Flutter

Flutter Mobile Programming

Intermediate

(Integrative-Generative AI Edition)

Student Price Book Center

คำนำ

การพัฒนาแอปพลิเคชันบนแพลตฟอร์มมือถือในยุคปัจจุบันมีความท้าทายและความซับซ้อนเพิ่มมากขึ้นอย่างต่อเนื่อง นักพัฒนาจำเป็นต้องมีความรู้และทักษะที่หลากหลาย ไม่เพียงแต่ด้านการเขียนโค้ดพื้นฐาน แต่ยังรวมถึงการออกแบบ UI/UX การจัดการข้อมูลภายในแอป การสื่อสารกับ API ภายนอก รวมถึงการจัดการสถานะของแอปให้อยู่ในสภาพที่เหมาะสมและมีประสิทธิภาพสูงสุด เพื่อตอบสนองต่อความต้องการและพฤติกรรมของผู้ใช้อย่างแท้จริง หนังสือ *Flutter Mobile Programming: Intermediate* เล่มนี้จึงถูกเขียนขึ้นเพื่อเติมเต็มช่องว่างและเพิ่มพูนความรู้ในระดับกลางสำหรับผู้ที่มีพื้นฐาน Flutter มาแล้ว และต้องการก้าวสู่การพัฒนาแอปที่ครบวงจรและมีประสิทธิภาพ

หนังสือเล่มนี้เน้นการเรียนรู้เชิงลึกในประเด็นสำคัญที่ใช้บ่อยและเป็นหัวใจของการพัฒนาแอปมือถือด้วย Flutter ตั้งแต่ระบบการนำทาง (Navigation และ Routing) ซึ่งเป็นโครงสร้างสำคัญที่ช่วยให้แอปพลิเคชันสามารถเปลี่ยนหน้าจอและส่งผ่านข้อมูลได้อย่างมีประสิทธิภาพ ไปจนถึงการแสดงข้อมูลในรูปแบบไดนามิกผ่าน ListView และ GridView ซึ่งเป็นหัวใจของการจัดการข้อมูลที่มีจำนวนมากและเปลี่ยนแปลงได้ตลอดเวลา นอกจากนี้ยังเจาะลึกการจัดเก็บข้อมูลทั้งชั่วคราวและถาวรในเครื่อง ด้วย SharedPreferences, SQLite และ Local JSON File เพื่อให้แอปพลิเคชันสามารถทำงานได้ต่อเนื่องแม้ในสถานการณ์ออฟไลน์

อีกส่วนที่สำคัญไม่แพ้กันคือการเชื่อมต่อกับ API ภายนอกผ่าน HTTP Request ซึ่งเป็นช่องทางหลักในการนำข้อมูลจากแหล่งข้อมูลภายนอกเข้าสู่แอป การเรียนรู้การใช้ http package, การแปลงข้อมูล JSON ให้เป็น Dart Model รวมไปถึงการจัดการสถานะการโหลดและข้อผิดพลาด ทำให้ผู้อ่านสามารถสร้างแอปที่มีการสื่อสารกับเซิร์ฟเวอร์ได้อย่างราบรื่นและมีประสิทธิภาพ

ในส่วนของการจัดการ State ระดับกลาง หนังสือเล่มนี้แนะนำเทคนิคและเครื่องมือยอดนิยมอย่าง Provider และ ChangeNotifier ที่ช่วยให้การแยก Business Logic ออกจาก UI เป็นไปอย่างเป็นระบบ เพิ่มความง่ายในการบำรุงรักษาและพัฒนาต่อยอด รวมถึงตัวอย่างการประยุกต์ใช้จริงที่ชัดเจนและเข้าใจง่าย

สุดท้าย การตกแต่ง UI และการจัดการ Theme ได้รับการถ่ายทอดอย่างละเอียดเพื่อให้แอปพลิเคชันมีรูปลักษณ์ที่โดดเด่นและรองรับการใช้งานในทุกสถานการณ์ เช่น Dark Mode การใช้ Custom Fonts และ Icons รวมถึงการสร้าง Custom Widget ที่นำกลับมาใช้ซ้ำได้ เพื่อเพิ่มประสิทธิภาพในการพัฒนาและความสวยงามของแอป

หนังสือเล่มนี้รวบรวมเนื้อหาที่จำเป็นและครบถ้วนสำหรับผู้ที่ต้องการพัฒนาทักษะ Flutter ในระดับกลาง พร้อมด้วยตัวอย่างโค้ดจริงและคำอธิบายเชิงลึกที่ช่วยให้ผู้อ่านสามารถนำไปประยุกต์ใช้ได้ทันทีทั้งในโครงการเรียนรู้และงานจริง ด้วยเนื้อหาที่ครอบคลุมตั้งแต่บทที่ 6 ถึงบทที่ 11 ผู้อ่านจะได้รับทั้งความรู้ ความเข้าใจ และความมั่นใจในการพัฒนาแอปที่มีคุณภาพและตอบโจทย์การใช้งานในยุคดิจิทัลนี้อย่างเต็มที่

ขอให้ผู้อ่านได้รับประโยชน์สูงสุดจากหนังสือเล่มนี้ และสามารถก้าวไปสู่การเป็นนักพัฒนา Flutter มืออาชีพได้อย่างมั่นใจและประสบความสำเร็จในทุกโปรเจกต์ที่ท่านเลือกทำ

ด้วยรักและปรารถนาดี

ศูนย์หนังสือราคานักเรียน

สารบัญ

หน้า

บทที่ 6

Navigation และ Routing

(Navigation and Routing)

เนื้อหา

- Navigation และ Routing ใน Flutter
- Navigation และ Routing — รายละเอียดเชิงลึก
- การเปลี่ยนหน้าโดยใช้ Navigator.push/pop ใน Flutter
- Named Routes ใน Flutter
- ส่งข้อมูลระหว่างหน้า (Passing Data Between Screens)
- BottomNavigationBar และ TabBar ใน Flutter

บทนำ บทที่ 6: Navigation และ Routing

การพัฒนาแอปพลิเคชันสมัยใหม่ไม่เพียงแต่ต้องมีหน้าตาที่สวยงามและฟังก์ชันที่ครบถ้วนเท่านั้น แต่ยังต้องมีโครงสร้างการนำทาง (Navigation) ที่ชัดเจนและใช้งานง่าย เพื่อให้ผู้ใช้สามารถเข้าถึงข้อมูลและฟังก์ชันต่าง ๆ ได้อย่างรวดเร็วและไม่สับสน บทที่ 6 นี้จะมุ่งเน้นการอธิบายแนวคิดและการใช้งาน Navigation และ Routing ในการพัฒนาแอป ซึ่งเป็นหัวใจสำคัญของการสร้างประสบการณ์ผู้ใช้ที่ราบรื่นและต่อเนื่อง

ในตอนแรก เราจะเริ่มจากการทำความเข้าใจกับ **การเปลี่ยนหน้าโดยใช้ Navigator.push และ Navigator.pop** ซึ่งเป็นวิธีการพื้นฐานที่สุดในการจัดการเส้นทางการนำทาง ผู้พัฒนาจะได้เรียนรู้การเปิดหน้าใหม่ (push) และย้อนกลับไปหน้าก่อนหน้า (pop) รวมถึงการประยุกต์ใช้เพื่อสร้างโฟลว์การทำงานแบบลำดับขั้นตอน (sequential flow) ที่เข้าใจง่ายและควบคุมได้

