าง Container ที่มี text

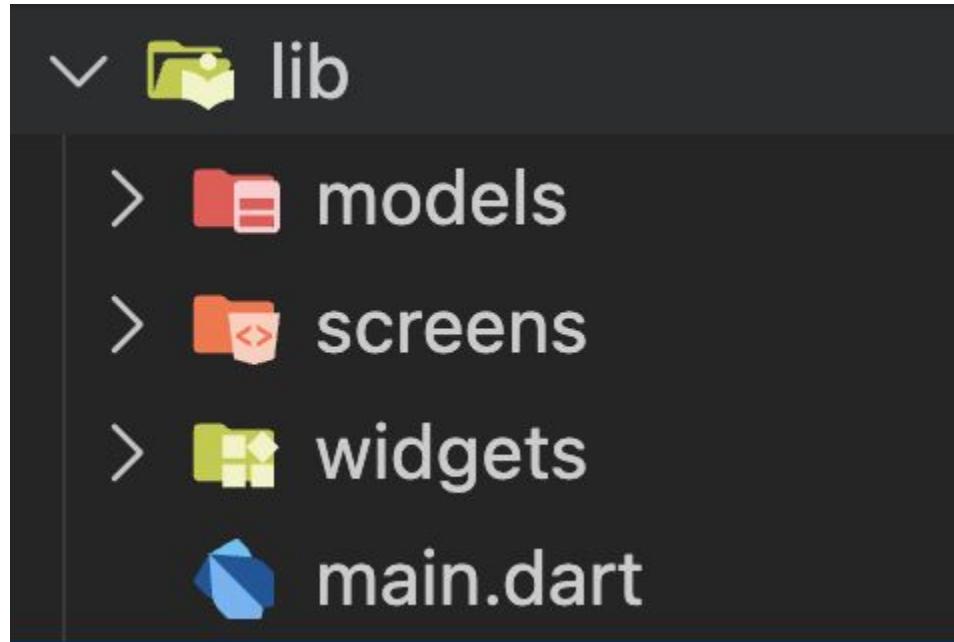
```
main.dart X

lib > main.dart > _MyHomePageState > build
56   class _MyHomePageState extends State<MyHomePage> {
71     Widget build(BuildContext context) {
105       mainAxisAlignment: MainAxisAlignment.center,
106       children: [
107         const Text('You have pushed the button this many times:'),
108         Text(
109           '_counter',
110           style: Theme.of(context).textTheme.headlineMedium,
111         ), Container( // Text
112           padding: EdgeInsets.all(20),
113           child: Text('Bangkok', style: TextStyle(fontSize: 24)),
114         ), // Container
115         Container(
116           padding: EdgeInsets.all(20),
117           child: Text('Chiang Mai', style: TextStyle(fontSize: 24)),
118         ), // Container
119       ],
120     ), // Column
121   ), // Center
122   floatingActionButton: FloatingActionButton(
123
```

เป็นการ Hardcode ลงไว้ในโปรแกรม



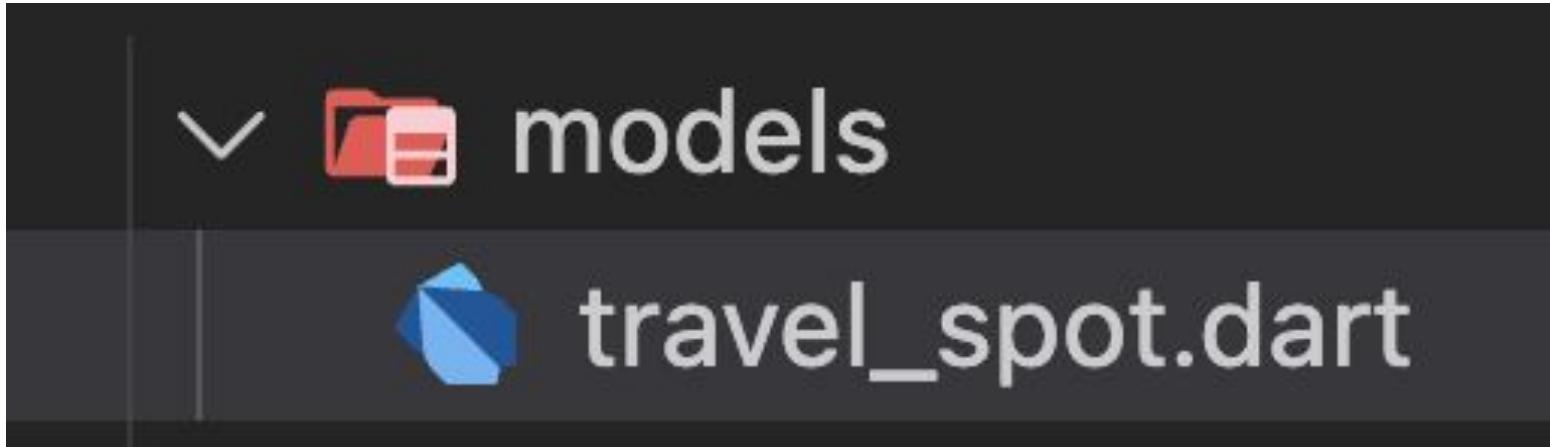
สร้าง Folder ทั้งหมด 3 ตัว



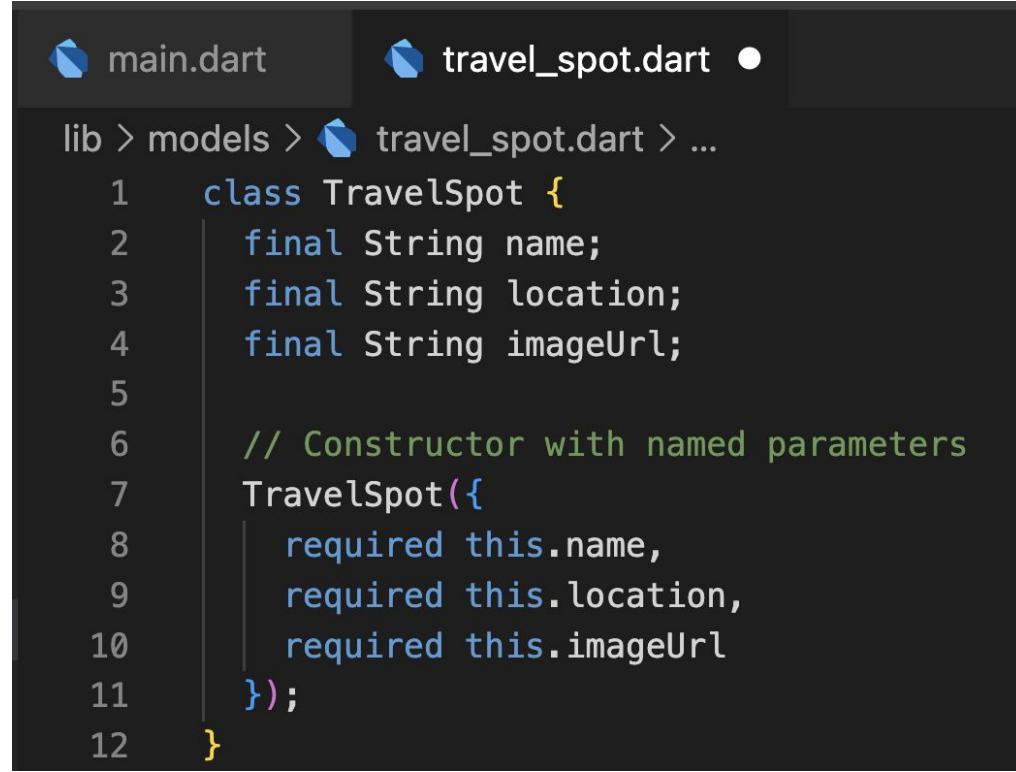
**models/** (เก็บพิมพ์เขียวของข้อมูล - ฝึกเขียน *Dart Class*)

