ができません。また、ディープリンクで中間の画面を生成すると、Webアプリとして実行した際にブラウザの進む／戻るボタンの挙動が不自然になります。そのため、Flutterは名前付きルートを推奨しないとしています。

なお、実際にディープリンクとして動作させるにはネイティブの設定や、構成ファイルのホスティングが必要になります。

Routerウィジェットによる画面遷移 —— Navigator 2.0

続いてRouterウィジェットを使った画面遷移を見ていきましょう。いよいよNavigator 2.0の登場です。画面履歴を一度に書き換えるような挙動を確認することができます。Routerウィジェットを利用した実装は複雑になるためラップしたパッケージを使うのがよいでしょう。本書ではgo_routerパッケージを紹介します。先ほどの名前付きルートで画面遷移するサンプルをgo_routerパッケージを使って書き換えてみましょう。

パッケージを導入するためにプロジェクトのディレクトリで、ターミナルから以下のコマンドを実行してください。

```
# go_routerパッケージを導入
$ flutter pub add go_router
```

go_routerによる画面スタックの書き換えを体験する

次にMaterialAppウィジェットのコンストラクタを修正します。

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

void main() {
  runApp(
    MaterialApp.router( ——❶
      routerConfig: _router,
    ),
  );
}

final _router = GoRouter( ——❷
  routes: [
    GoRoute( ——❹
      path: '/',
      builder: (context, state) => const FirstScreen(), ——❸
    ),
```

```
    GoRoute(
      path: '/second',
      builder: (context, state) => const SecondScreen(),
    ),
    GoRoute(
      path: '/third',
      builder: (context, state) => const ThirdScreen(),
    ),
  ],
);
```

❸

MaterialAppウィジェットのrouterという名前付きコンストラクタを利用すると、内部でRouterウィジェットが生成されます(❶)。routerConfigパラメータは、Routerウィジェットを利用する際に必要な関連オブジェクトをバンドルして渡すことのできる便利なパラメータです。

続いて、go_routerパッケージを扱っていきましょう。似た名前のクラスが連続するので注意してください。GoRouterクラス(❷)はRouterConfigクラスのサブクラスで、routerコンストラクタ(❶)のrouterConfigパラメータに渡すことができます。GoRouteクラス(❹)は遷移先のパスやPageクラスの生成方法を保持するクラスです。

GoRouterクラス(❷)のroutesパラメータにリスト型で渡します(❸)。pathパラメータには遷移先のパスを、builderパラメータにはウィジェットを生成する関数型を渡します。

続いて、画面遷移の実装を修正します。

```
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Secondへ'),
              onPressed: () {
                // Navigator.of(context).pushNamed('/second');
```

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

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Thirdへ'),
              onPressed: () {
                // Navigator.of(context).pushNamed('/second/third');
                GoRouter.of(context).go('/third'); ——❸
              },
            ),
            ElevatedButton(
              child: const Text('戻る'),
              onPressed: () {
                Navigator.of(context).pop(); ——❹
              },
            ),
          ],
        ),
      ),
    );
```

```
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('ThirdScreen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              child: const Text('戻る'),
              onPressed: () {
                Navigator.of(context).pop(); ——❺
              },
            ),
          ],
        ),
      ),
    );
  }
}
```

❶〜❸の部分を修正しました。GoRouterクラスの静的メソッドofからインスタンスを取り出し、goメソッドを呼び出して画面遷移します。goメソッドの引数には遷移先のパスを渡します。さっそく、動作を確認してみましょう。