ต่อมา เราจะศึกษา **Named Routes** ซึ่งเป็นอีกหนึ่งวิธีการนำทางที่มีความยืดหยุ่นและเหมาะกับแอปที่มีหลายหน้าและโครงสร้างซับซ้อน การใช้ Named Routes ช่วยให้โค้ดมีความกระชับ ง่ายต่อการดูแล และลดความผิดพลาดที่อาจเกิดจากการเขียนเส้นทางซ้ำซ้อน ผู้พัฒนาจะได้เห็นตัวอย่างการกำหนดชื่อเส้นทางในส่วนของการตั้งค่า (route settings) และการเรียกใช้งานผ่านชื่อที่กำหนดไว้

หัวข้อถัดไปจะกล่าวถึง **การส่งข้อมูลระหว่างหน้า** ซึ่งเป็นความสามารถที่สำคัญอย่างยิ่งเมื่อเราต้องส่งค่าหรืออ็อบเจ็กต์จากหน้าหนึ่งไปยังอีกหน้าหนึ่ง ไม่ว่าจะเป็นการส่งข้อมูลจากหน้ารายการไปยังหน้ารายละเอียด หรือจากแบบฟอร์มไปยังหน้าสรุป ผู้พัฒนาจะได้เรียนรู้ทั้งวิธีการส่งข้อมูลผ่านพารามิเตอร์และการรับข้อมูลกลับจากหน้าอื่นเพื่อประมวลผลต่อ

นอกจากนี้ บทนี้ยังครอบคลุมการใช้งาน **BottomNavigationBar** และ **TabBar** ซึ่งเป็นองค์ประกอบ UI ที่ช่วยให้ผู้ใช้สามารถสลับระหว่างหน้าหรือส่วนต่าง ๆ ของแอปได้อย่างรวดเร็ว โดยไม่ต้องย้อนกลับไปยังหน้าเมนูหลัก การออกแบบและใช้งาน Navigation รูปแบบนี้ต้องคำนึงถึงความสม่ำเสมอของการจัดวาง ปฏิสัมพันธ์ของผู้ใช้ และความเหมาะสมกับเนื้อหาของแต่ละแท็บ

บทนี้จะไม่เพียงแต่ให้ความรู้ด้านทฤษฎีเท่านั้น แต่ยังมีตัวอย่างโค้ดแบบเต็มรูปแบบเพื่อให้ผู้อ่านสามารถทดลองและปรับใช้ได้จริง การผสมผสานระหว่างการเรียนรู้เชิงแนวคิดและการลงมือปฏิบัติจะช่วยเสริมสร้างความเข้าใจอย่างลึกซึ้งและความมั่นใจในการนำไปใช้

สุดท้าย การจัดการ Navigation และ Routing อย่างมีประสิทธิภาพไม่เพียงช่วยให้ผู้ใช้ได้รับประสบการณ์ที่ดีขึ้น แต่ยังช่วยให้การพัฒนาและบำรุงรักษาแอปทำได้ง่ายขึ้นในระยะยาว ด้วยแนวทางและเทคนิคที่ได้จากบทนี้ ผู้อ่านจะสามารถออกแบบและสร้างระบบนำทางที่ทั้งใช้งานง่าย ยืดหยุ่น และรองรับการขยายในอนาคตได้อย่างมั่นใจ

Navigation และ Routing ใน Flutter

- การเปลี่ยนหน้าโดยใช้ Navigator.push/pop
- Named Routes
- ส่งข้อมูลระหว่างหน้า
- BottomNavigationBar และ TabBar

---

1. การเปลี่ยนหน้าโดยใช้ Navigator.push/pop

แนวคิด

- Navigator คือวิดเจ็ตที่จัดการสแต็กของหน้า (Route Stack)
- push() คือการเพิ่มหน้าใหม่ไปยังสแต็ก (เปิดหน้าใหม่)
- pop() คือการลบหน้าปัจจุบันออกจากสแต็ก (ย้อนกลับ)

ตัวอย่าง

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

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
```

```
    return MaterialApp(home: const FirstPage());
  }
}

class FirstPage extends StatelessWidget {
  const FirstPage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('First Page')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go to Second Page'),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => const SecondPage()),
            );
          },
        ),
      ),
    );
  }
}

class SecondPage extends StatelessWidget {
  const SecondPage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Second Page')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go Back'),
```

```
            onPressed: () {
              Navigator.pop(context);
            },
          ),
        ),
      );
  }
}
```

---

2. Named Routes

แนวคิด

- กำหนดชื่อเส้นทาง (route names) ใน MaterialApp ผ่าน routes map
- เรียกใช้ Navigator.pushNamed เพื่อเปลี่ยนหน้าโดยใช้ชื่อแทนหน้า Widget ตรงๆ
- ช่วยให้ออกแบบเส้นทางโปรเจกต์ได้ง่ายและเป็นระเบียบ

ตัวอย่าง

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

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => const FirstPage(),
        '/second': (context) => const SecondPage(),
      },
    );
  }
}
```

```
class FirstPage extends StatelessWidget {
  const FirstPage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('First Page')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go to Second Page'),
          onPressed: () {
            Navigator.pushNamed(context, '/second');
          },
        ),
      ),
    );
  }
}

class SecondPage extends StatelessWidget {
  const SecondPage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Second Page')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go Back'),
          onPressed: () {
            Navigator.pop(context);
          },
        ),
      ),
    );
```

```
  }
}
```

---

3. การส่งข้อมูลระหว่างหน้า

แนวคิด

- ส่งข้อมูลผ่าน constructor ของหน้าปลายทาง (ใน push/pushNamed)
- หรือ รับข้อมูลกลับผ่าน Future (เช่น push แล้ว await ผลลัพธ์ที่ pop กลับมา)

ตัวอย่างส่งข้อมูลแบบส่งไปอย่างเดียว

```
class FirstPage extends StatelessWidget {
  // ...
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // ...
      body: Center(
        child: ElevatedButton(
          child: const Text('Go to Detail Page with Data'),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => DetailPage(data: 'Hello from FirstPage'),
              ),
            );
          },
        ),
      ),
    );
  }
}

class DetailPage extends StatelessWidget {
  final String data;
  const DetailPage({super.key, required this.data});
```

```
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Detail Page')),
      body: Center(child: Text(data, style: const TextStyle(fontSize: 24))),
    );
  }
}
```

ตัวอย่างรับค่ากลับ (Result) จากหน้า

```
class FirstPage extends StatelessWidget {
  // ...
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      // ...
      body: Center(
        child: ElevatedButton(
          child: const Text('Pick a Color'),
          onPressed: () async {
            final result = await Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => const ColorPickerPage()),
            );
            ScaffoldMessenger.of(context).showSnackBar(
              SnackBar(content: Text('You picked: $result')),
            );
          },
        ),
      ),
    );
  }
}
```

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Pick a Color')),
      body: ListView(
        children: [
          ListTile(
            title: const Text('Red'),
            onTap: () => Navigator.pop(context, 'Red'),
          ),
          ListTile(
            title: const Text('Green'),
            onTap: () => Navigator.pop(context, 'Green'),
          ),
          ListTile(
            title: const Text('Blue'),
            onTap: () => Navigator.pop(context, 'Blue'),
          ),
        ],
      ),
    );
  }
}
```

---

4. BottomNavigationBar และ TabBar

BottomNavigationBar

- ใช้สร้างเมนูด้านล่างเพื่อสลับหน้าจอ
- มักใช้ร่วมกับ IndexedStack หรือ PageView เพื่อรักษาสถานะแต่ละหน้า

ตัวอย่าง BottomNavigationBar

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

