05

# Navigator

## 다양한 위젯

# 내비게이션 사용하기

## 라우트 이해하기


그림 14-1 앱의 화면 구성

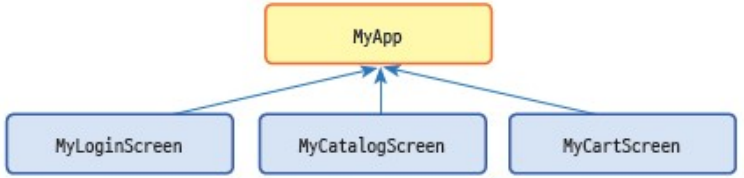


그림 14-2 위젯의 계층 구조

- 화면을 구성하는 여러 위젯을 준비하고 한 순간에 하나의 위젯만 보이게 화면을 전환
- 화면 전환을 제공하려면 Route와 Navigator를 사용

# 내비게이션 사용하기

## 라우트 이해하기

- Route는 화면을 지칭하는 객체
- Navigator 위젯은 Route 객체로 화면을 전환

• 앱 화면 구성

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: OneScreen(),
    );
  }

}

class OneScreen extends StatelessWidget {
  ... (생략) ...
}

class TwoScreen extends StatelessWidget {
  ... (생략) ...
}
```

# 내비게이션 사용하기

## 라우트 이해하기

- 화면을 전환하고 싶다면 Navigator.push() 함수를 호출
- Navigator.push() 함수는 두 번째 매개변수로 지정한 라우트 객체를 Navigator가 관리하는 스택에 추가

• 화면 전환하기

```
ElevatedButton(
  child: Text('Go Two Screen'),
  onPressed: () {
    Navigator.push(
      context,
      MaterialPageRoute(builder: (context) => TwoScreen()),
    );
  },
),
```

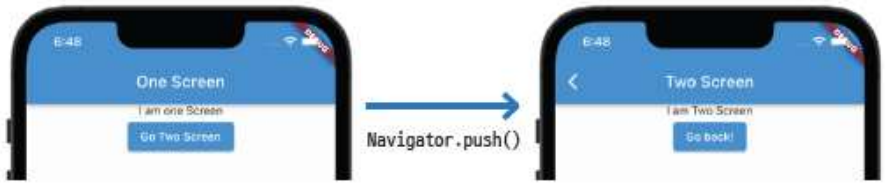


그림 14-3 Navigator.push()로 화면 전환

# 내비게이션 사용하기

## 라우트 이해하기

- 이전 화면으로 되돌아가자면 Navigator.pop() 함수를 이용

• 이전 화면으로 돌아가기

```
ElevatedButton(
  onPressed: () {
    Navigator.pop(context);
  },
  child: Text('Go back!'),
),
```

# 내비게이션 사용하기

## 라우트 이름으로 화면 전환하기

- MaterialApp의 routes 속성을 이용하면 앱의 화면을 등록하고 이 정보를 바탕으로 화면을 전환
- initialRoute 속성에는 처음 출력할 라우트 이름을 설정

• 라우트 등록하기

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => OneScreen(),
        '/two': (context) => TwoScreen(),
        '/three': (context) => ThreeScreen()
      },
    );
  }
}
```

# 내비게이션 사용하기

## 라우트 이름으로 화면 전환하기

- routes에 등록한 이름으로 화면을 전환하려면 Navigator의 pushNamed() 함수를 이용

• 라우트 이름으로 화면 전환하기

```
ElevatedButton(
  child: Text('Go Two Screen'),
  onPressed: () {
    Navigator.pushNamed(context, '/two');
  },
),
```

# 내비게이션 사용하기

## 화면 전환할 때 데이터 전달하기

- push() 함수로 화면 전환할 때 데이터 전달
  - 화면에 전달할 데이터는 생성자의 매개변수로 전달

• 데이터 전달하기

```
Navigator.push(
  context,
  MaterialPageRoute(builder: (context) => TwoScreen("hello"))
);
```

# 내비게이션 사용하기

## 화면 전환할 때 데이터 전달하기

- pushNamed() 함수로 화면 전환할 때 데이터 전달
  - arguments라는 매개변수를 이용
  - 전달한 데이터는 ModalRoute.of()를 이용해 획득

• 데이터 전달하기

```
ElevatedButton(
  onPressed: () {
    Navigator.pushNamed(
        context,
        '/three',
        arguments: 10
    );
  },
  child: Text('Go Three Screen'),
),
```

