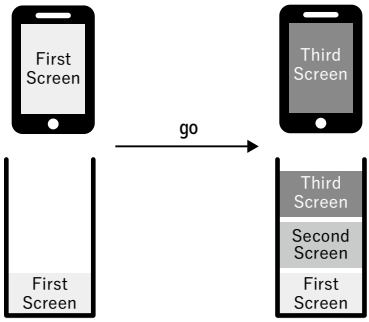
wassyoi：t--sekiguchi@hotmail.co.jp

る」ボタンをタップするとSecondScreen画面が表示されます。これはGoRouterクラスのgoメソッドがGoRouteクラスの入れ子構造をそのまま画面スタックに再現するためです。ThirdScreen画面のGoRouteはSecondScreen画面のGoRouteのroutesに追加されていました。そのため、❶を実行すると、画面スタックは**図5.9**のようになるのです。

図5.9　**goメソッドでのスタックの変化**



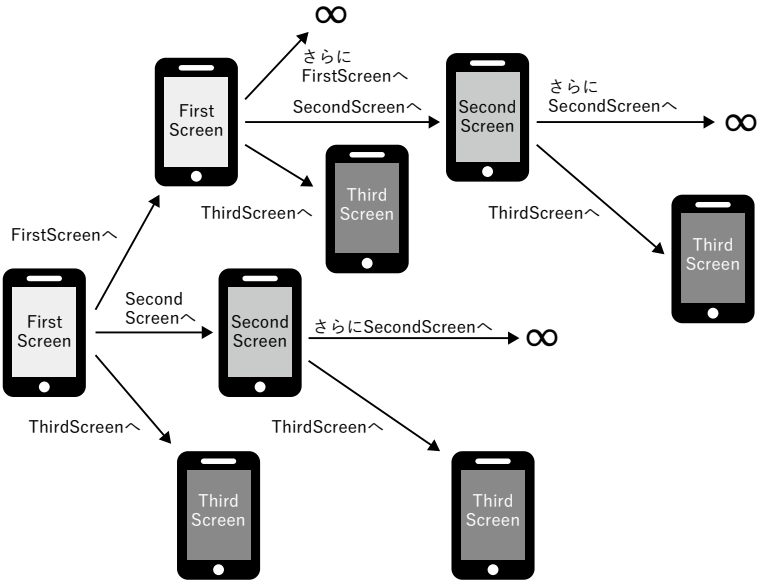
加えて、元の画面に戻る処理はGoRouterクラスのpopメソッドに書き換えました(❸、❹)。go_routerパッケージでは、Navigatorウィジェットを入れ子構造で利用する機能があり、このpopメソッドはそのようなケースを考慮して実装されています。Navigatorウィジェットを入れ子構造にするにはShellRoute[注4]クラスを利用します。今回のサンプルではShellRouteクラスを採用していないためGoRouterクラスのpopメソッド、NavigatorStateクラスのpopメソッド、どちらを使っても結果は同じになります。

goとpushの違い

先ほどの例ではGoRouterクラスのgoメソッドで画面遷移を行いましたが、画面遷移の要件がより複雑な**図5.10**のような場合を考えてみましょう。

注4　https://pub.dev/documentation/go_router/latest/go_router/ShellRoute-class.html

図5.10 **複雑な画面遷移の要件**



これを`GoRoute`クラスの入れ子構造で表現するのは困難です。このようなケースにも対応できるよう、go_routerパッケージは画面スタックにプッシュするメソッドも提供しています。この機能を使い、図5.10のような画面遷移を実現してみましょう。まずは、`FirstScreen`画面から修正します。

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('FirstScreen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              child: const Text('FirstからFirstへ'),
              onPressed: () {
                GoRouter.of(context).push('/');
              },
            ),
```

❶

wassyoi：t--sekiguchi@hotmail.co.jp


```
            ElevatedButton(
              child: const Text('FirstからSecondへ'),
              onPressed: () {
                // GoRouter.of(context).go('/second');
                GoRouter.of(context).push('/second'); ——❷
              },
            ),
            ElevatedButton(
              child: const Text('FirstからThirdへ'),
              onPressed: () {
                // GoRouter.of(context).go('/second/third');
                GoRouter.of(context).push('/second/third'); ——❸
              },
            ),
          ],
        ),
      ),
    );
  }
}
```

FirstScreen画面からFirstScreen画面へ遷移するボタンを追加しました(❶)。またSecondScreen画面、ThirdScreen画面への遷移処理をgoメソッドからpushメソッドに変更しました(❷、❸)。

続いて、SecondScreen画面を修正します。

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('SecondScreen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              child: const Text('SecondからSecondへ'),
              onPressed: () {
                GoRouter.of(context).push('/second');
              },
            ),
            ElevatedButton(
```

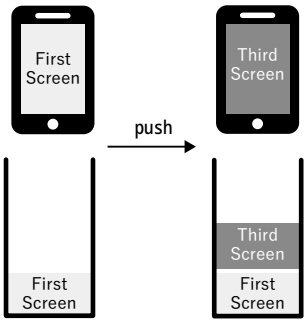
❶

```
              child: const Text('SecondからThirdへ'),
              onPressed: () {
                // GoRouter.of(context).go('/second/third');
                GoRouter.of(context).push('/second/third'); ❷
              },
            ),
            ElevatedButton(
              child: const Text('戻る'),
              onPressed: () {
                GoRouter.of(context).pop();
              },
            ),
          ],
        ),
      ),
    );
  }
}
```

SecondScreen画面からSecondScreen画面へ遷移するボタンを追加しました(❶)。ThirdScreen画面への遷移処理をgoメソッドからpushメソッドに変更しました(❷)。

それでは動作を確認してみましょう。おおむね期待どおりに動作しているように見えますが、FirstScreen画面からThirdScreen画面へ遷移した際に中間のSecondScreen画面がスタックに積まれなくなってしまいました(**図5.11**)。

図5.11 **pushメソッドでのスタックの変化**



GoRouterクラスのpushメソッドはNavigator 1.0のpushメソッドと同様に、1つのRouteクラスをスタックにプッシュすることしかできません。FirstScreen画面からThirdScreen画面への遷移処理は、GoRouterクラスの

wassyoi：t--sekiguchi@hotmail.co.jp

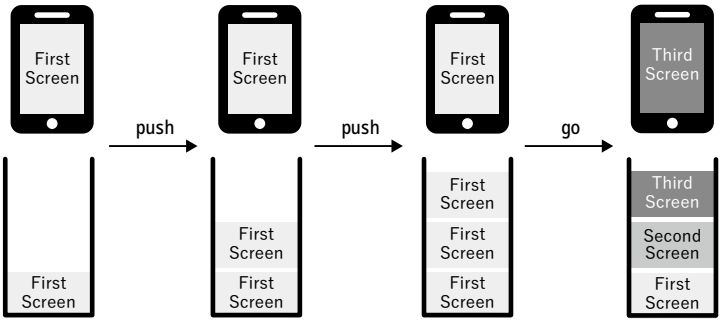
goメソッドに戻してみましょう。

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('FirstScreen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              child: const Text('FirstからFirstへ'),
              onPressed: () {
                GoRouter.of(context).push('/');
              },
            ),
            ElevatedButton(
              child: const Text('FirstからSecondへ'),
              onPressed: () {
                GoRouter.of(context).push('/second');
              },
            ),
            ElevatedButton(
              child: const Text('FirstからThirdへ'),
              onPressed: () {
                // GoRouter.of(context).push('/second/third');
                GoRouter.of(context).go('/second/third'); ——❶
              },
            ),
          ],
        ),
      ),
    );
  }
}
```

FirstScreen画面からThirdScreen画面への遷移処理をgoメソッドに戻しました(❶)。これで、ThirdScreen画面へ遷移し、「戻る」ボタンをタップするとSecondScreen画面が表示されるようになりました。しかし、**図5.12**のように遷移すると一部のFirstScreen画面がスタックから消えてしまいます。

図5.12 goメソッドでのスタックの変化



goメソッドはGoRouteの入れ子構造をそのまま画面スタックに再現するのでした。そのため、pushメソッドで複数のFirstScreen画面をスタックに積んだとしても、その後goメソッドで遷移するとスタックが書き換えられ、FirstScreen画面は1つしかスタックに残らないのです。

画面スタックをどのように変化させたいのかを意識して、適切なメソッドを選択しましょう。

5.3 まとめ

Flutterフレームワークが提供する「テーマ」と「画面遷移」に関する機能を紹介しました。

MaterialAppウィジェットやThemeDataクラスを使うことで、マテリアルデザインにのっとったアプリを簡単に作ることができます。ThemeDataクラスがテーマを自動計算してくれるので、テーマのカスタマイズやダークモード対応も容易です。

アプリ独自のテーマ、世界観を演出するなら、Theme Extensionを使うのがよいでしょう。

アプリの画面遷移の実装方法を学ぶために、Navigator 1.0とNavigator 2.0の違いを確認しました。シンプルな画面遷移であればNavigator 1.0のpushメソッド、popメソッドで対応できるケースも十分あるでしょう。複雑な画面遷移を実装する場合はNavigator 2.0が候補に挙がります。本章では、例としてgo_routerパッケージを使った画面遷移の実装を紹介しました。

第6章

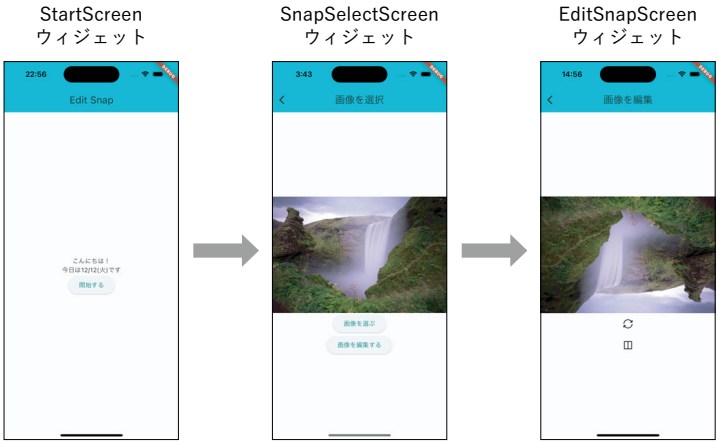
実践ハンズオン①

画像編集アプリを開発

本章ではFlutterアプリをハンズオン形式で実装していきます。簡単な画像編集アプリを作成する過程で、第4章や第5章で学んだ内容を実践します。本章のハンズオンではiOS Simulatorを使用します。あわせてもらえばすべての工程を体験できます。なお、本章でもfvmコマンドを省略してflutterコマンドを記載しています。ご自身の環境、コマンドを実行するディレクトリにあわせて読み替えてください。

図6.1が完成イメージです。

図6.1 **アプリの完成イメージ**



6.1 開発するアプリの概要

このハンズオンで実装するアプリの概要を説明します。スマートフォンの画像ライブラリから取得した画像を回転、反転させて編集するアプリです。画面は全部で3つあります。

スタート画面

アプリ起動後に表示される画面です(**図6.2**)。現在の日付が表示され、「開始する」ボタンをタップすると画像選択画面に遷移します。