```
  @override
  State<BottomNavExample> createState() => _BottomNavExampleState();
}

class _BottomNavExampleState extends State<BottomNavExample> {
  int _selectedIndex = 0;

  static const List<Widget> _pages = [
    Center(child: Text('Home Page', style: TextStyle(fontSize: 24))),
    Center(child: Text('Search Page', style: TextStyle(fontSize: 24))),
    Center(child: Text('Profile Page', style: TextStyle(fontSize: 24))),
  ];

  void _onItemTapped(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Bottom Navigation Example')),
      body: _pages[_selectedIndex],
      bottomNavigationBar: BottomNavigationBar(
        currentIndex: _selectedIndex,
        onTap: _onItemTapped,
        items: const [
          BottomNavigationBarItem(icon: Icon(Icons.home), label: 'Home'),
          BottomNavigationBarItem(icon: Icon(Icons.search), label: 'Search'),
          BottomNavigationBarItem(icon: Icon(Icons.person), label: 'Profile'),
        ],
      ),
    );
```

```
  }
}
```

TabBar กับ TabBarView

- ใช้สร้างแท็บบนส่วนบน (มักอยู่ใน AppBar)
- ต้องใช้ DefaultTabController ควบคุมแท็บ

ตัวอย่าง TabBar

```
class TabBarExample extends StatelessWidget {
  const TabBarExample({super.key});
  @override
  Widget build(BuildContext context) {
    return DefaultTabController(
      length: 3,
      child: Scaffold(
        appBar: AppBar(
          title: const Text('TabBar Example'),
          bottom: const TabBar(
            tabs: [
              Tab(icon: Icon(Icons.directions_car), text: 'Car'),
              Tab(icon: Icon(Icons.directions_transit), text: 'Transit'),
              Tab(icon: Icon(Icons.directions_bike), text: 'Bike'),
            ],
          ),
        ),
        body: const TabBarView(
          children: [
            Center(child: Text('Car Tab')),
            Center(child: Text('Transit Tab')),
            Center(child: Text('Bike Tab')),
          ],
        ),
      ),
    );
  }
```

```
}
```

สรุป

หัวข้อ	คำอธิบาย	ตัวอย่างที่ใช้
Navigator.push/pop	เปิดหน้าใหม่/ย้อนกลับแบบธรรมดา	ตัวอย่างแรก
Named Routes	กำหนดชื่อหน้าใน routes map	ตัวอย่างที่สอง
ส่งข้อมูลระหว่างหน้า	ส่งข้อมูลผ่าน constructor และ await ผลลัพธ์	ตัวอย่างที่สาม
BottomNavigationBar	เมนูด้านล่าง สลับหน้าหลัก	ตัวอย่าง BottomNavigationBar
TabBar/TabBarView	แท็บบน AppBar เปลี่ยนเนื้อหา	ตัวอย่าง TabBar

---

Navigation และ Routing — รายละเอียดเชิงลึก

---

1. การเปลี่ยนหน้าโดยใช้ Navigator.push/pop

หลักการทำงานของ Navigator Stack

- Navigator คือ Stack ของหน้าจอ (Route Stack) ที่ทำงานเหมือน Stack แบบ LIFO (Last In, First Out)
- Navigator.push() คือการนำหน้าใหม่ (Route) ขึ้นไปบน Stack เพื่อแสดงหน้าใหม่
- Navigator.pop() คือการนำหน้าปัจจุบันออกจาก Stack เพื่อย้อนกลับไปหน้าเดิม
- ทุกหน้าที่เราสร้างขึ้นใน Flutter จะเป็น Route หนึ่งใน Stack นี้

Route คืออะไร

- Route คือ abstraction ของหน้าในแอป
- ปกติเราใช้ MaterialPageRoute ที่เป็น Route แบบ Material Design
- หรือสร้าง Custom Route เพื่อกำหนด animation เองได้

ทำไมต้องใช้ Navigator

- Flutter ไม่มีการจัดการหน้าจอแบบอัตโนมัติ ต้องใช้ Navigator เพื่อควบคุมการเปลี่ยนหน้าและประวัติการนำทาง (navigation history)
- Navigator จัดการ Back Stack ให้เหมือนระบบปฏิบัติการ ทำให้ระบบ Back Button ทำงานได้ถูกต้อง

การใช้งาน push และ pop

```
Navigator.push(context, MaterialPageRoute(builder: (context) => NewScreen()));
// เปิดหน้า NewScreen
Navigator.pop(context);
```

// ปิดหน้าปัจจุบัน ย้อนกลับหน้าเดิม

การใช้งาน pop พร้อมส่งค่ากลับ

Navigator.pop(context, result);

ค่าที่ส่งกลับนี้จะถูก await ใน push

---

2. Named Routes

ทำไมต้องใช้ Named Routes

- ช่วยให้การจัดการ route ง่ายขึ้น โดยเฉพาะเมื่อแอปมีหลายหน้า
- แทนที่จะสร้าง Route ด้วย Widget ตรง ๆ ใช้ String name แทน
- แยกการกำหนด Route ออกจาก UI code

การกำหนด Named Routes ใน MaterialApp

```
MaterialApp(
  initialRoute: '/',
  routes: {
    '/': (context) => HomeScreen(),
    '/details': (context) => DetailScreen(),
  },
);
```

การเรียกใช้งาน Named Route

```
Navigator.pushNamed(context, '/details');
```

การส่งข้อมูลผ่าน Named Routes

- ใช้ arguments property ใน pushNamed

```
Navigator.pushNamed(context, '/details', arguments: someData);
```

- รับข้อมูลในหน้าใหม่โดย

```
final args = ModalRoute.of(context)!.settings.arguments;
```

---

3. การส่งข้อมูลระหว่างหน้า

ส่งข้อมูลแบบ Forward (ส่งไปหน้าใหม่)

- ส่งผ่าน Constructor ของ Widget หน้าใหม่ตรง ๆ (แบบ push)

```
Navigator.push(context, MaterialPageRoute(
  builder: (context) => DetailPage(data: myData),
));
```

ส่งข้อมูลแบบ Backward (ส่งกลับผลลัพธ์)

- รอผลลัพธ์จากการ pop หน้า

final result = await Navigator.push(...);

- ในหน้าปลายทางส่งค่ากลับ

Navigator.pop(context, returnValue);

ตัวอย่างกรณีใช้การส่งกลับ เช่น เลือกข้อมูลหรือยืนยันคำสั่ง

---

4. BottomNavigationBar และ TabBar

BottomNavigationBar

- ใช้สร้างเมนูด้านล่างที่มีไอคอนและข้อความ ให้ผู้ใช้เลือกสลับหน้าหลักในแอป
- ควรรักษาสถานะของแต่ละหน้าไว้ (เช่นข้อมูลในแต่ละแท็บ)
- เทคนิคทั่วไปคือใช้ IndexedStack เพื่อเก็บหน้าหลายหน้าในตัวเดียว โดยแสดงเฉพาะหน้าที่เลือก

โครงสร้างการใช้งาน

- ตัวแปรเก็บ index ปัจจุบันของแท็บ
- เมธอดเปลี่ยน index เมื่อผู้ใช้กดแท็บ
- แสดง widget ของแต่ละหน้าโดยใช้ index

ตัวอย่างเชิงลึก