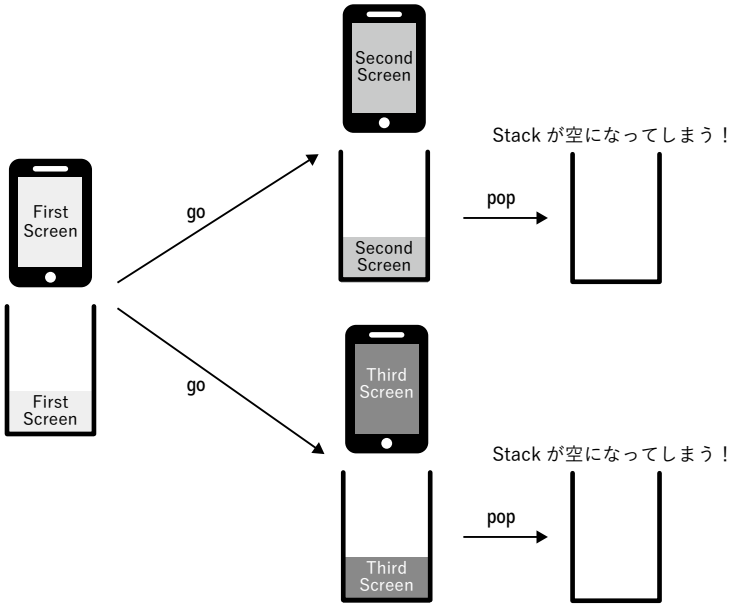
FirstScreen画面からSecondScreen画面へ、SecondScreen画面からThirdScreen画面へは問題なく遷移します。しかし、画面左上のバックボタンが表示されません。また、SecondScreen画面の「戻る」ボタン、ThirdScreen画面の「戻る」ボタンをタップするとアサーションエラーが発生します。

GoRouteで入れ子構造を作る

前項の動作はGoRouterクラスのgoメソッドが画面スタックに新しい画面をプッシュしているわけではなく、画面スタックを置き換えているためです。FirstScreen画面からSecondScreen画面へ遷移(❶)した際に画面スタックはFirstScreen画面のみの状態からSecondScreen画面のみへ書き換えられたの

です(**図5.8**)。

図5.8 **goメソッドでのスタックの変化**



よって、**NavigatorState**クラスの**pop**メソッド(❹、❺)を実行すると戻る画面が存在しないため、アサーションエラーが発生したのです。

この問題を解決するために、ルートの構成を変更します。

```
final _router = GoRouter(
  routes: [
    GoRoute(
      path: '/',
      builder: (context, state) => const FirstScreen(),
      routes: [
        GoRoute(
          path: 'second',
          builder: (context, state) => const SecondScreen(),
        ),
      ],
    ),
    // GoRoute(
    //   path: '/second',
    //   builder: (context, state) => const SecondScreen(),
    // ),
    GoRoute(
```

❶

```
      path: '/third',
      builder: (context, state) => const ThirdScreen(),
    ),
  ],
);
```

SecondScreen画面へのGoRouteを、FirstScreen画面のGoRouteのroutesパラメータに移動しました(❶)。このように、GoRouteは入れ子構造にすることができます。

同様に、ThirdScreen画面へのGoRouteもSecondScreen画面のGoRouteの入れ子にしましょう。

```
final _router = GoRouter(
  routes: [
    GoRoute(
      path: '/',
      builder: (context, state) => const FirstScreen(),
      routes: [
        GoRoute(
          path: 'second',
          builder: (context, state) => const SecondScreen(),
          routes: [
            GoRoute(
              path: 'third',
              builder: (context, state) => const ThirdScreen(),
            ),
          ],
        ),
      ],
    ),
    // GoRoute(
    //   path: '/third',
    //   builder: (context, state) => const ThirdScreen(),
    // ),
  ],
);
```

ThirdScreen画面へのGoRouteも移動しました。これによって、ThirdScreen画面へのパスが変化しますので、画面遷移処理も修正します。

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

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Thirdへ'),
              onPressed: () {
                // GoRouter.of(context).go('/third');
                GoRouter.of(context).go('/second/third'); ——❷
              },
            ),
```

```
            ElevatedButton(
              child: const Text('戻る'),
              onPressed: () {
                // Navigator.of(context).pop();
                GoRouter.of(context).pop(); ——❸
              },
            ),
          ],
        ),
      ),
    );
  }
}

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

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('ThirdScreen'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              child: const Text('戻る'),
              onPressed: () {
                // Navigator.of(context).pop();
                GoRouter.of(context).pop();  ——❹
              },
            ),
          ],
        ),
      ),
    );
  }
}
```

ThirdScreen画面への遷移処理を修正しました(❶、❷)。それでは動作確認してみましょう。

画面左上のバックボタンは表示され、SecondScreen画面の「戻る」ボタン、ThirdScreen画面の「戻る」ボタンをタップしてもアサーションエラーは発生しません。さらに、FirstScreen画面からThirdScreen画面への遷移し、「戻