• 데이터 얻기

```
int arg = ModalRoute.of(context)?.settings.arguments as int;
```

# 내비게이션 사용하기

## 화면 전환할 때 데이터 전달하기

- 여러 개의 데이터를 전달하려면 JSON으로 데이터

• JSON 타입으로 데이터 여러 개 전달하기

```
Navigator.pushNamed(
    context,
    '/three',
    arguments: {
      "arg1": 10,
      "arg2": "hello"
    });
```

• 사용자 정의 객체 전달하기

```
Navigator.pushNamed(
    context,
    '/three',
    arguments: {
      "arg1": 10,
      "arg2": "hello",
      "arg3": User('kkang','seoul')
    });
```

• JSON 타입의 데이터 얻기

```
Map<String, Object> args = ModalRoute.of(context)?.settings.arguments as Map<String,
Object>;
```

# 내비게이션 사용하기

## 화면 전환할 때 데이터 전달하기

- pop() 함수로 화면 전환할 때 데이터 전달
  - pop() 함수의 두 번째 매개변수를 이용

• 데이터 전달하기

```
ElevatedButton(
    onPressed: () {
      Navigator.pop(context, 'world');
    },
    child: Text('Go Back')
),
```

• 데이터 얻기

```
onPressed: () async {
  final result = await Navigator.push(
    context,
    MaterialPageRoute(builder: (context) => TwoScreen("hello"))
  );
  print('result: ${result}');
},
```

# 내비게이션 사용하기

## 화면 전환할 때 데이터 전달하기

• 사용자 정의 객체 전달하기

```
ElevatedButton(
    onPressed: () {
      Navigator.pop(context, User('kim', 'busan'));
    },
    child: Text('Go Back')
),
```

• 사용자 정의 객체 얻기

```
onPressed: () async {
  final result = await Navigator.pushNamed(
      context,
      '/three'
  );
  print('result:${(result as User).name}');
},
```

# 내비게이션 사용하기

## 동적 라우트 등록 방법 — onGenerateRoute

- MaterialApp에 라우트를 등록할 때 onGenerateRoute 속성을 이용하는 방법
- 동적인 라우트가 필요할 때 onGenerateRoute 속성을 이용

• 동적 라우트 등록

```
onGenerateRoute: (settings) {
  if (settings.name == '/two') {
    final arg = settings.arguments;
    if (arg != null) {
      return MaterialPageRoute(
          builder: (context) => ThreeScreen(),
          settings: settings
      );
    } else {
      return MaterialPageRoute(
          builder: (context) => TwoScreen(),
      );
    }
  }
}
```

# 내비게이션 사용하기

## 동적 라우트 등록 방법 — onGenerateRoute

- onGenerateRoute 속성에 등록하는 함수의 매개변수는 라우트 정보가 담긴 RouteSettings 객체
- name 속성은 화면을 전환할 때 지정한 라우트 이름
- arguments 속성은 전달한 데이터
- MaterialApp의 routes와 onGenerateRoute 속성을 함께 등록해도 됩니다.

# 내비게이션 사용하기

## 동적 라우트 등록 방법 — onGenerateRoute

• routes와 onGenerateRoute 속성에 똑같은 라우트 이름 등록

```
return MaterialApp(
  initialRoute: '/',
  routes: {
    '/': (context) => OneScreen(),
    '/two': (context) => TwoScreen(),
    '/three': (context) => ThreeScreen()
  },
  onGenerateRoute: (settings) {
    if (settings.name == '/two') {
      return MaterialPageRoute(
          builder: (context) => ThreeScreen(),
          settings: settings
      );
    } else if (settings.name == '/four') {
      return MaterialPageRoute(
          builder: (context) => ThreeScreen(),
          settings: settings
      );
    }
  },
);
```

# 스택 제어

## 내비게이터 스택 제어하기

- Navigator는 라우트 객체를 스택 구조로 관리하는 위젯
- push()나 pushNamed() 함수로 라우트 객체를 생성하면 스택에 추가