```
IndexedStack(
  index: _selectedIndex,
  children: [
    HomePage(),
    SearchPage(),
    ProfilePage(),
  ],
)
```

- ทำให้หน้าอื่น ๆ ถูกเก็บสถานะไว้ ไม่ถูกสร้างใหม่ทุกครั้งที่เปลี่ยนแท็บ

TabBar และ TabBarView

- ใช้สร้างแท็บ (tabs) ที่อยู่ใน AppBar ด้านบน พร้อมสลับหน้าจอเนื้อหาด้านล่าง
- ต้องใช้ DefaultTabController เพื่อจัดการ index ของแท็บ
- สามารถปรับแต่งได้ทั้งแท็บที่เป็นไอคอน, ข้อความ หรือทั้งสองอย่าง

การใช้งาน

```
DefaultTabController(
  length: 3,
  child: Scaffold(
```

```
    appBar: AppBar(
      bottom: TabBar(
        tabs: [Tab(text: 'Tab 1'), Tab(text: 'Tab 2'), Tab(text: 'Tab 3')],
      ),
    ),
    body: TabBarView(
      children: [Widget1(), Widget2(), Widget3()],
    ),
  ),
);
```

การจัดการสถานะ

- แต่ละแท็บจะสร้าง Widget ครั้งแรกที่ถูกเปิด
- สามารถใช้ AutomaticKeepAliveClientMixin เพื่อเก็บสถานะ widget ในแต่ละแท็บได้

แนวทางและ Best Practices

ประเด็น	คำแนะนำ
ควรใช้ Navigator แบบไหน?	สำหรับแอปขนาดเล็ก ใช้ push/pop ง่าย ๆ พอแอปขนาดใหญ่ควรใช้ Named Routes หรือ Navigation Libraries (เช่น go_router)
ส่งข้อมูลระหว่างหน้า	ส่งผ่าน constructor หรือ arguments ของ pushNamedถ้าค่าที่ส่งมีขนาดใหญ่หรือซับซ้อน แนะนำใช้ State Management
เก็บสถานะใน BottomNavigationBar	ใช้ IndexedStack เพื่อรักษาสถานะหน้าต่าง ๆ
แท็บควรใช้ TabBar หรือ BottomNavigationBar?	แท็บเหมาะกับกลุ่มเนื้อหาที่เกี่ยวข้องบนหน้าจอเดียว BottomNavigationBar เหมาะกับเมนูหลักของแอปที่สลับหน้าหลัก
ใช้ Navigator อย่างไรให้รองรับ Back Button	Navigator ของ Flutter เชื่อมกับ Back Button ของ Android/iOS โดยอัตโนมัติ แต่ต้องใช้ pop() ให้ถูกต้อง

การเปลี่ยนหน้าโดยใช้ Navigator.push/pop ใน Flutter

แนวคิดหลัก

Flutter ใช้ **Navigator** เพื่อจัดการการนำทาง (navigation) ระหว่างหน้าจอ (pages/screens) หรือที่เรียกว่า **Routes** โดย Navigator จะทำงานเหมือน **stack** ที่เก็บ route ที่เปิดอยู่ โดยเราจะ:

- **push()** — ใส่หน้าจอใหม่ลงบน stack (เปิดหน้าใหม่)
- **pop()** — นำหน้าจอปัจจุบันออกจาก stack (ย้อนกลับไปหน้าเดิม)

Navigator Stack

- หน้าจอที่แสดงอยู่คือ route ที่อยู่บนสุดของ stack
- กดปุ่มย้อนกลับ (Back) บนอุปกรณ์ จะทำการ pop หน้าออกไป

การใช้งาน

เปิดหน้าใหม่ด้วย Navigator.push()

```
Navigator.push(
  context,
  MaterialPageRoute(builder: (context) => NewPage()),
);
```

- MaterialPageRoute เป็น route ที่สร้าง animation แบบ Material Design
- builder จะคืน widget ของหน้าใหม่

ย้อนกลับด้วย Navigator.pop()

```
Navigator.pop(context);
```

- จะนำหน้าจอปัจจุบันออกจาก stack
- Flutter จะแสดงหน้าจอที่อยู่ล่างสุด (หน้าก่อนหน้า)

ส่งค่ากลับเมื่อ pop

เราสามารถส่งค่ากลับไปยังหน้าก่อนหน้าได้ผ่าน pop(value)

ตัวอย่าง

```
// ในหน้าปลายทาง
Navigator.pop(context, 'ผลลัพธ์');

// ในหน้าต้นทาง
final result = await Navigator.push(
  context,
  MaterialPageRoute(builder: (context) => NewPage()),
);
print('ได้ผลลัพธ์: $result');
```

ตัวอย่างเต็ม

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: const FirstPage());
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('First Page')),
      body: Center(
        child: ElevatedButton(
          child: const Text('ไปหน้าสอง'),
          onPressed: () async {
            // เปิดหน้าสองและรอผลลัพธ์กลับ
            final result = await Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => const SecondPage()),
            );
            // แสดงผลลัพธ์ที่ได้จากหน้าสอง
            ScaffoldMessenger.of(context).showSnackBar(
              SnackBar(content: Text('ผลลัพธ์: $result')),
            );
          },
```

```
        ),
      ),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Second Page')),
      body: Center(
        child: ElevatedButton(
          child: const Text('ย้อนกลับพร้อมส่งผลลัพธ์'),
          onPressed: () {
            Navigator.pop(context, 'Hello from Second Page!');
          },
        ),
      ),
    );
  }
}
```

สรุป

ฟังก์ชัน	การทำงาน
Navigator.push()	เปิดหน้าจอใหม่ ใส่ route ใหม่ใน stack
Navigator.pop()	ปิดหน้าปัจจุบัน ลบ route ออกจาก stack
Navigator.pop(value)	ปิดหน้าจอพร้อมส่งค่ากลับ

ตัวอย่างโปรแกรม **การเปลี่ยนหน้าโดยใช้ Navigator.push/pop** ให้ทั้งหมด 6 ตัวอย่าง แบ่งเป็น

- 3 ตัวอย่าง **พื้นฐาน**

- 3 ตัวอย่าง **แนวประยุกต์**

พร้อมโครงสร้างไฟล์ + คำอธิบายโค้ด + ผลการรันทุกตัวครับ

ตัวอย่างพื้นฐาน (3 โปรแกรม)

ตัวอย่างที่ 1: เปิดหน้าใหม่แบบง่าย + กลับหน้าเดิม

โครงสร้างไฟล์

```
lib/
 └── main.dart
```

main.dart

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: const FirstPage());
  }
}

class FirstPage extends StatelessWidget {
  const FirstPage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('First Page')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go to Second Page'),
          onPressed: () {
            Navigator.push(
              context,
```

```
              MaterialPageRoute(builder: (context) => const SecondPage()),
            );
          },
        ),
      ),
    );
  }
}

class SecondPage extends StatelessWidget {
  const SecondPage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Second Page')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go Back'),
          onPressed: () {
            Navigator.pop(context);
          },
        ),
      ),
    );
  }
}
```

คำอธิบาย

- หน้าแรกมีปุ่มกดเปิดหน้า 2 ด้วย Navigator.push()
- หน้า 2 มีปุ่มย้อนกลับด้วย Navigator.pop()
- หน้าจอเปลี่ยนพร้อม animation slide จากขวาไปซ้าย

ผลการรัน

- กดปุ่ม "Go to Second Page" จะเปลี่ยนไปหน้าใหม่

- กด "Go Back" จะย้อนกลับหน้าแรก

ตัวอย่างที่ 2: ส่งข้อมูลแบบง่าย ๆ ขณะ push

main.dart

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: const HomePage());
  }
}

class HomePage extends StatelessWidget {
  const HomePage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Home Page')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go to Detail with Message'),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => DetailPage(message: 'Hello from HomePage!'),
              ),
            );
          },
        ),
```

```
      ),
    );
  }
}

class DetailPage extends StatelessWidget {
  final String message;
  const DetailPage({super.key, required this.message});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Detail Page')),
      body: Center(child: Text(message, style: const TextStyle(fontSize: 24))),
    );
  }
}
```

คำอธิบาย

- ส่งข้อมูล String ผ่าน constructor ไปแสดงที่หน้าปลายทาง
- ใช้ required เพื่อบังคับส่งข้อมูล

ผลการรัน

- กดปุ่ม "Go to Detail with Message" จะเปิดหน้าใหม่และแสดงข้อความ "Hello from HomePage!"