**screens/** (เก็บหน้าจอหลักทั้งหมด - *Scaffolds*)

**widgets/** (เก็บชิ้นส่วน UI ที่ใช้ซ้ำ - *Reusable Components*)



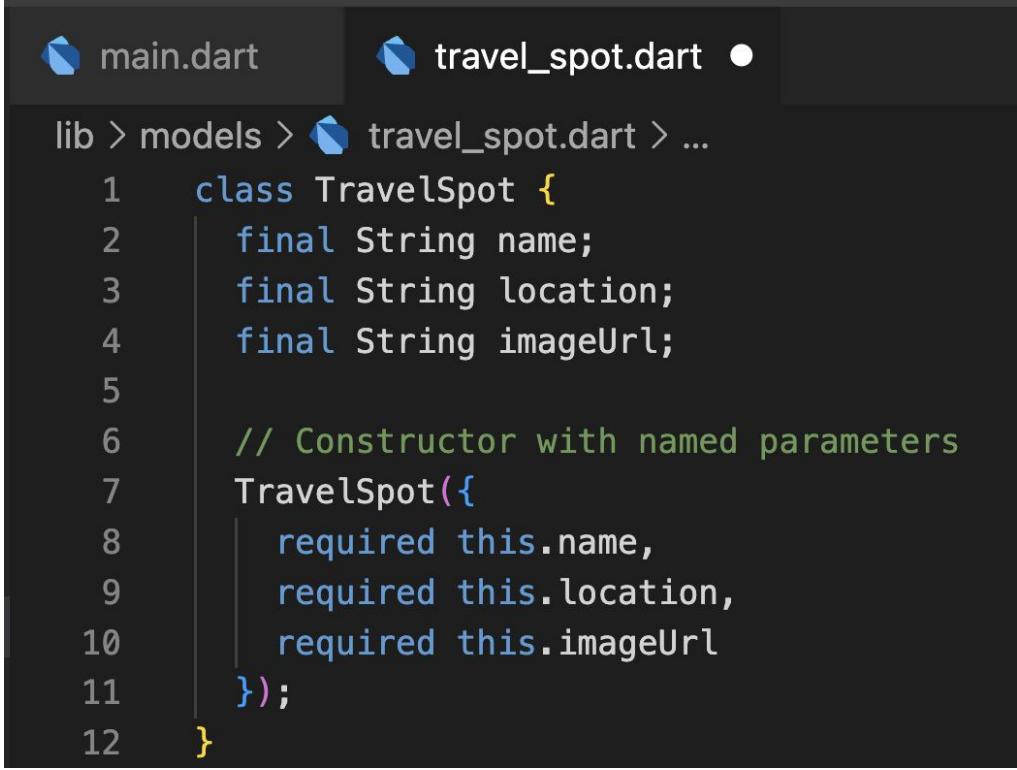
สร้างไฟล์ lib/models/travel\_spot.dart



The screenshot shows a code editor with two tabs at the top: 'main.dart' and 'travel\_spot.dart'. The 'travel\_spot.dart' tab is active, indicated by a yellow dot. The code in the editor is as follows:

```
lib > models > travel_spot.dart > ...
1   class TravelSpot {
2     final String name;
3     final String location;
4     final String imageUrl;
5
6     // Constructor with named parameters
7     TravelSpot({
8       required this.name,
9       required this.location,
10      required this.imageUrl
11    });
12 }
```

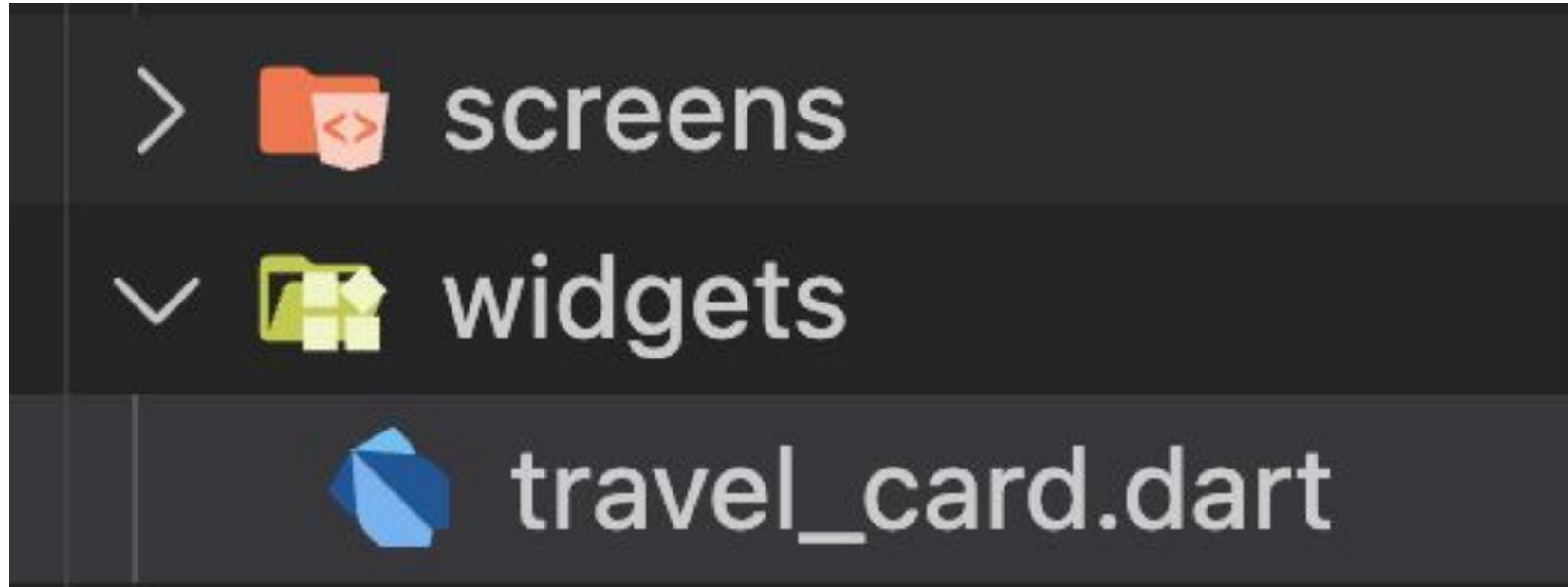
# สร้างไฟล์ lib/models/travel\_spot.dart



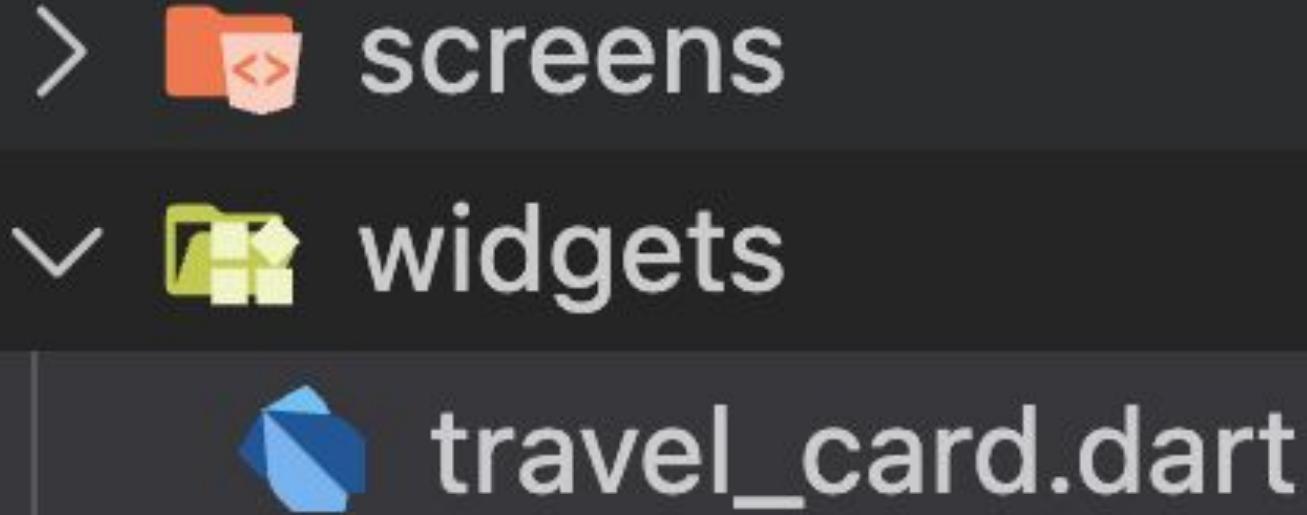
The screenshot shows a code editor with two tabs at the top: 'main.dart' and 'travel\_spot.dart'. The 'travel\_spot.dart' tab is active, indicated by a yellow dot next to its name. The code in the editor is as follows:

```
lib > models > travel_spot.dart > ...
1   class TravelSpot {
2     final String name;
3     final String location;
4     final String imageUrl;
5
6     // Constructor with named parameters
7     TravelSpot({
8       required this.name,
9       required this.location,
10      required this.imageUrl
11    });
12 }
```

**required** คือการบังคับว่าต้องส่งค่ามา ห้ามเป็น **null**



สร้างไฟล์ lib/widgets/travel\_card.dart



**Quiz : สร้างเพื่ออะไร ?**

```
lib > widgets > travel_card.dart > TravelCard
1   import 'package:flutter/material.dart';
2   import '../models/travel_spot.dart'; // อายาลีม Import Model ที่เราเพิ่งสร้าง
3
4   class TravelCard extends StatelessWidget {
5       // รับ Data Model เข้ามา แทนที่จะรับ String ที่ลากด้วย
6       final TravelSpot spot;
7
8   const TravelCard({super.key, required this.spot});
9
10  @override
11  Widget build(BuildContext context) {
12      return Card(
13          margin: EdgeInsets.all(10),
14          child: Column(
15              children: [
16                  // แสดงรูปภาพ (สมมติใช้ NetworkImage)
17                  Image.network(spot.imageUrl, height: 150, fit: BoxFit.cover),
18                  ListTile(
19                      title: Text(spot.name, style: TextStyle(fontWeight: FontWeight.bold)),
20                      subtitle: Text(spot.location),
21                      trailing: Icon(Icons.arrow_forward),
22                  ), // ListTile
23                  ],
24              ), // Column
25      ); // Card
26  }
27 }
```

travel\_spot.dart

travel\_card.dart X

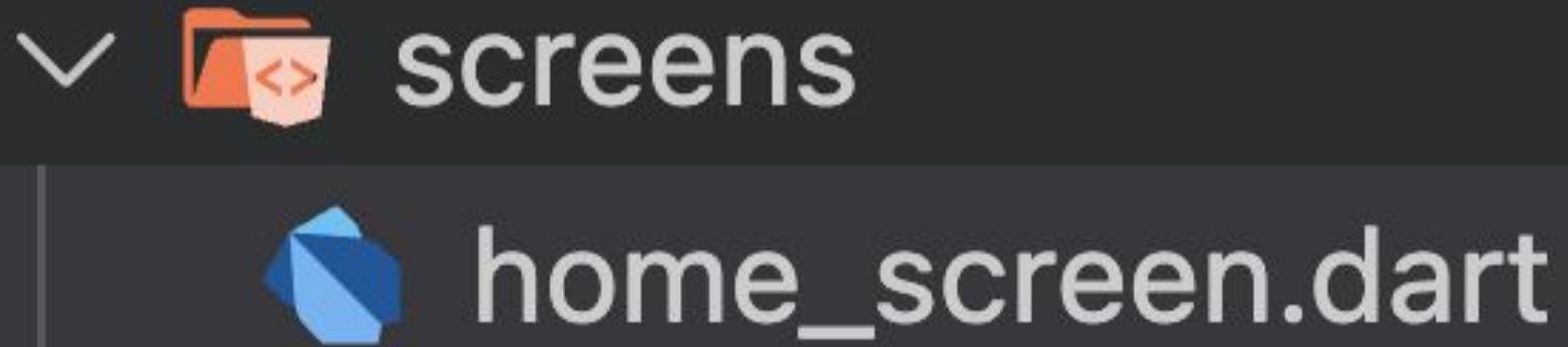
home\_screen.dart

main.dart

lib &gt; widgets &gt; travel\_card.dart &gt; TravelCard

```
1 import 'package:flutter/material.dart';
2 import '../models/travel_spot.dart'; // อย่าลืม Import Model ที่เราเพิ่งสร้าง
3
4 class TravelCard extends StatelessWidget {
5     // รับ Data Model เข้ามา แทนที่จะรับ String ทีละตัว
6     final TravelSpot spot;
7
8     const TravelCard({super.key, required this.spot});
9 }
```

```
8 const TravelCard({super.key, required this.spot});  
9  
10 @override  
11 Widget build(BuildContext context) {  
12     return Card(  
13         margin: EdgeInsets.all(10),  
14         child: Column(  
15             children: [  
16                 // แสดงรูปภาพ (สมมติใช้ NetworkImage)  
17                 Image.network(spot.imageUrl, height: 150, fit: BoxFit.cover),  
18                 ListTile(  
19                     title: Text(spot.name, style: TextStyle(fontWeight: FontWeight.bold)),  
20                     subtitle: Text(spot.location),  
21                     trailing: Icon(Icons.arrow_forward),  
22                 ), // ListTile  
23             ],  
24         ), // Column  
25     ); // Card  
26 }  
27 }
```



สร้างไฟล์ lib/screens/home\_screen.dart

```
lib > screens > home_screen.dart > ...
1  import 'package:flutter/material.dart';
2  import '../models/travel_spot.dart';
3  import '../widgets/travel_card.dart';
4
5  class HomeScreen extends StatelessWidget {
6
7      // 1. จัดองค์ข้อมูล (Mock Data) เป็น List<TravelSpot>
8      final List<TravelSpot> spots = [
9          TravelSpot(
10              name: 'Grand Palace',
11              location: 'Bangkok, Thailand',
12              imageUrl: 'https://link_to_image_1.jpg', // ทางกรุณาจัดให้
13          ),
14          TravelSpot(
15              name: 'Doi Inthanon',
16              location: 'Chiang Mai, Thailand',
17              imageUrl: 'https://link_to_image_2.jpg',
18          ),
19      ];
20
21      @override
22      Widget build(BuildContext context) {
23          return Scaffold(
24              appBar: AppBar(title: Text('Thailand Travel')),
25              // 2. ใช้ ListView.builder เพื่อ Loop ข้อมูลที่มีประสิทธิภาพ
26              body: ListView.builder(
27                  itemCount: spots.length,
28                  itemBuilder: (context, index) {
29                      // ส่ง Object แต่ละตัวเข้าไปใน Widget และ
30                      return TravelCard(spot: spots[index]);
31                  },
32              ), // ListView.builder
33          ); // Scaffold
34      }
35 }
```

```
1 import 'package:flutter/material.dart';
2 import '../models/travel_spot.dart';
3 import '../widgets/travel_card.dart';
4
5 class HomeScreen extends StatelessWidget {
6 }
```

```
7 // 1. จำลองข้อมูล (Mock Data) เป็น List<TravelSpot>
8 final List<TravelSpot> spots = [
9   TravelSpot(
10     name: 'Grand Palace',
11     location: 'Bangkok, Thailand',
12     imageUrl: 'https://link\_to\_image\_1.jpg', // ภาพกรุปจริง立刻
13   ),
14   TravelSpot(
15     name: 'Doi Inthanon',
16     location: 'Chiang Mai, Thailand',
17     imageUrl: 'https://link\_to\_image\_2.jpg' ,
18   ),
19 ];
```

```
21 @override
22 Widget build(BuildContext context) {
23     return Scaffold(
24         appBar: AppBar(title: Text('Thailand Travel'),
25             // 2. ใช้ ListView.builder เพื่อ Loop ข้อมูลที่มีประสิทธิภาพ
26         body: ListView.builder(
27             itemCount: spots.length,
28             itemBuilder: (context, index) {
29                 // ส่ง Object แต่ละตัวเข้าไปใน Widget แยก
30                 return TravelCard(spot: spots[index]);
31             },
32         ), // ListView.builder
33     ); // Scaffold
34 }
35 }
36 }
```

travel\_spot.o

lib &gt; screens &gt;

```
1 import
```

```
2 import
```

```
3 import
```

```
4 ?
```

```
5 class HomeScreen extends StatelessWidget {
```

```
6
```

```
7 // 1. ข้อมูลตัวอย่าง (Mock Data) : List<TravelSpot>
```

Constructors for public widgets should have a named 'key' parameter.

Try adding a named parameter to the constructor. dart([use\\_key\\_in\\_widget\\_constructors](#))

```
class HomeScreen extends StatelessWidget
```

Declared in *package:flutter\_application\_1/screens/home\_screen.dart*.

[View Problem](#) (F8) Quick Fix... (F6)

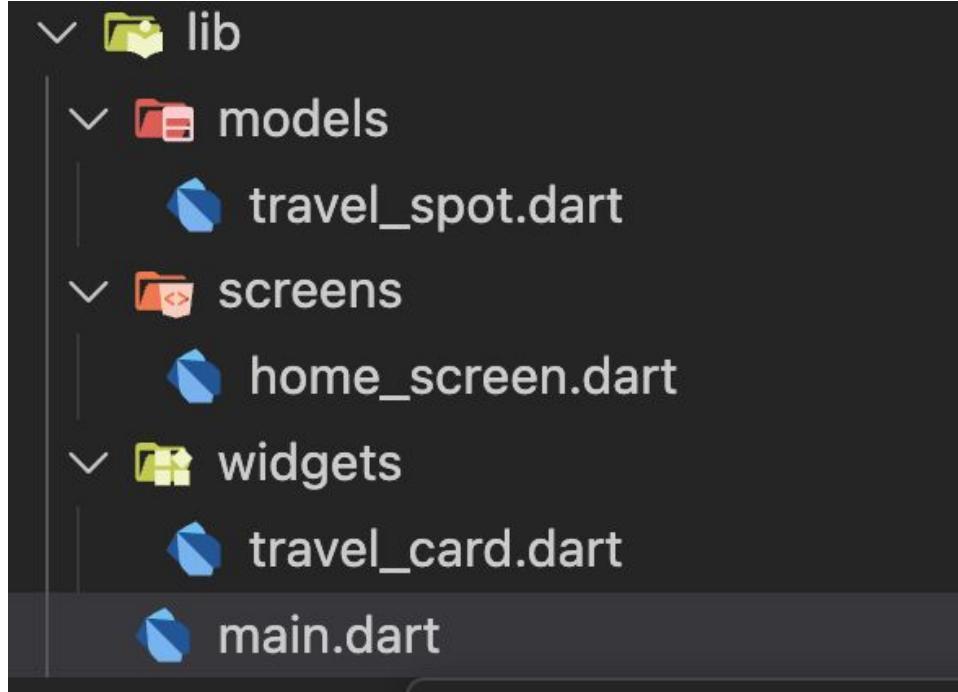
**สังเกต Warning : Constructors for public widgets should have a named 'key' parameter**

```
20
21   • HomeScreen({super.key});
22
23   @override
24   Widget build(BuildContext context) {
25     return Scaffold(
26       appBar: AppBar(title: Text('Thailand Tr
```