그림 14-6 push(), pushNamed() 호출 후 스택 구조

- pop() 함수가 호출되면 스택에서 맨 위에 있는 라우트 객체를 제거한 후 그다음 라우트에 해당하는 화면이 출력

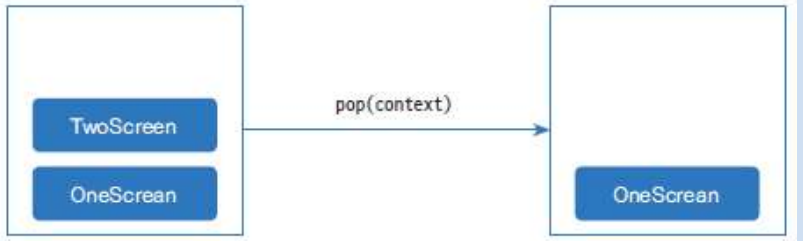


그림 14-7 pop() 호출 후 스택 구조

# 스택 제어

## 내비게이터 스택 제어하기

- Navigator로 화면을 전환할 때 위젯은 싱글톤singleton으로 동작하지 않습니다.
- OneScreen 화면이 나온 적이 있어서 스택에 있더라도 다시 OneScreen으로 화면을 전환하면 객체가 다시 생성


그림 14-8 중복 호출 후 스택 구조

# 스택 제어

## 내비게이터 스택 제어하기

- mybePop()과 canPop() 함수
  - maybePop() 함수는 위젯이 스택 맨 아래에 있지 않다면 이전 화면으로 되돌아가고 스택 맨 아래에 있다면 아무 일도 일어나지 않습니다.
  - canPop() 함수는 스택에서 제거할 수 있으면(스택 맨 아래에 있지 않다면) true, 제거할 수 없으면(스택 맨 아래에 있다면) false를 반환

• 맨 아래 스택 보호

```
Navigator.maybePop(context);
```

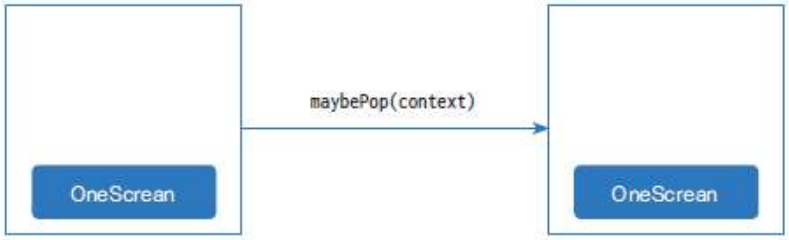


그림 14-9 maybePop() 호출 후 스택 구조

• 스택에서 제거할 수 있는지 확인

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

# 스택 제어

## 내비게이터 스택 제어하기

- pushReplacementNamed(), popAndPushNamed() 함수
  - pushReplacementNamed(), popAndPushNamed()는 현재 위젯을 대체하거나 제거한 후 새로운 위젯을 실행하는 함수

• 위젯 대체와 제거 후 새 위젯 실행 함수

```
Navigator.pushReplacementNamed(context, '/three');
// 또는
Navigator.popAndPushNamed(context, '/three');
```

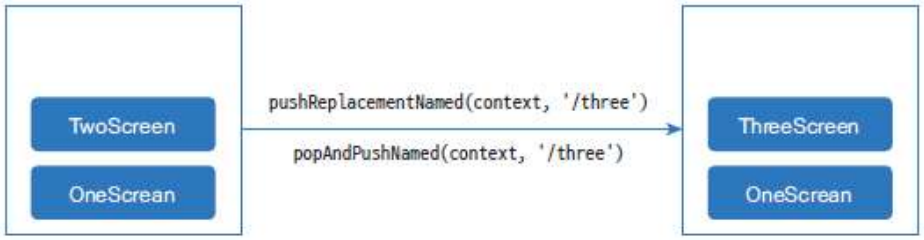


그림 14-10 pushReplacementNamed() 혹은 popAndPushNamed() 호출 후 스택 구조

# 스택 제어

## 내비게이터 스택 제어하기

- pushNamedAndRemoveUntil() 함수
  - pushNamedAndRemoveUntil() 함수도 특정 위젯을 화면에 출력하는 함수입니다. 그런데 원하는 위치까지 스택에서 제거한 후 화면을 이동
  - 함수의 세 번째 매개변수에 지정한 함수가 true를 반환하면 pushNamed() 함수와 똑같이 동작

• 위젯 추가하기

```
Navigator.pushNamedAndRemoveUntil(context, '/four', (route) => true);
```