ตัวอย่างที่ 3: รับค่ากลับเมื่อ pop

main.dart

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
```

```
  Widget build(BuildContext context) {
    return MaterialApp(home: const SelectPage());
  }
}

class SelectPage extends StatelessWidget {
  const SelectPage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Select a Color')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Choose Color'),
          onPressed: () async {
            final selectedColor = await Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => const ColorPickerPage()),
            );
            ScaffoldMessenger.of(context).showSnackBar(
              SnackBar(content: Text('You selected: $selectedColor')),
            );
          },
        ),
      ),
    );
  }
}

class ColorPickerPage extends StatelessWidget {
  const ColorPickerPage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
      appBar: AppBar(title: const Text('Pick a Color')),
      body: ListView(
        children: [
          ListTile(
            title: const Text('Red'),
            onTap: () => Navigator.pop(context, 'Red'),
          ),
          ListTile(
            title: const Text('Green'),
            onTap: () => Navigator.pop(context, 'Green'),
          ),
          ListTile(
            title: const Text('Blue'),
            onTap: () => Navigator.pop(context, 'Blue'),
          ),
        ],
      ),
    );
  }
}
```

คำอธิบาย

- หน้าแรกมีปุ่มกดเปิดหน้าเลือกสี
- หน้าเลือกสีมีรายการสี 3 ตัวเลือก
- เมื่อเลือกสีจะ pop พร้อมส่งค่ากลับ
- หน้าแรกรอรับค่าผ่าน await แล้วแสดง SnackBar

ผลการรัน

- กดปุ่ม "Choose Color" เปิดหน้าสี
- เลือกสีใดสีหนึ่ง จะกลับมาหน้าแรกพร้อมแสดงข้อความสีที่เลือก

ตัวอย่างแนวประยุกต์ (3 โปรแกรม)

ตัวอย่างที่ 4: เปิดหน้าใหม่แบบ fullscreenDialog พร้อม animation แบบ modal

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: const MainPage());
  }
}

class MainPage extends StatelessWidget {
  const MainPage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Main Page')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Open Modal Page'),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                fullscreenDialog: true, // เปิดหน้าแบบ modal dialog
                builder: (context) => const ModalPage(),
              ),
            );
          },
        ),
      ),
    );
  }
```

```
}

class ModalPage extends StatelessWidget {
  const ModalPage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Modal Page')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Close Modal'),
          onPressed: () => Navigator.pop(context),
        ),
      ),
    );
  }
}
```

ตัวอย่างที่ 5: เปิดหน้าใหม่ด้วยการส่งข้อมูลหลายค่าผ่าน Model Class

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

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

class Person {
  final String name;
  final int age;

  Person(this.name, this.age);
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
```

```
    return MaterialApp(home: const HomeScreen());
  }
}

class HomeScreen extends StatelessWidget {
  const HomeScreen({super.key});
  @override
  Widget build(BuildContext context) {
    final person = Person('John Doe', 28);
    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Show Person Details'),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => DetailScreen(person: person),
              ),
            );
          },
        ),
      ),
    );
  }
}

class DetailScreen extends StatelessWidget {
  final Person person;
  const DetailScreen({super.key, required this.person});

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

```
    return Scaffold(
      appBar: AppBar(title: const Text('Detail')),
      body: Center(
        child: Text(
          '${person.name}, Age: ${person.age}',
          style: const TextStyle(fontSize: 24),
        ),
      ),
    );
  }
}
```

ตัวอย่างที่ 6: เปิดหน้าใหม่ และใช้ WillPopScope ควบคุมปุ่ม Back กรณีต้องการยืนยันก่อนปิด

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: const ConfirmBackPage());
  }
}

class ConfirmBackPage extends StatefulWidget {
  const ConfirmBackPage({super.key});
  @override
  State<ConfirmBackPage> createState() => _ConfirmBackPageState();
}

class _ConfirmBackPageState extends State<ConfirmBackPage> {
  Future<bool> _onWillPop() async {
    final shouldLeave = await showDialog<bool>(
```

```
      context: context,
      builder: (context) => AlertDialog(
        title: const Text('Confirm Exit'),
        content: const Text('Do you want to leave this page?'),
        actions: [
          TextButton(onPressed: () => Navigator.pop(context, false), child: const Text('No')),
          TextButton(onPressed: () => Navigator.pop(context, true), child: const Text('Yes')),
        ],
      ),
    );
    return shouldLeave ?? false;
  }

  @override
  Widget build(BuildContext context) {
    return WillPopScope(
      onWillPop: _onWillPop, // ควบคุมการกด Back button
      child: Scaffold(
        appBar: AppBar(title: const Text('Confirm Back')),
        body: Center(child: const Text('Press Back button to test confirmation')),
      ),
    );
  }
}
```

สรุป

หมายเลข	หัวข้อ	คำอธิบาย
1	เปิดหน้าใหม่และกลับด้วย push/pop	หน้าแรกกดปุ่มเปิดหน้า 2, หน้า 2 กดกลับ
2	ส่งข้อมูลผ่าน constructor	ส่งข้อความจากหน้าแรกไปหน้าใหม่
3	รับค่ากลับจากหน้าใหม่	เลือกสี ส่งค่ากลับหน้าแรกแสดง SnackBar
4	เปิดหน้าแบบ modal fullscreenDialog	หน้าป๊อปอัปแบบ modal dialog
5	ส่งข้อมูลแบบ Object Model	ส่ง Person object ไปหน้าใหม่
6	ควบคุม Back Button ยืนยันก่อนออก	กด Back แล้วแสดงกล่องยืนยัน

Named Routes ใน Flutter

1. ความหมายและข้อดีของ Named Routes

- Named Routes คือการกำหนดชื่อเส้นทาง (route names) ให้กับแต่ละหน้าในแอป
- ใช้แทนการสร้าง Route ด้วย Widget ตรง ๆ (MaterialPageRoute builder)
- ข้อดีคือจัดการเส้นทางได้ง่าย, อ่านโค้ดสะดวก และเหมาะกับแอปที่มีหลายหน้า
- แยกส่วนการกำหนดเส้นทางจาก UI ทำให้โค้ดสะอาดและ maintain ง่ายขึ้น

2. การกำหนด Named Routes

ใน MaterialApp จะมีพารามิเตอร์ routes เป็น Map<String, WidgetBuilder>

```
MaterialApp(
  initialRoute: '/',
  routes: {
    '/': (context) => const HomePage(),
    '/about': (context) => const AboutPage(),
    '/settings': (context) => const SettingsPage(),
  },
);
```

- initialRoute คือ route เริ่มต้นที่แอปจะแสดง
- Key ของ Map คือชื่อ route แบบ String
- Value คือฟังก์ชันสร้าง Widget สำหรับ route นั้น ๆ

3. การนำทางด้วย Named Routes

เปิดหน้าใหม่โดยใช้ชื่อ route

```
Navigator.pushNamed(context, '/about');
```

ย้อนกลับ

```
Navigator.pop(context);
```

4. การส่งข้อมูลผ่าน Named Routes

สามารถส่งข้อมูลผ่านพารามิเตอร์ arguments ของ pushNamed

```
Navigator.pushNamed(
  context,
  '/details',
```

```
  arguments: 'Hello from Home',
);
```

รับข้อมูลในหน้าใหม่ด้วย

```
final args = ModalRoute.of(context)!.settings.arguments as String;
```

---

5. การใช้ onGenerateRoute (สำหรับกรณีซับซ้อน)

- routes map เหมาะกับกรณี route ตายตัว
- ถ้าอยากสร้าง route แบบ dynamic หรือจัดการ route ไม่เจาะจงชื่อ สามารถใช้ onGenerateRoute

ตัวอย่าง

```
MaterialApp(
  onGenerateRoute: (settings) {
    if (settings.name == '/profile') {
      final userId = settings.arguments as int;
      return MaterialPageRoute(builder: (_) => ProfilePage(userId: userId));
    }
    // route อื่น ๆ
    return null;
  },
);
```

ตัวอย่างโค้ดเต็ม

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
```

```
        '/': (context) => const HomePage(),
        '/about': (context) => const AboutPage(),
        '/details': (context) => const DetailsPage(),
      },
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              onPressed: () {
                Navigator.pushNamed(context, '/about');
              },
              child: const Text('Go to About'),
            ),
            ElevatedButton(
              onPressed: () {
                Navigator.pushNamed(context, '/details', arguments: 'Hello from Home!');
              },
              child: const Text('Go to Details with Data'),
            ),
          ],
        ),
      ),
```

```
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('About')),
      body: const Center(child: Text('About Page')),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    final args = ModalRoute.of(context)!.settings.arguments as String?;

    return Scaffold(
      appBar: AppBar(title: const Text('Details')),
      body: Center(
        child: Text(args ?? 'No Data', style: const TextStyle(fontSize: 24)),
      ),
    );
  }
}
```

สรุปข้อดีของ Named Routes

ข้อดี	คำอธิบาย

ข้อดี	คำอธิบาย
แยกเส้นทางออกจาก UI	ทำให้โค้ดอ่านง่ายและ maintain ได้ง่าย
ใช้ชื่อกำหนด route	ลดการใช้ widget builder ตรง ๆ ซ้ำ ๆ
ส่งข้อมูลง่าย	ผ่าน arguments ของ pushNamed
เหมาะกับแอปใหญ่	มีหลายหน้าหลาย route จัดการง่าย

ตัวอย่างโปรแกรม **Named Routes** ให้ทั้งหมด 6 ตัวอย่าง แบ่งเป็น

- 3 ตัวอย่าง **พื้นฐาน**
- 3 ตัวอย่าง **แนวประยุกต์**

พร้อมโครงสร้างไฟล์ + คำอธิบายโค้ด + ผลการรัน

ตัวอย่างพื้นฐาน (3 โปรแกรม)

ตัวอย่างที่ 1: ใช้ Named Routes แบบง่าย ๆ

โครงสร้างไฟล์

```
lib/
 └── main.dart
```

main.dart

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => const HomePage(),
        '/about': (context) => const AboutPage(),
      },
    );
```

```
  }
}

class HomePage extends StatelessWidget {
  const HomePage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pushNamed(context, '/about');
          },
          child: const Text('Go to About Page'),
        ),
      ),
    );
  }
}

class AboutPage extends StatelessWidget {
  const AboutPage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('About')),
      body: const Center(child: Text('This is the About Page')),
    );
  }
}
```

คำอธิบาย

- กำหนด named routes / และ /about ใน routes map

- เริ่มแอปที่ route '/' คือ HomePage
- กดปุ่มเปิดหน้า About ผ่าน Navigator.pushNamed(context, '/about')

ผลการรัน

- หน้าแรกมีปุ่มกด “Go to About Page”
- เมื่อกดจะเปลี่ยนไปหน้า About และแสดงข้อความ

ตัวอย่างที่ 2: ส่งข้อมูลผ่าน Named Routes

main.dart

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => const HomePage(),
        '/details': (context) => const DetailsPage(),
      },
    );
  }
}

class HomePage extends StatelessWidget {
  const HomePage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
```

```
        child: ElevatedButton(
          onPressed: () {
            Navigator.pushNamed(
              context,
              '/details',
              arguments: 'Hello from HomePage!',
            );
          },
          child: const Text('Go to Details with Data'),
        ),
      ),
    );
  }
}

class DetailsPage extends StatelessWidget {
  const DetailsPage({super.key});
  @override
  Widget build(BuildContext context) {
    final args = ModalRoute.of(context)!.settings.arguments as String?;
    return Scaffold(
      appBar: AppBar(title: const Text('Details')),
      body: Center(
        child: Text(args ?? 'No data received', style: const TextStyle(fontSize: 24)),
      ),
    );
  }
}
```

คำอธิบาย

- ส่งข้อมูล String ผ่าน arguments ของ pushNamed
- รับข้อมูลในหน้าปลายทางด้วย ModalRoute.of(context)!.settings.arguments

---

ผลการรัน

- หน้าแรกมีปุ่ม “Go to Details with Data”
- กดแล้วจะเปิดหน้ารายละเอียดและแสดงข้อความที่ส่งมา

ตัวอย่างที่ 3: ใช้ onGenerateRoute เพื่อจัดการ Named Routes แบบไดนามิก

main.dart

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      onGenerateRoute: (settings) {
        if (settings.name == '/') {
          return MaterialPageRoute(builder: (context) => const HomePage());
        } else if (settings.name == '/profile') {
          final userId = settings.arguments as int;
          return MaterialPageRoute(
            builder: (context) => ProfilePage(userId: userId),
          );
        }
        return null;
      },
    );
  }
}

class HomePage extends StatelessWidget {
  const HomePage({super.key});
  @override
  Widget build(BuildContext context) {
```

```
    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pushNamed(
              context,
              '/profile',
              arguments: 123,
            );
          },
          child: const Text('Go to Profile'),
        ),
      ),
    );
  }
}

class ProfilePage extends StatelessWidget {
  final int userId;
  const ProfilePage({super.key, required this.userId});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Profile #$userId')),
      body: Center(
        child: Text('User ID: $userId', style: const TextStyle(fontSize: 24)),
      ),
    );
  }
}
```

คำอธิบาย

- ใช้ onGenerateRoute เพื่อจัดการ route แบบไดนามิก
- รับ arguments และส่งไปหน้าปลายทางที่ต้องการข้อมูล
- เหมาะกับแอปที่มี route ซับซ้อน หรือหลายพารามิเตอร์

ผลการรัน

- หน้าแรกมีปุ่ม “Go to Profile”
- กดแล้วเปิดหน้ารายละเอียดโปรไฟล์พร้อมแสดง User ID ที่ส่งมา

ตัวอย่างแนวประยุกต์ (3 โปรแกรม)

ตัวอย่างที่ 4: Named Routes + ส่ง Object เป็น argument

main.dart

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

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

class User {
  final String name;
  final int age;

  User(this.name, this.age);
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => const HomePage(),
        '/userDetails': (context) => const UserDetailsPage(),
      },
    );
```

```
  }
}

class HomePage extends StatelessWidget {
  const HomePage({super.key});
  @override
  Widget build(BuildContext context) {
    final user = User('Alice', 30);
    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pushNamed(
              context,
              '/userDetails',
              arguments: user,
            );
          },
          child: const Text('Go to User Details'),
        ),
      ),
    );
  }
}

class UserDetailsPage extends StatelessWidget {
  const UserDetailsPage({super.key});
  @override
  Widget build(BuildContext context) {
    final user = ModalRoute.of(context)!.settings.arguments as User?;
    return Scaffold(
      appBar: AppBar(title: Text(user?.name ?? 'No User')),
      body: Center(
```

```
        child: Text(
          user != null ? 'Name: ${user.name}, Age: ${user.age}' : 'No data',
          style: const TextStyle(fontSize: 24),
        ),
      ),
    );
  }
}
```

ตัวอย่างที่ 5: Named Routes + onGenerateRoute + Route Guard

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

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

bool isLoggedIn = false; // สมมติสถานะล็อกอิน

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      onGenerateRoute: (settings) {
        if (settings.name == '/') {
          return MaterialPageRoute(builder: (_) => const HomePage());
        } else if (settings.name == '/profile') {
          if (!isLoggedIn) {
            // ถ้าไม่ได้ล็อกอิน ไปหน้า login แทน
            return MaterialPageRoute(builder: (_) => const LoginPage());
          }
          return MaterialPageRoute(builder: (_) => const ProfilePage());
        }
        return null;
      },
```

```
    );
  }
}

class HomePage extends StatelessWidget {
  const HomePage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pushNamed(context, '/profile');
          },
          child: const Text('Go to Profile'),
        ),
      ),
    );
  }
}

class LoginPage extends StatelessWidget {
  const LoginPage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Login')),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            isLoggedIn = true;
            Navigator.pop(context);
          },
```

```
            child: const Text('Log In'),
          ),
        ),
      );
  }
}

class ProfilePage extends StatelessWidget {
  const ProfilePage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Profile')),
      body: const Center(child: Text('Welcome to your profile!')),
    );
  }
}
```

ตัวอย่างที่ 6: Named Routes + การส่งข้อมูลแบบหลายพารามิเตอร์ผ่าน Map

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => const HomePage(),
        '/product': (context) => const ProductPage(),
      },
    );
```

```
  }
}

class HomePage extends StatelessWidget {
  const HomePage({super.key});
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
        child: ElevatedButton(
          onPressed: () {
            Navigator.pushNamed(
              context,
              '/product',
              arguments: {
                'id': 101,
                'name': 'Laptop',
                'price': 25999.99,
              },
            );
          },
          child: const Text('View Product'),
        ),
      ),
    );
  }
}

class ProductPage extends StatelessWidget {
  const ProductPage({super.key});
  @override
  Widget build(BuildContext context) {
    final args = ModalRoute.of(context)!.settings.arguments as Map<String, dynamic>?;
```

```
    return Scaffold(
      appBar: AppBar(title: Text(args?['name'] ?? 'No Product')),
      body: Center(
        child: Text(
          args != null
              ? 'Product ID: ${args['id']}\nPrice: \$${args['price']}'
              : 'No product data',
          style: const TextStyle(fontSize: 24),
          textAlign: TextAlign.center,
        ),
      ),
    );
  }
}
```

สรุป

หมายเลข	หัวข้อ	คำอธิบาย
1	Named Routes แบบง่าย	กำหนด route ใน routes map และเรียกใช้ชื่อ route
2	ส่งข้อมูลผ่าน arguments	ส่งข้อมูล String ผ่าน pushNamed
3	ใช้ onGenerateRoute	จัดการ route แบบไดนามิกและรับ arguments
4	ส่ง Object ผ่าน Named Routes	ส่ง Object class เป็น argument
5	Route Guard	ตรวจสอบสถานะก่อนเข้า route (ล็อกอิน)
6	ส่งข้อมูลหลายพารามิเตอร์	ส่ง Map ผ่าน arguments

ส่งข้อมูลระหว่างหน้า (Passing Data Between Screens)

---

1. แนวคิด

- ใน Flutter การเปลี่ยนหน้าเกิดจาก Navigator.push หรือ Navigator.pushNamed
- เราสามารถส่งข้อมูลจากหน้าหนึ่งไปยังอีกหน้าหนึ่งผ่านพารามิเตอร์ของ push หรือ arguments ใน pushNamed

- หน้าปลายทางจะดึงข้อมูลที่ส่งมาจากหน้าต้นทางมาใช้ได้ผ่าน ModalRoute.of(context)!.settings.arguments หรือผ่านพารามิเตอร์ constructor ของ Widget

---

2. วิธีส่งข้อมูลระหว่างหน้าแบบง่าย ๆ

วิธีที่ 1: ส่งข้อมูลผ่าน constructor ของ Widget (Explicit Route)

```
// ส่งข้อมูลผ่าน constructor ของหน้าใหม่
Navigator.push(
  context,
  MaterialPageRoute(
    builder: (context) => DetailPage(data: 'Hello from Home!'),
  ),
);
ใน DetailPage รับข้อมูลผ่าน constructor
class DetailPage extends StatelessWidget {
  final String data;
  const DetailPage({super.key, required this.data});
  // ใช้ data ใน build method
}
```

วิธีที่ 2: ส่งข้อมูลผ่าน Named Routes ด้วย arguments

```
Navigator.pushNamed(
  context,
  '/detail',
  arguments: 'Hello from Home!',
);
ในหน้าปลายทาง
final args = ModalRoute.of(context)!.settings.arguments as String?;
```

---

3. ตัวอย่างโปรแกรมเต็ม

โครงสร้างไฟล์

```
lib/
  └── main.dart
```

---

main.dart

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Passing Data Demo',
      initialRoute: '/',
      routes: {
        '/': (context) => const HomePage(),
        '/detail': (context) => const DetailPage(),
      },
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    String message = 'Hello from HomePage!';

    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
```

```
            child: const Text('Send Data via Constructor'),
            onPressed: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => DetailPageWithConstructor(data: message),
                ),
              );
            },
          ),
          const SizedBox(height: 20),
          ElevatedButton(
            child: const Text('Send Data via Named Route'),
            onPressed: () {
              Navigator.pushNamed(
                context,
                '/detail',
                arguments: message,
              );
            },
          ),
        ],
      ),
    ),
  );
 }
}

// วิธี 1: รับข้อมูลผ่าน constructor
class DetailPageWithConstructor extends StatelessWidget {
  final String data;
  const DetailPageWithConstructor({super.key, required this.data});

  @override
```

```
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Detail (Constructor)')),
      body: Center(
        child: Text(data, style: const TextStyle(fontSize: 24)),
      ),
    );
  }
}

// วิธี 2: รับข้อมูลผ่าน arguments ของ named route
class DetailPage extends StatelessWidget {
  const DetailPage({super.key});

  @override
  Widget build(BuildContext context) {
    final args = ModalRoute.of(context)!.settings.arguments as String?;

    return Scaffold(
      appBar: AppBar(title: const Text('Detail (Named Route)')),
      body: Center(
        child: Text(args ?? 'No data received', style: const TextStyle(fontSize: 24)),
      ),
    );
  }
}
```

---

4. คำอธิบาย

- หน้า HomePage มีปุ่ม 2 ปุ่ม
  - ปุ่มแรกส่งข้อมูลผ่าน constructor ของหน้าปลายทางโดยตรง
  - ปุ่มที่สองส่งข้อมูลผ่าน Named Route โดยใช้ arguments
- หน้า DetailPageWithConstructor รับข้อมูลผ่าน constructor
- หน้า DetailPage รับข้อมูลผ่าน ModalRoute.of(context)!.settings.arguments

---

5. ผลการรัน

- หน้าแรกมีปุ่ม "Send Data via Constructor" กับ "Send Data via Named Route"
- กดปุ่มใดจะเปิดหน้าใหม่ที่แสดงข้อความที่ส่งมาอย่างถูกต้อง

ตัวอย่างโปรแกรม **ส่งข้อมูลระหว่างหน้า (Passing Data Between Screens)** จำนวน 6 ตัวอย่าง แบ่งเป็น

- 3 ตัวอย่าง **พื้นฐาน**
- 3 ตัวอย่าง **แนวประยุกต์**

แต่ละตัวอย่างมีโครงสร้างไฟล์, โค้ดเต็ม, คำอธิบาย และผลการรัน

ตัวอย่างพื้นฐาน (3 โปรแกรม)

ตัวอย่างที่ 1: ส่งข้อมูลแบบง่ายผ่าน Constructor

โครงสร้างไฟล์

```
lib/
 └── main.dart
```

main.dart

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

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

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

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      home: HomePage(),
    );
  }
}

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

```
  @override
  Widget build(BuildContext context) {
    final String message = 'Hello from HomePage!';

    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go to Detail Page'),
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => DetailPage(data: message),
              ),
            );
          },
        ),
      ),
    );
  }
}

class DetailPage extends StatelessWidget {
  final String data;
  const DetailPage({super.key, required this.data});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Detail Page')),
      body: Center(
        child: Text(
```

```
            data,
            style: const TextStyle(fontSize: 24),
          ),
        ),
      );
  }
}
```

คำอธิบาย

- หน้า Home มีปุ่มกด
- เมื่อกดจะเปิดหน้า DetailPage พร้อมส่งข้อความผ่าน constructor
- DetailPage รับข้อมูลและแสดงผลบนหน้าจอ

ผลการรัน

- หน้าแรกมีปุ่ม "Go to Detail Page"
- กดแล้วไปหน้าใหม่ แสดงข้อความ "Hello from HomePage!"

ตัวอย่างที่ 2: ส่งข้อมูลผ่าน Named Routes และ arguments

main.dart

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => const HomePage(),
        '/detail': (context) => const DetailPage(),
      },
```

```
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    final String message = 'Hello from HomePage via Named Route!';

    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go to Detail Page'),
          onPressed: () {
            Navigator.pushNamed(
              context,
              '/detail',
              arguments: message,
            );
          },
        ),
      ),
    );
  }
}

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

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

```
    final args = ModalRoute.of(context)!.settings.arguments as String?;

    return Scaffold(
      appBar: AppBar(title: const Text('Detail Page')),
      body: Center(
        child: Text(
          args ?? 'No data received',
          style: const TextStyle(fontSize: 24),
        ),
      ),
    );
  }
}
```

คำอธิบาย

- กำหนด named routes / และ /detail
- ส่งข้อมูลผ่าน arguments ใน pushNamed
- ดึงข้อมูลในหน้าปลายทางผ่าน ModalRoute.of(context)!.settings.arguments

---

ผลการรัน

- หน้าแรกมีปุ่ม "Go to Detail Page"
- กดแล้วเปิดหน้า Detail แสดงข้อความที่ส่งมา

ตัวอย่างที่ 3: ส่ง Object ผ่าน Named Routes

main.dart

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

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

class User {
  final String name;
  final int age;

  User(this.name, this.age);
```

```
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => const HomePage(),
        '/userDetail': (context) => const UserDetailPage(),
      },
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    final user = User('Alice', 28);

    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
        child: ElevatedButton(
          child: const Text('Go to User Detail'),
          onPressed: () {
            Navigator.pushNamed(
              context,
              '/userDetail',
              arguments: user,
```

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

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

  @override
  Widget build(BuildContext context) {
    final user = ModalRoute.of(context)!.settings.arguments as User?;

    return Scaffold(
      appBar: AppBar(title: Text(user?.name ?? 'No User')),
      body: Center(
        child: Text(
          user != null ? 'Name: ${user.name}, Age: ${user.age}' : 'No data',
          style: const TextStyle(fontSize: 24),
        ),
      ),
    );
  }
}
```

คำอธิบาย

- สร้างคลาส User
- ส่ง Object User ผ่าน arguments ของ named route
- ดึงข้อมูล Object ในหน้าปลายทางและแสดงผล

ผลการรัน

- หน้าแรกมีปุ่ม "Go to User Detail"

- กดแล้วเปิดหน้าใหม่แสดงชื่อและอายุของ User

ตัวอย่างแนวประยุกต์ (3 โปรแกรม)

ตัวอย่างที่ 4: ส่งข้อมูลกลับจากหน้า Detail (Return Data)

main.dart

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

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

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

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(home: HomePage());
  }
}

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  String message = 'No message yet';

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
```

```
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(message, style: const TextStyle(fontSize: 24)),
            const SizedBox(height: 20),
            ElevatedButton(
              child: const Text('Go to Detail and Wait for Result'),
              onPressed: () async {
                final result = await Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => const DetailPage()),
                );
                if (result != null) {
                  setState(() {
                    message = result;
                  });
                }
              },
            ),
          ],
        ),
      ),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Detail')),
      body: Center(
```

```
        child: ElevatedButton(
          child: const Text('Send Data Back'),
          onPressed: () {
            Navigator.pop(context, 'Hello from Detail!');
          },
        ),
      ),
    );
  }
}
```

คำอธิบาย

- หน้า Detail ส่งข้อมูลกลับด้วย Navigator.pop(context, data)
- หน้า Home รอผลลัพธ์ด้วย await Navigator.push(...)
- เมื่อได้ผลลัพธ์มา อัปเดต UI ด้วย setState

ผลการรัน

- หน้าแรกแสดงข้อความเริ่มต้น
- กดปุ่มไปหน้า Detail แล้วกดปุ่มส่งข้อมูลกลับ
- ข้อความบนหน้า Home เปลี่ยนตามข้อมูลที่ส่งกลับมา

ตัวอย่างที่ 5: ส่งข้อมูลแบบ Map ผ่าน Named Routes และแสดงในหลาย Widgets

main.dart

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

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

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
```

```
      routes: {
        '/': (context) => const HomePage(),
        '/product': (context) => const ProductPage(),
      },
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    final productData = {
      'id': 1001,
      'name': 'Smartphone',
      'price': 15999.99,
      'description': 'A powerful smartphone with great camera.',
    };

    return Scaffold(
      appBar: AppBar(title: const Text('Home')),
      body: Center(
        child: ElevatedButton(
          child: const Text('View Product Details'),
          onPressed: () {
            Navigator.pushNamed(
              context,
              '/product',
              arguments: productData,
            );
          },
        ),
      ),
```