ໃສ const HomeScreen({super.key});

```
20
21   • HomeScreen({super.key});
22
23   @override
24   Widget build(BuildContext context) {
25     return Scaffold(
26       appBar: AppBar(title: Text('Thailand Tr...
```

เปิดช่องให้ส่งค่า Key เข้ามาได้เสมอ เพื่อกรณีที่ต้องมีการสลับ  
ตำแหน่ง Widget หรือจัดการ State  
ถึงแม้ตอนนี้เราจะยังไม่ได้ใช้ก็ตาม แต่เป็นมาตรฐานที่ต้องมี



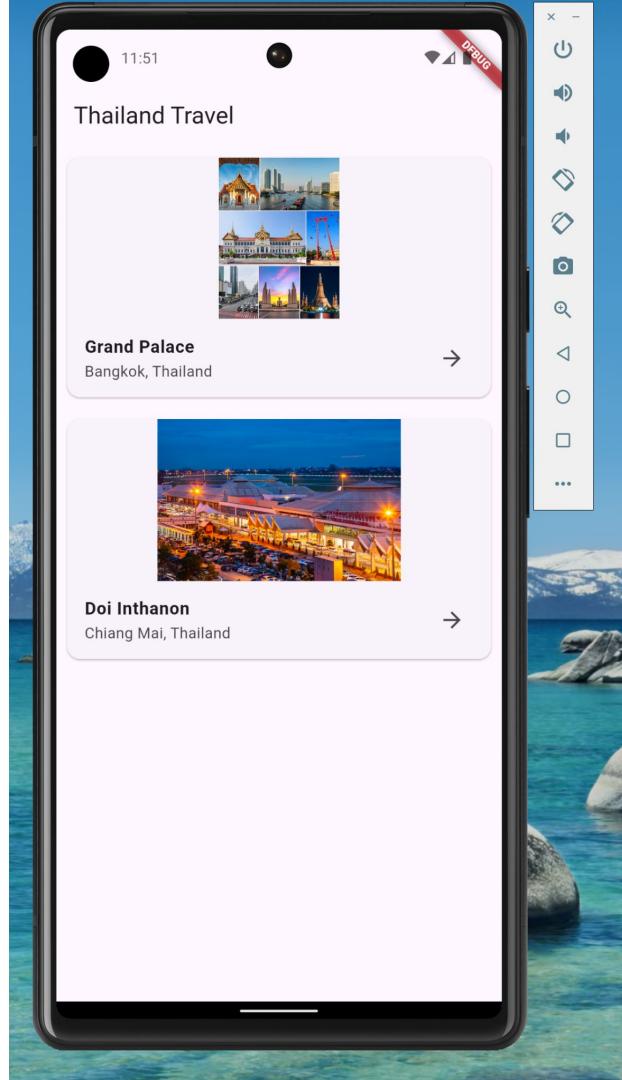
กลับมาที่ main.dart ลบขยะทิ้งให้หมด  
ให้เหลือแค่ทางเข้า (Entry Point)

lib &gt; main.dart &gt; ...

```
1 import 'package:flutter/material.dart';
2 import 'screens/home_screen.dart'; // Import หน้า Home
3
4 Run | Debug | Profile
5 void main() {
6     runApp(MyApp());
7 }
8
9 class MyApp extends StatelessWidget {
10    @override
11    Widget build(BuildContext context) {
12        return MaterialApp(
13            title: 'Travel App',
14            theme: ThemeData(primarySwatch: Colors.blue),
15            home: HomeScreen(), // เรียกใช้ HomeScreen ที่เราแยกไปแล้ว
16        ); // MaterialApp
17    }
18 }
```

 travel\_spot.dart travel\_card.dart home\_screen.dart main.dart •lib >  main.dart > ...

```
1 import 'package:flutter/material.dart';
2 import 'screens/home_screen.dart'; // Import หน้า Home
3
4 Run | Debug | Profile
5 void main() {
6   runApp(MyApp());
7 }
8 class MyApp extends StatelessWidget {
9   const MyApp({super.key});
10  @override
11  Widget build(BuildContext context) {
12    return MaterialApp(
13      title: 'Travel App',
14      theme: ThemeData(primarySwatch: Colors.blue),
15      home: HomeScreen(), // เรียกใช้ HomeScreen ที่เราแยกไปแล้ว
16    ); // MaterialApp
17 }
18 }
```

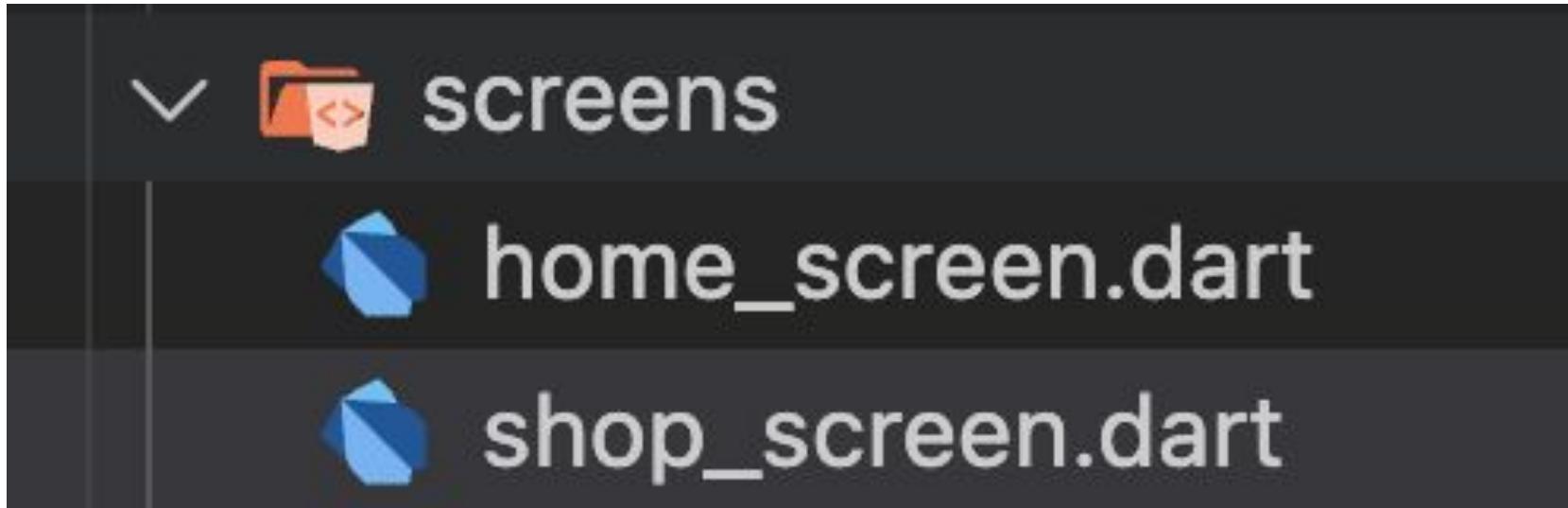


# Refactoring UI

<https://bornto.dev/FlutterKMITL68>



## Step 0: The "Messy" Code



**สร้างไฟล์ : lib/screens/shop\_screen.dart**

shop\_screen.dart X

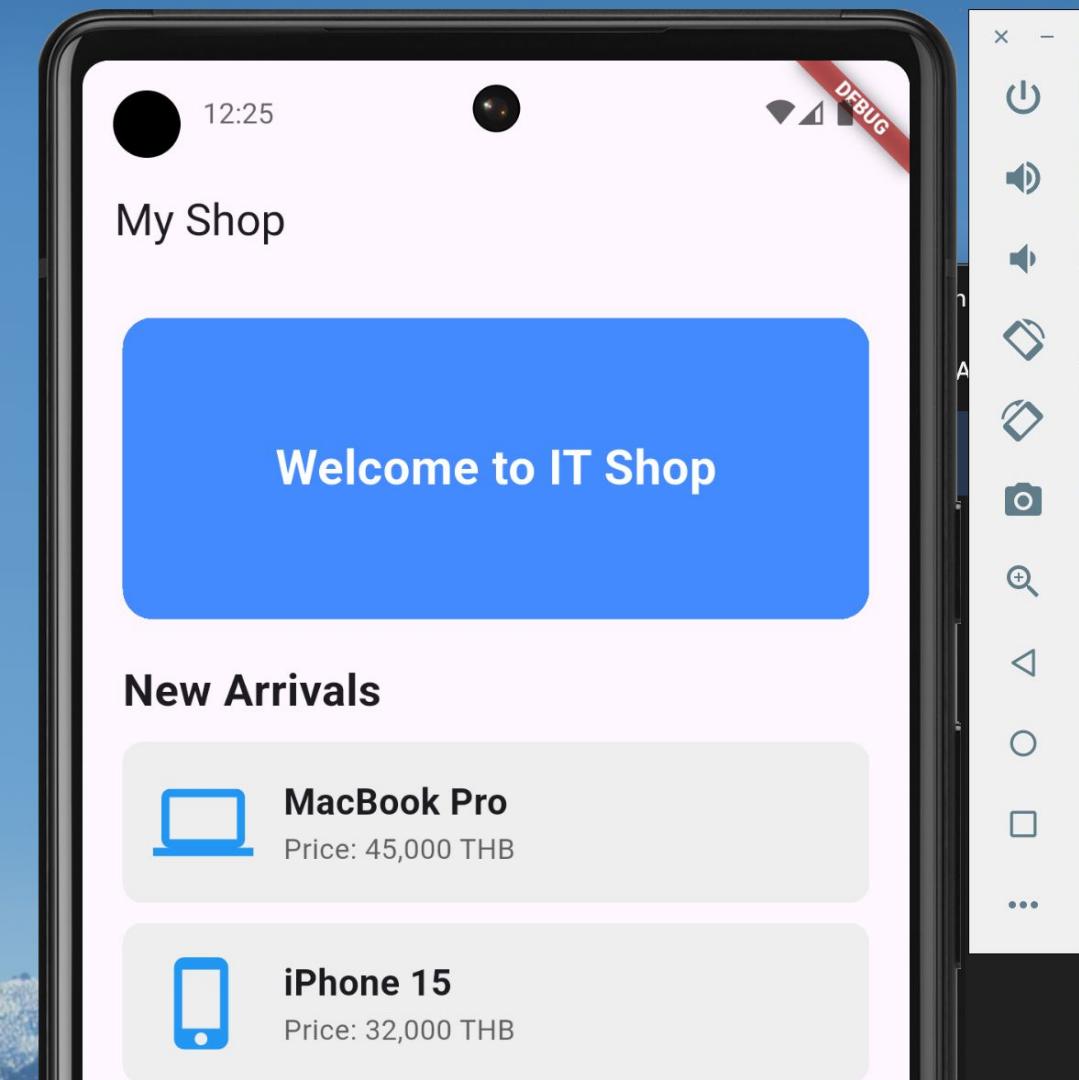
travel\_spot.dart

travel\_card.dart

home\_screen.dart

main.dart

```
lib > screens > shop_screen.dart > ShopScreen > build
1   import 'package:flutter/material.dart';
2
3   class ShopScreen extends StatelessWidget {
4     const ShopScreen({super.key});
5
6     @override
7     Widget build(BuildContext context) {
8       return Scaffold(
9         appBar: AppBar(title: Text('My Shop')),
10        body: ListView(
11          padding: EdgeInsets.all(20),
12          children: [
13            // --- សោន្ន 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.bold),
25                ), // Text
26              ), // Center
27            ), // Container
28            SizedBox(height: 20),
29
30            // --- សោន្ន 2: Section Title ---
31            Text('New Arrivals', style: TextStyle(fontSize: 22, fontWeight: FontWeight.bold)),
32            SizedBox(height: 10),
33
34            // --- សោន្ន 3: Product Items (Card) ---
```



Quiz : ทำให้น้าต่าง  
แสดงออกมาก

[shop\\_screen.dart](#)[travel\\_spot.dart](#)[travel\\_card.dart](#)[home\\_screen.dart](#)[main.dart 1](#)

```
lib > main.dart > MyApp > build
1   import 'package:flutter/material.dart';
2   import 'screens/home\_screen.dart'; // Import หน้า Home
3   import 'screens/shop\_screen.dart';
4
5   Run | Debug | Profile
6   void main() {
7     runApp(MyApp\(\));
8   }
9   class MyApp extends StatelessWidget {
10     const MyApp({super.key});
11     @override
12     Widget build(BuildContext context) {
13       return MaterialApp(
14         title: 'Travel App',
15         theme: ThemeData(primarySwatch: Colors.blue),
16         home: ShopScreen(), // เรียกใช้ HomeScreen ที่เราแยกไปแล้ว
17       ); // MaterialApp
18     }
19   }
20
21 }
```



# 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.bold),
25     ), // Text
26   ), // Center
27 ), // Container
```

**Ctrl + x**

shop\_screen.dart X

travel\_spot.dart

travel\_card.dart

home

lib &gt; screens &gt; shop\_screen.dart &gt; ShopScreen &gt; build

```
3   class ShopScreen extends StatelessWidget {
7     Widget build(BuildContext context) {
9       appBar: AppBar(title: Text('My Shop')),
10      body: ListView(
11        padding: EdgeInsets.all(20),
12        children: [
13          // --- ส่วนที่ 1: Header (Banner) ---
14          SizedBox(height: 20),
15
16          // --- ส่วนที่ 2: Section Title ---
17          Text('New Arrivals', style: TextStyle(fontSize: 22, font
18          SizedBox(height: 10),
19
20          // --- ส่วนที่ 3: Product Items (Card) ---
21          // ลิบค้าชิ้นที่ 1
```

# ตัดทิ้ง

```
10    body: ListView(  
11        padding: EdgeInsets.all(20),  
12        children: [  
13            // --- ส่วนที่ 1: Header (Banner) ---  
14            buildHeader(),  
15            SizedBox(height: 20),  
16            // --- ส่วนที่ 2: Section Title ---  
17            Text('New Arrivals', style: TextStyle(  
18                SizedBox(height: 10),
```

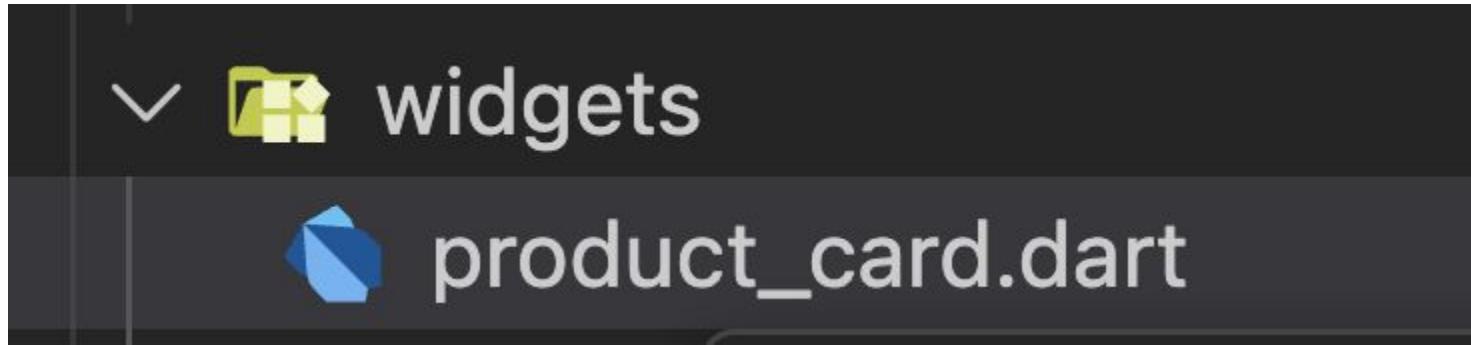
## Call Function

```
71  Widget _buildHeader() {  
72    return Container(  
73      height: 150,  
74      decoration: BoxDecoration(  
75        color: const Color.fromARGB(255, 33, 149, 243),  
76        borderRadius: BorderRadius.circular(10),  
77      ), // BoxDecoration  
78      alignment: Alignment.center,  
79      child: Text(  
80        'Welcome to My Shop',  
81        style: TextStyle(color: Colors.white, fontSize: 24, fontWeight: FontWeight.bold),  
82      ), // Text  
83    ); // Container  
84  }  
85 }
```

# สร้าง Widget



## Step 2: Extract Widget (สร้าง Class ใหม่เพื่อใช้ซ้ำ)



สร้างไฟล์ใหม่ lib/widgets/product\_card.dart

The screenshot shows a dark-themed code editor interface. At the top, there are tabs for 'shop\_screen.dart', 'product\_card.dart' (which has a red dot indicating changes), 'travel\_spot.dart', and 'travel.dart'. Below the tabs, the file path is shown as 'lib > widgets > product\_card.dart > [o] stless'. A lightbulb icon with the number '1' indicates a suggestion. The word 'stless' is underlined with a red wavy line, suggesting it's misspelled. A dropdown menu is open, listing four suggestions:

- Flutter **Stateless** Widget
- statelessW
- StatelessElement
- StatelessElement

Details for the first suggestion are provided: 'Stateless Widget' and 'package:flutter/material.dart'. The second suggestion is 'package:flutter/widgets.dart'.

พิมพ์ **stless** และ Enter เพื่อสร้าง Class เป็นๆ  
ชื่อ ProductCard

shop\_screen.dart

product\_card.dart

travel\_spo

```
lib > widgets > product_card.dart > MyWidget
1 import 'package:flutter/material.dart';
2
3 class MyWidget extends StatelessWidget {
4   const MyWidget({super.key});
5
6   @override
7   Widget build(BuildContext context) {
8     return const Placeholder();
9   }
10 }
```

พิมพ์ `stless` และ `Enter` เพื่อสร้าง Class เป็นล่าชื่อ `ProductCard`

The screenshot shows a code editor with three tabs at the top: 'shop\_screen.dart', 'product\_card.dart' (which is the active tab), and 'travel'. The code in 'product\_card.dart' is as follows:

```
lib > widgets > product_card.dart > ProductCard
1 import 'package:flutter/material.dart';
2
3 class ProductCard extends StatelessWidget {
4     const ProductCard({super.key});
5
6     @override
7     Widget build(BuildContext context) {
8         return const Placeholder();
9     }
10 }
```

พิมพ์ `stless` และ `Enter` เพื่อสร้าง Class เป็นๆ  
ชื่อ `ProductCard`

```
shop_screen.dart product_card.dart ✘ travel_spot.dart travel_card.dart home_screen.d
lib > widgets > product_card.dart > ProductCard > build
30  }
6   @override
7   Widget build(BuildContext context) {
8     return Container(
9       padding: EdgeInsets.all(15),
10      margin: EdgeInsets.only(bottom: 10),
11      decoration: BoxDecoration(
12        color: Colors.grey[200],
13        borderRadius: BorderRadius.circular(10),
14      ), // BoxDecoration
15      child: Row(
16        children: [
17          Icon(Icons.laptop, size: 50, color: Colors.blue),
18          SizedBox(width: 15),
19          Column(
20            crossAxisAlignment: CrossAxisAlignment.start,
21            children: [
22              Text('MacBook Pro', style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold)),
23              Text('Price: 45,000 THB', style: TextStyle(color: Colors.grey[700])),
24            ],
25          ), // Column
26        ],
27      ), // Row
28    ); // Container
29 }
```

Copy ส่วน Code Container สินค้าจากหน้าเดิม  
มาวางใน build ของไฟล์ใหม่

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

เอามาใช้ว่าให้มีกำลังใจ

```
20          // --- ส่วนที่ 3: Product Items (Card) ---
21      ProductCard(),
22
23      Enter to Apply, ⌘Enter to Preview
24
25      ⚡ Import library 'package:flutter_application_1/widgets/product_card.dart'
26
27      ⚡ Import library 'package:flutter_application_1/widgets/product_card.dart' with 'show'
28      Wj
29      ⚡ Import library '../widgets/product_card.dart'
30      ⚡ Import library '../widgets/product_card.dart' with 'show'
```

ເອົາມາໃຊ້ວ່າໄໝກຳລັງໄຈ

shop\_screen.dart X

product\_card.dart

travel\_spot.dart

travel\_card.dart

lib &gt; screens &gt; shop\_screen.dart &gt; ...

```
1 import 'package:flutter/material.dart';
2 import 'package:flutter_application_1/widgets/product_card.dart';
3 |
4 class ShopScreen extends StatelessWidget {
5     const ShopScreen({super.key});
6
7     @override
8     Widget build(BuildContext context) {
9         return Scaffold(
10             appBar: AppBar(title: Text('My Shop')),
```

ເຄົາມາໂຈວ່າໃໝ່ມີກຳລັງໄຈ

shop\_screen.dart X

product\_card.dart

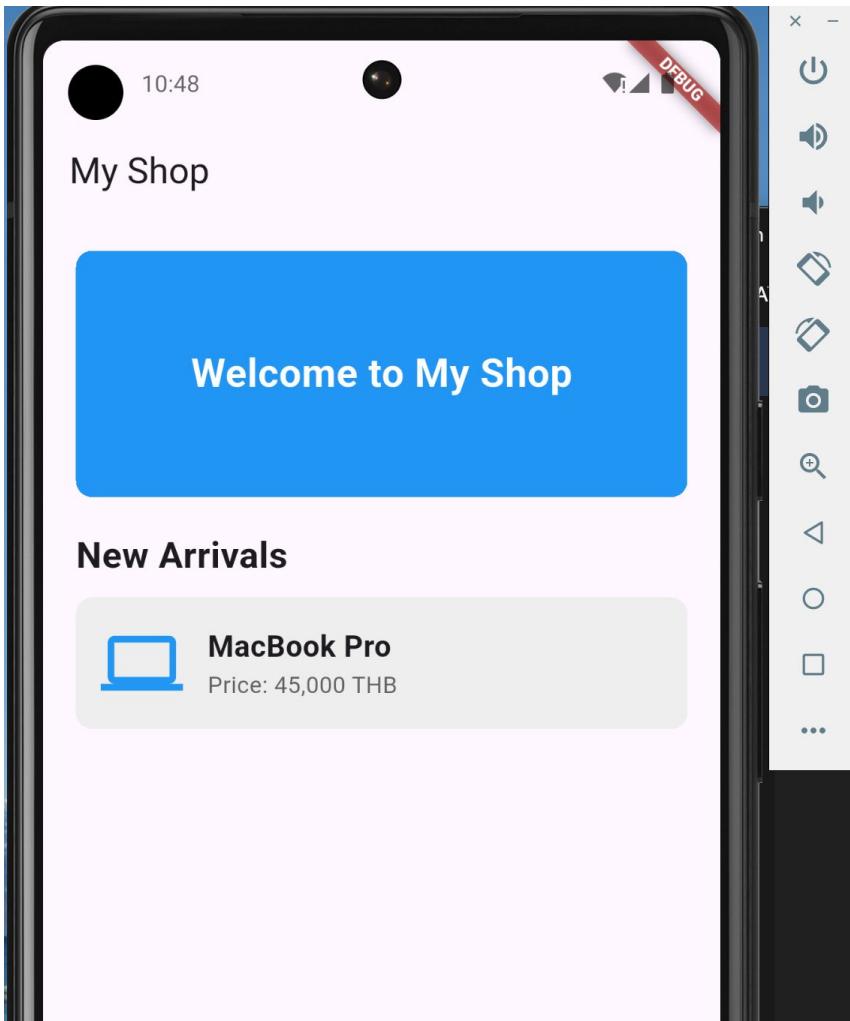
travel\_spot.dart

travel\_card.dart

lib &gt; screens &gt; shop\_screen.dart &gt; ...

```
1 import 'package:flutter/material.dart';
2 import 'package:flutter_application_1/widgets/product_card.dart';
3
4 class ShopScreen extends StatelessWidget {
5     const ShopScreen({super.key});
6
7     @override
8     Widget build(BuildContext context) {
9         return Scaffold(
10             appBar: AppBar(title: Text('My Shop')),
```

ເຄົາມາໂຈວ່າໃໝ່ມີກຳລັງໄຈ





## Step 3: Parameter & Constructor (หัวใจสำคัญ)

```
shop_screen.dart 3          product_card.dart ●          travel_spot.dart
lib > widgets > product_card.dart > ProductCard > build
1   import 'package:flutter/material.dart';
2
3   class ProductCard extends StatelessWidget {
4
5       // 1. ประกาศตัวแปร (Fields) : จงที่ไว้เก็บข้อมูล
6       // ต้องใส่ final เพราะ StatelessWidget ห้ามตัวแปรเปลี่ยนค่า
7       final String name;
8       final String price;
9       final IconData icon; // เพิ่ม Icon ด้วยจะได้ครบสูตร
10
11      // 2. แก้ไข Constructor ทำหน้าที่ "รับของ" จากภายนอก
12      const ProductCard({
13          super.key,
14          required this.name,    // required = บังคับว่าต้องส่งมาแน่ ห้ามลีฟ
15          required this.price,   // รับมาแล้วเอาไปยัดใส่ตัวแปร price ข้างบน
16          required this.icon,
17      });
18
```

ประกาศตัวแปรรับค่า ใช้ final  
 เพราะ Widget เป็น immutable

lib > widgets > product\_card.dart > ProductCard > build

```
1 import 'package:flutter/material.dart';
2
3 class ProductCard extends StatelessWidget {
4
5     // 1. ประกาศตัวแปร (Fields) : จองที่ໄວ້ເກີບຂໍ້ມູນ
6     // ຕ້ອງໃສ່ final ເພົ່າ StatelessWidget ທໍາມຕັ້ງແປຣເປີ່ຍນຄ່າ
7     final String name;
8     final String price;
9     final IconData icon; // ເພີ່ມ Icon ດ້ວຍຈະໄດ້ຮັບສູດຮຣ
10
11    // 2. ແກ້ໄຂ Constructor ທໍາທຳກຳ "ຮັບຂອງ" ຈາກກາຍນອກ
12    const ProductCard({
13        super.key,
14        required this.name,    // required = ບັນກັບວ່າຕ້ອງສົ່ງມານະ ທໍາມລື່ມ
15        required this.price,   // ຮັບມາແລ້ວເອົາໄປຢັດໃສ່ຕັ້ງແປຣ price ຂ້າງບນ
16        required this.icon,
17    });
18
```

```
@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(
                    mainAxisAlignment: MainAxisAlignment.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
```

1

ใช้ของที่รับมาแทนการ Hardcode

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

```
ProductCard(
```

```
  name: 'MacBook Pro',
```

```
  price: '450,000 THB',
```

```
  icon: Icons.laptop,
```

```
), // ProductCard
```

1

**Next Challenge:** "ลองเพิ่มสินค้าอีก 2 ชิ้น และลองเปลี่ยน  
ลีส์ ProductCard โดยแก้ที่ไฟล์เดียวดูสิ"

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

```
ProductCard(
```

```
  name: 'MacBook Pro',
```

```
  price: '450,000 THB',
```

```
  icon: Icons.laptop,
```

```
), // ProductCard
```

1

**Next Challenge:** "ลองเพิ่มสินค้าอีก N ชิ้น ใน List แล้ว  
แสดงออกมา"

shop\_screen.dart

product\_card.dart

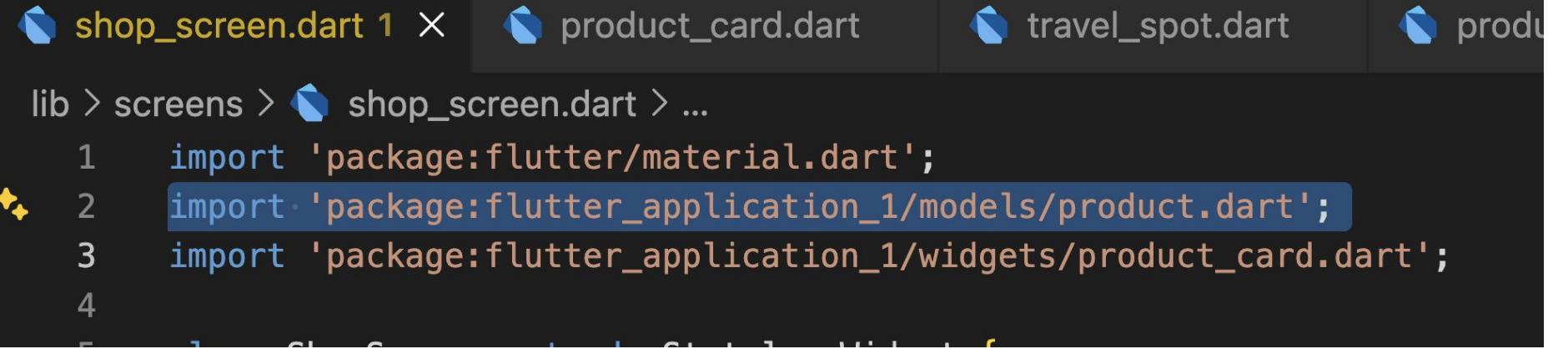
travel\_spot.dart

product.dart X

lib > models > product.dart > Product

```
1 import 'package:flutter/material.dart';
2
3 class Product {
4     final String name;
5     final String price;
6     final IconData icon;
7
8     Product({required this.name, required this.price, required this.icon});
9 }
```

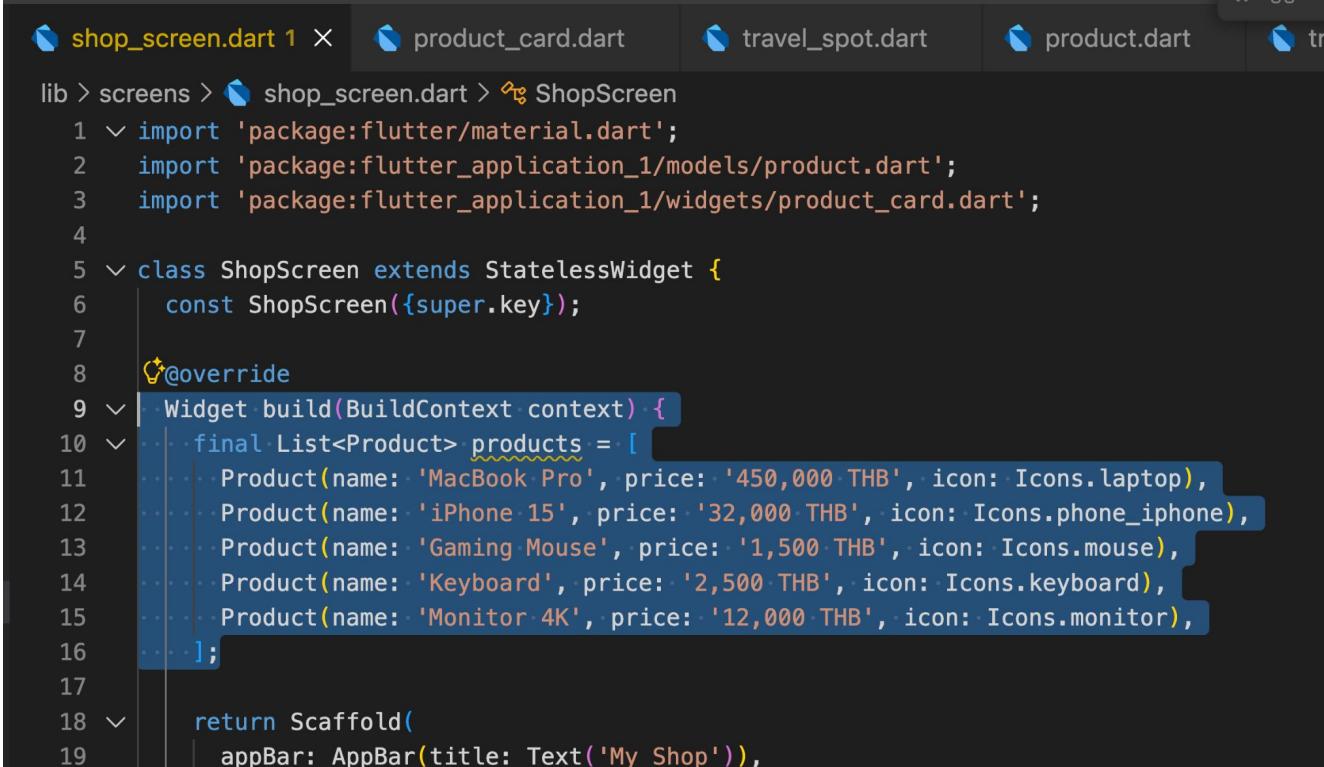
# แยกไฟล์ models/product.dart



The screenshot shows a code editor with several tabs at the top: shop\_screen.dart, product\_card.dart, travel\_spot.dart, and product. The current file is shop\_screen.dart, which contains the following code:

```
lib > screens > shop_screen.dart > ...
1 import 'package:flutter/material.dart';
2 import 'package:flutter_application_1/models/product.dart';
3 import 'package:flutter_application_1/widgets/product_card.dart';
4
```

ใน ShopScreen ให้สร้างตัวแปร List ขึ้นมาเพื่อเก็บ  
ข้อมูลสินค้า N ชิ้นที่เราต้องการ