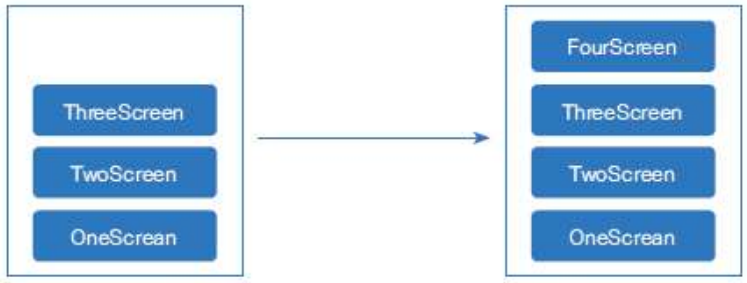


그림 14-11 세 번째 매개변수의 함수에서 true를 반환한 경우

# 스택 제어

## 내비게이터 스택 제어하기

- pushNamedAndRemoveUntil() 함수의 세 번째 매개변수에 지정한 함수가 false를 반환하면 스택을 비우고 새로운 위젯을 추가

• 스택을 비우고 추가하기

```
Navigator.pushNamedAndRemoveUntil(context, '/four', (route) => false);
```

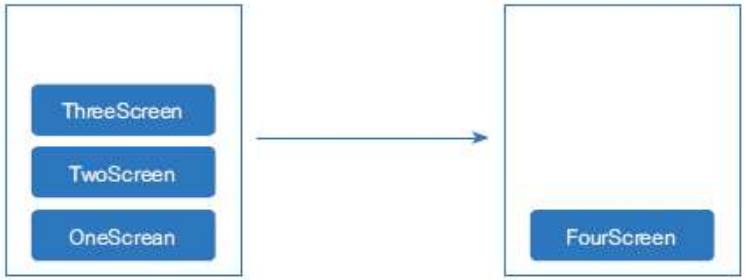


그림 14-12 세 번째 매개변수의 함수에서 false 가 리턴되는 경우

# 스택 제어

## 내비게이터 스택 제어하기

- pushNamedAndRemoveUntil() 함수의 세 번째 매개변수에 withName() 함수로 스택에 있는 특정 위젯을 지정하면 해당 위젯 위에 있는 위젯들만 스택에서 제거한 후 새로운 위젯을 추가

• 특정 위젯까지만 남기고 추가하기

```
Navigator.pushNamedAndRemoveUntil(context, '/four', ModalRoute.withName('/one'));
```

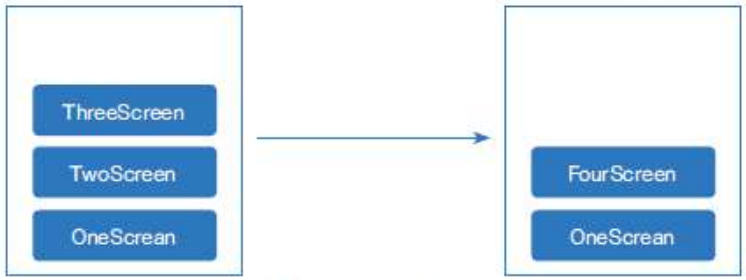


그림 14-13 세 번째 매개변수의 함수에서 특정 위젯이 지정된 경우

# 스택 제어

## 내비게이터 스택 제어하기

- popUntil() 함수
  - popUntil() 함수의 두 번째 매개변수에 스택에 있는 위젯을 설정하면 이 위젯 위에 있는 모든 위젯을 제거

• 특정 위젯으로 되돌아가기

```
ElevatedButton(
  onPressed: () {
    Navigator.popUntil(context, ModalRoute.withName('/two'));
  },
  child: Text('Pop Until'),
),
```

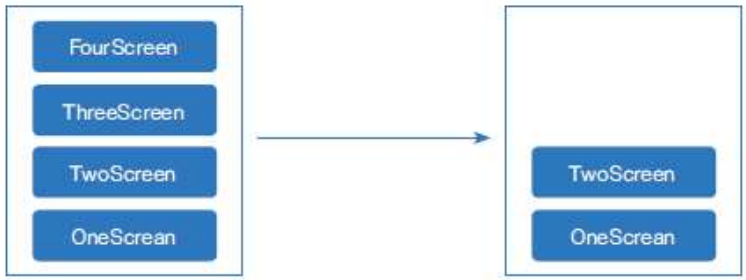


그림 14-14 popUntil() 호출 후 스택 구조

# 감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare