Flutter 페이지 네비게이션 개발 가이드

Flutter 네비게이션 기초

학습목표

- Flutter에서 페이지 간 네비게이션 구현 방법 이해
- 다양한 네비게이션 패턴(드로어, 탭바, 하단 네비게이션) 구현 방법 습득
- 페이지 간 데이터 전달 방법 학습

Flutter 네비게이션 개요

- Flutter는 라우트(Routes)를 사용하여 페이지 간 이동 메커니즘 제공
- MaterialPageRoute를 통해 새 페이지를 애플리케이션 스택에 추가
- 명령형(Navigator V1)과 선언형(Navigator V2) 두 가지 네비게이션 방식 지원
- 네비게이션 드로어, 탭바, 하단 네비게이션 바 등 다양한 UI 패턴 제공

명령형 네비게이션 구현

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

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

class MyApp extends StatelessWidget {
  MyApp({Key? key}) : super(key: key);

  final List<String> items = [
    'January', 'February', 'March', 'April',
    'May', 'June', 'July', 'August',
    'September', 'October', 'November', 'December'
  ];

  @override
  Widget build(BuildContext context) {
    const title = 'MyAwesomeApp';
    return MaterialApp(
      title: title,
      home: Scaffold(
        appBar: AppBar(
          title: const Text(title),
        ),
        body: ListView.builder(
          itemCount: items.length,
          itemBuilder: (context, index) {
            return MyListView(items[index]);
          },
        ),
      ),
    );
  }
}
```

- **명령형 네비게이션**: `Navigator.push`를 사용하여 새 페이지를 스택에 추가합니다.
- `MaterialApp` 위젯은 앱의 기본 구조를 설정하고, `Scaffold`는 앱의 레이아웃을 정의합니다.
- `ListView.builder`를 사용하여 동적으로 리스트를 생성하고, 각 아이템을 `MyListView` 위젯으로 표시합니다.

MyListView 구현

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

class MyListView extends StatelessWidget {
  const MyListView(this.title);

  final String title;

  @override
  Widget build(BuildContext context) {
    return ListTile(
      title: Text(title),
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => MyDetails(title),
          ),
        );
      },
    );
  }
}
```

- **`MyListView` 위젯**: 리스트의 각 항목을 나타내며, `onTap` 이벤트 발생 시 `Navigator.push`를 사용하여 `MyDetails` 페이지로 이동합니다.
- `MaterialPageRoute`는 새로운 페이지를 생성하고, 애니메이션 효과를 제공합니다.

MyDetails 구현

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

class MyDetails extends StatelessWidget {
  const MyDetails(this.itemTitle);

  final String itemTitle;

  @override
  Widget build(BuildContext context) {
    const title = 'Details Page';
    return Scaffold(
      appBar: AppBar(
        title: const Text(title),
      ),
      body: SafeArea(
        top: false,
        bottom: false,
        child: Padding(
          padding: const EdgeInsets.all(8.0),
          child: Column(
            children: [
              SizedBox(
                height: 338.0,
                width: 800.0,
                child: Card(
                  clipBehavior: Clip.antiAlias,
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [
                      Padding(
                        padding: const EdgeInsets.all(10.0),
                        child: Text(itemTitle),
                      ),
                    ],
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

- `MyDetails` **위젯**: 선택된 항목의 상세 정보를 표시하는 페이지입니다.
- `SafeArea` 위젯은 화면의 안전한 영역 내에 콘텐츠를 배치하여, 노치나 상태 표시줄과 같은 시스템 UI 요소에 가려지지 않도록 합니다.

선언형 네비게이션 구현

```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: 'Flutter Code Sample for Navigator',
      initialRoute: '/signup',
      routes: <String, WidgetBuilder>{
        '/': (BuildContext context) => const HomePage(),
        '/signup': (BuildContext context) => const SignUpPage(),
      },
    );
  }
}
```

- **선언형 네비게이션**: `routes` 속성을 사용하여 앱의 라우트를 정의합니다.
- `initialRoute`는 앱이 시작될 때 표시할 초기 라우트를 지정합니다.
- `routes` 맵은 라우트 이름과 해당 라우트에 해당하는 위젯 빌더를 연결합니다.

SignUpPage 구현

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

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

  @override
  Widget build(BuildContext context) {
    return Navigator(
      initialRoute: 'signup/personal_info',
      onGenerateRoute: (RouteSettings settings) {
        WidgetBuilder builder;
        switch (settings.name) {
          case 'signup/personal_info':
            builder = (BuildContext context) => const CollectPersonalInfoPage();
            break;
          case 'signup/choose_credentials':
            builder = (BuildContext context) => ChooseCredentialsPage(
              onSignupComplete: () {
                Navigator.of(context).pop();
              },
            );
            break;
          default:
            throw Exception('Invalid route: ${settings.name}');
        }
        return MaterialPageRoute<void>(builder: builder, settings: settings);
      },
    );
  }
}
```

- **`SignUpPage` 위젯**: 회원 가입 과정을 처리하는 페이지입니다.
- `Navigator` 위젯을 사용하여 내부 라우트를 관리하고, `onGenerateRoute` 콜백을 통해 라우트에 따라 다른 위젯을 빌드합니다.

네비게이션 드로어 구현

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

class MyDrawerWidget extends StatelessWidget {
  const MyDrawerWidget({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Drawer(
      child: ListView(
        children: [
          const DrawerHeader(
            child: Icon(Icons.home, size: 35),
          ),
          ListTile(
            leading: const Icon(Icons.home),
            title: const Text('Drawer Item #1'),
            onTap: () {
              Navigator.of(context).push(
                MaterialPageRoute(builder: (context) => DemoPageOne()),
              );
            },
          ),
          // 추가 ListTile 항목들...
        ],
      ),
    );
  }
}
```

- **`MyDrawerWidget` 위젯**: 앱의 네비게이션 드로어를 구현합니다.
- `Drawer` 위젯은 앱의 사이드 메뉴를 생성하며, `ListView`를 사용하여 메뉴 항목을 표시합니다.
- `ListTile` 위젯은 각 메뉴 항목을 나타내며, `onTap` 이벤트 발생 시 `Navigator.push`를 사용하여 해당 페이지로 이동합니다.

탭을 사용하여 콘텐츠 관리하기

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter and Dart Cookbook Demo',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        tabBarTheme: const TabBarTheme(
          labelColor: Colors.white,
          labelStyle: TextStyle(color: Colors.grey),
        ),
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter and Dart Cookbook'),
    );
  }
}

class MyHomePage extends StatelessWidget {
  final String title;

  const MyHomePage({
    Key? key,
    required this.title,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.black,
      body: DefaultTabController(
        length: 4,
        child: Scaffold(
          appBar: AppBar(
            title: const Text('MyAwesomeTabBar'),
            bottom: const TabBar(
              tabs: [
                Tab(
                  icon: Icon(Icons.home, color: Colors.white),
                  child: Text('Home',
                      style: TextStyle(fontWeight: FontWeight.bold)),
                ),
                Tab(
                  icon: Icon(Icons.account_balance, color: Colors.white),
                  child: Text('Account',
                      style: TextStyle(fontWeight: FontWeight.bold)),
                ),
                Tab(
                  icon: Icon(Icons.calculate, color: Colors.white),
                  child: Text('Payments',
                      style: TextStyle(fontWeight: FontWeight.bold)),
                ),
                Tab(
                  icon: Icon(Icons.credit_score, color: Colors.white),
                  child: Text('Card',
                      style: TextStyle(fontWeight: FontWeight.bold)),
                ),
              ],
            ),
          ),
          body: const TabBarView(
            children: [
              SizedBox(
                child: Center(
                  child: Text('Home Page Tab 1'),
                ),
              ),
              SizedBox(
                child: Center(
                  child: Text('Account Page Tab 2'),
                ),
              ),
              SizedBox(
                child: Center(
                  child: Text('Payments Page Tab 3'),
                ),
              ),
              SizedBox(
                child: Center(
                  child: Text('Card Page Tab 4'),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

- **탭을 사용한 콘텐츠 관리**: `TabBar`와 `TabBarView`를 사용하여 탭 인터페이스를 구현합니다.
- `DefaultTabController`는 탭의 상태를 관리하고, `TabBar`는 탭 버튼을 표시하며, `TabBarView`는 탭에 해당하는 콘텐츠를 표시합니다.

하단 네비게이션 바 추가하기

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

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

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return const MaterialApp(
      title: "Bottom Navigation Widget",
      home: MyBottomNavigationWidget(),
    );
  }
}

final List<Widget> _navigationPages = [
  const Center(child: Text('Page: Home')),
  const Center(child: Text('Page: News')),
  const Center(child: Text('Demo: Favorites')),
  const Center(child: Text('Demo: List')),
];

class MyBottomNavigationWidget extends StatefulWidget {
  const MyBottomNavigationWidget({Key? key}) : super(key: key);

  @override
  State<MyBottomNavigationWidget> createState() => _MyBottomNavigationWidget();
}

class _MyBottomNavigationWidget extends State<MyBottomNavigationWidget> {
  final appTitle = 'Bottom Navigation Widget';
  int _itemSelected = 0;

  void _bottomBarNavigation(int index) {
    setState(() {
      _itemSelected = index;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(appTitle),
      ),
      body: _navigationPages[_itemSelected],
      bottomNavigationBar: BottomNavigationBar(
        currentIndex: _itemSelected,
        onTap: _bottomBarNavigation,
        type: BottomNavigationBarType.fixed,
        items: const [
          BottomNavigationBarItem(icon: Icon(Icons.home), label: 'Home'),
          BottomNavigationBarItem(icon: Icon(Icons.info), label: 'News'),
          BottomNavigationBarItem(icon: Icon(Icons.favorite), label: 'Favorites'),
          BottomNavigationBarItem(icon: Icon(Icons.list), label: 'List'),
        ],
      ),
    );
  }
}
```

- **하단 네비게이션 바 추가**: `BottomNavigationBar` 위젯을 사용하여 하단 네비게이션 바를 구현합니다.
- `BottomNavigationBar`는 앱의 하단에 고정된 네비게이션 메뉴를 표시하며, `BottomNavigationBarItem`을 사용하여 각 메뉴 항목을 정의합니다.

키를 사용하여 정보 전달하기

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

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

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    const paramTitle = 'My Title';
    const paramName  = 'My Name';

    return MaterialApp(
      title: paramTitle,
      home: Scaffold(
        appBar: AppBar(
          title: const Text(paramTitle),
        ),
        body: const MyTextWidget(name: paramName, title: paramTitle),
      ),
    );
  }
}

class MyTextWidget extends StatelessWidget {
  final String title;
  final String name;
  const MyTextWidget({Key ?key, required this.title, required this.name})
    : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Text("$title $name"),
    );
  }
}
```

- **키를 사용하여 정보 전달**: `Key`를 사용하여 위젯 간에 정보를 전달합니다.
- `Key`는 위젯을 식별하고, 상태를 유지하는 데 사용됩니다.

요약

- Flutter는 라우트 기반의 페이지 네비게이션 시스템을 제공
- 명령형과 선언형 두 가지 네비게이션 패턴을 지원하여 다양한 사용 사례에 대응
- 네비게이션 드로어, 탭바, 하단 네비게이션 바 등 Material Design 기반의 UI 패턴 구현 가능

END