```
shop_screen.dart 1 × product_card.dart travel_spot.dart product.dart tra
lib > screens > shop_screen.dart > ShopScreen
1 ✓ import 'package:flutter/material.dart';
2   import 'package:flutter_application_1/models/product.dart';
3   import 'package:flutter_application_1/widgets/product_card.dart';
4
5 ✓ class ShopScreen extends StatelessWidget {
6   const ShopScreen({super.key});
7
8   @override
9   Widget build(BuildContext context) {
10  final List<Product> products = [
11    Product(name: 'MacBook Pro', price: '450,000 THB', icon: Icons.laptop),
12    Product(name: 'iPhone 15', price: '32,000 THB', icon: Icons.phone_iphone),
13    Product(name: 'Gaming Mouse', price: '1,500 THB', icon: Icons.mouse),
14    Product(name: 'Keyboard', price: '2,500 THB', icon: Icons.keyboard),
15    Product(name: 'Monitor 4K', price: '12,000 THB', icon: Icons.monitor),
16  ];
17
18   return Scaffold(
19     appBar: AppBar(title: Text('My Shop')),
```

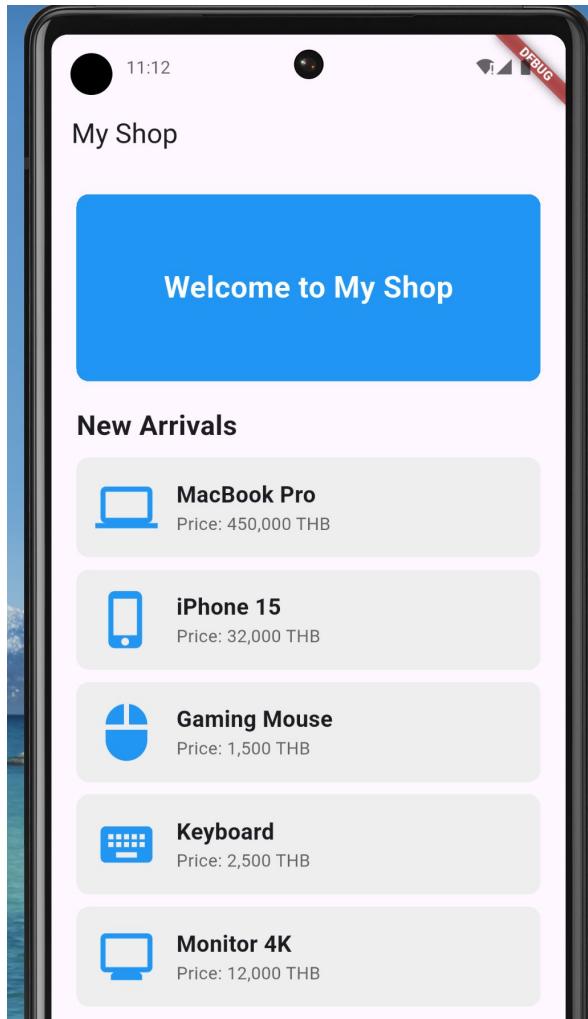
ใน ShopScreen ให้สร้างตัวแปร List ขึ้นมาเพื่อเก็บข้อมูลสินค้า N ชิ้นที่เราต้องการ

lib > screens > shop\_screen.dart > ShopScreen

```
1  ↘ import 'package:flutter/material.dart';
2  import 'package:flutter_application_1/models/product.dart';
3  import 'package:flutter_application_1/widgets/product_card.dart';
4
5  ↘ class ShopScreen extends StatelessWidget {
6      const ShopScreen({super.key});
7
8      ↗@override
9      Widget build(BuildContext context) {
10         final List<Product> products = [
11             Product(name: 'MacBook Pro', price: '450,000 THB', icon: Icons.laptop),
12             Product(name: 'iPhone 15', price: '32,000 THB', icon: Icons.phone_iphone),
13             Product(name: 'Gaming Mouse', price: '1,500 THB', icon: Icons.mouse),
14             Product(name: 'Keyboard', price: '2,500 THB', icon: Icons.keyboard),
15             Product(name: 'Monitor 4K', price: '12,000 THB', icon: Icons.monitor),
16         ];
17
18         return Scaffold(
19             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** แทนที่เราจะ  
**ProductCard** ทีละอัน เราจะใช้ **Collection For** (การวน  
ลูปใน **List** ของ **Widget**) แทน



อยากย้ายหน้าไปมา  
ทำอย่างไร?

docs.flutter.dev/cookbook/navigation/navigation-basics

Flutter Docs

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

Cookbook > Navigation > Navigate to a new screen and back

# 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()`.

On this page

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

Interactive example

Additional navigation methods

# 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:

# เพิ่มหน้า ก่อนลง API

```
182  ↘ class SecondRoute extends StatelessWidget {  
183      |   const SecondRoute({super.key});  
184  
185      |   @override  
186      ↘   Widget build(BuildContext context) {  
187          |       return Scaffold(  
188              |           appBar: AppBar(  
189                  |               title: const Text('Second Route'),  
190                  |           ), // AppBar  
191              |           body: Center(  
192                  |               child: ElevatedButton(  
193                      |                   onPressed: () {  
194                          |                       // Navigate back to first route when tapped.  
195                      |                   },  
196                      |                   child: const Text('Go back!'),  
197                      |               ), // ElevatedButton  
198                  |           ), // Center  
199              |       ); // Scaffold  
200      }  
201 }
```

## 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.

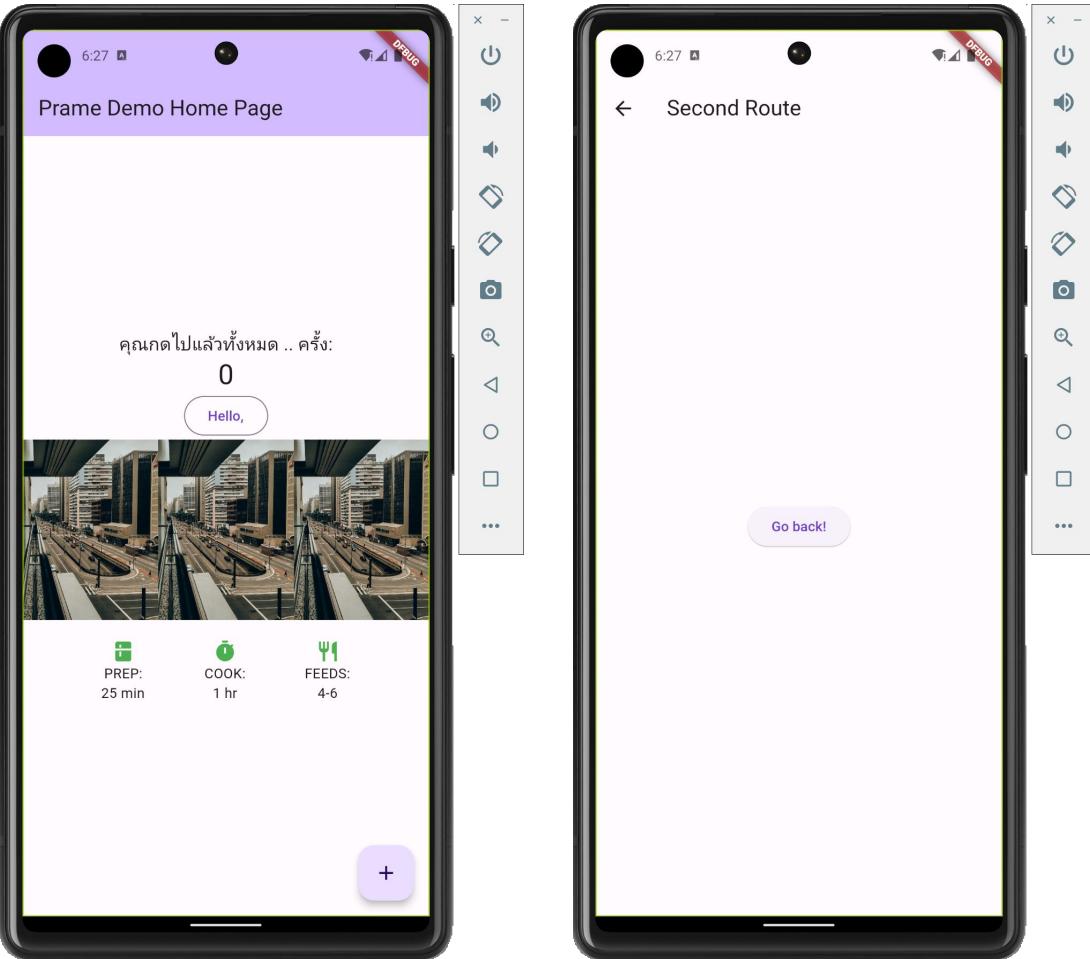
In the `build()` method of the `FirstRoute` widget, update the `onPressed()` callback:

# 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(
```



# เพิ่มหน้า ที่สองลงไป



### 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.

ลังค่าข้ามหน้า ทำอย่างไร ?

# Todos

Todo 2

Todo 3

Todo 4

Todo 5

Todo 6

Todo 7

Todo 8

DEBUG

## ← Todo 6

A description of what needs to be done for  
Todo 6

DEBUG

BURNTO  
DEV

# 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

```
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!

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



## 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

Todo 2

Todo 3

Todo 4

Todo 5

Todo 6

Todo 7

Todo 8

DEBUG

## ← Todo 6

A description of what needs to be done for  
Todo 6

DEBUG

BURNTO  
DEV

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.

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

## Nope button

```
dart\nElevatedButton(\n  onPressed: () {\n    // Close the screen and return "Nope." as the result.\n    Navigator.pop(context, 'Nope.');
```

}

```
  child: const Text('Nope.'),\n)
```

## 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

BURNTO  
DEV



## 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`:

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

dart

## 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.
  ),
);
```

### 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



## Index 1: Business

# การบ้านสำหรับเช็คชื่อรอบถัดไป ..

-  สร้างแอปพลิเคชันที่มีการ Navigate
-  มีหน้าอย่างน้อย 3 หน้า
-  มีการส่งข้อมูลข้ามหน้ากัน
-  เป็นแอปพลิเคชันที่มีใช้ได้จริง

PROJECT รวมกลุ่ม 3 คน

Technical 25% Idea 15%