s_step3.dart

소스 코드

전체 코드

```dart
import 'package:buyoil/view/widget/w_step_nav.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:go_router/go_router.dart';

import '../../common/app_colors.dart';
import '../../common/app_strings.dart';
import '../../common/app_styles.dart';
import '../../model/ui_state_step3.dart';
import '../../router.dart';
import '../../viewmodel/vm_step3.dart';
import '../widget/w_header.dart';

class Step3Screen extends ConsumerStatefulWidget {
  const Step3Screen({Key? key}) : super(key: key);

  @override
  ConsumerState<ConsumerStatefulWidget> createState() => Step3ScreenState();
}

class Step3ScreenState extends ConsumerState<Step3Screen> {
  @override
  void initState() {
    super.initState();
    afterLayout();
  }

  @override
  Widget build(BuildContext context) {
    final notifier = ref.watch(step3Provider.notifier);
    final state = ref.watch(step3Provider);
    return Scaffold(
      body: Column(
        children: [
          HeaderWidget(),
          Expanded(
            child: state.when(
              init: () {
                return _initBody();
              },
              closeDoor: () {
                return _closeDoorBody();
              },
              completed: () {
```

```
                return Container();
              })
          )
        ],
      )
    );
  }

  Widget? _closeButton(BuildContext context, WidgetRef ref) {
    return InkWell(
      borderRadius: BorderRadius.circular(338 / 2),
      onTap: () {
        ref.watch(step3Provider.notifier).pressedClose();
      },
      child: Container(
        width: 338,
        height: 338,
        decoration: BoxDecoration(image: DecorationImage(image:
Image.asset("${AppStrings.assetPath}img_close_btn.png", width: 338, height:
338,).image)),
        child: Center(
          child: Text(AppStrings.closeAction.tr(), style: AppStyles.tsOpenBtn,),
        ),
      )
    );
  }

  void afterLayout() {
    // ref.listenManual(step3Provider, (_, state) {
    //   if(state is UIStateStep3Completed) {
    //     context.goNamed(RouteGroup.Step4.name);
    //   }
    // });
  }

  Widget _initBody() {
    return Row(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: [
        StepNavWidget(currentStep: 3, totalSteps: 4),
        Expanded(
            child: Center(
              child: _closeButton(context, ref),
            )
        )
      ],
    );
  }

  _closeDoorBody() {
    return Container(
      width: double.maxFinite,
      height: double.maxFinite,
```

```
      color: AppColors.EFFDF6,
    );
  }
}
```

✅ 1. Step2 / Step3의 전체 구조(아키텍처) 분석

두 화면 모두 다음 패턴으로 구성되어 있음:

---

(1) 공통 구조

A. 구조 레이아웃

- `HeaderWidget()`
- 메인 영역 `Expanded()`:
  - Riverpod의 `state.when(...)`으로 상태별 UI 렌더링
  - Step2: `init`, `completed`
  - Step3: `init`, `closeDoor`, `completed`

---

(2) 이벤트 처리 흐름

공통적으로:

1. 버튼 클릭 →
2. `viewModel.notifier`에 있는
   - `pressedOpen()` (Step2)
   - `pressedClose()` (Step3)
3. ViewModel 내부에서 상태 변경 (UIStateStepX*)
4. ViewModel의 상태 변경을 감지하는 `ref.listenManual(...)`
5. UI 라우팅 수행 (context.goNamed(...))

---

(3) Step2의 추가 특징

✔ afterLayout()에서 상태 변경 listen

```
ref.listenManual(step2Provider, (_, state) {
  if(state is UIStateStep2Completed) {
    context.goNamed(RouteGroup.Step3.name);
  }
});
```

즉, Step2는:

- `pressedOpen()` 실행 →

- 상태가 `UIStateStep2Completed` →
- 자동으로 Step3 화면으로 이동

---

(4) Step3의 문제점

❌ afterLayout()에서 listen이 모두 주석 처리됨

```
// ref.listenManual(step3Provider, (_, state) {
//   if(state is UIStateStep3Completed) {
//     context.goNamed(RouteGroup.Step4.name);
//   }
// });
```

즉 Step3는:

- 버튼 눌러도
- ViewModel에서 상태가 바뀌어도
- 라우팅 로직이 주석이라 다음 화면 Step4로 **절대 이동하지 않는다.**

이게 핵심 문제1.

✅ 2. Step2 / Step3 구조적 차이 및 문제점

✔ Step2

정상 동작함

- 상태 listen → OK
- init UI → 버튼 → 이벤트 → 상태 변화 → Step3 이동

완벽한 구조.

❌ Step3 — 구조 불완전

문제점:

(1) 상태 변화 감지가 아예 없음

`listenManual` 주석 때문에 상태가 완성되어도 Step4로 이동 불가.

(2) closeDoor 상태일 때 UI가 단순한 빈 바탕

```
_closeDoorBody() {
  return Container(
    width: double.maxFinite,
    height: double.maxFinite,
    color: AppColors.EFFDF6,
  );
}
```

Step2와 구조를 맞추려면 “작동 중 프로그레스 + 텍스트" 있어야 하는데 없음.

(3) Step2와 Step3의 패턴이 일관되지 않음

- Step2: opener 버튼 → progress UI → completed 이동
- Step3: closer 버튼 → progress UI 없음 → completed 이동 없음

(4) notifier 변수를 가져오지만 사용 안 함

```
final notifier = ref.watch(step3Provider.notifier);
final state = ref.watch(step3Provider);
```

notifier를 읽었지만 실제로 build 내부에서는 사용하지 않음.

(5) Step3 클래스 구조는 Step2를 복붙한 티가 너무 강함

복붙 후 미완성된 형태로 남아 있음:

- 변수명 / 함수명 / UI/ 라우팅 구성 동일
- but 중요한 부분만 누락됨

✅ 3. Step3에서 의도했던 기능(추정)

Step2 Step3 구조를 비교해보면 의도는 다음 흐름:

✔ Step2 흐름

1. 문 열기 버튼
2. 서버에 “open door request”
3. 상태: openingDoor
4. UI: 회전 이미지 + 문 열리는 중
5. 완료되면 Step3 이동

✔ Step3 의도된 흐름 (추정)

1. 문 닫기 버튼 (`pressedClose`)
2. 서버에 “close door request”
3. 상태: closeDoor
4. UI: 문 닫는 중 회전 UI 필요
5. 완료되면 Step4 이동

❌ Step3 현재 코드의 문제

- (3), (4), (5)가 비어 있음
- 라우팅 주석 처리됨
- closeDoor UI 텅 비어 있음
- 완료 상태 UI 없음

✅ 4. Step3이 정상 구조가 되려면 필요한 코드

🔧 afterLayout() 복구

```
void afterLayout() {
  ref.listenManual(step3Provider, (_, state) {
    if(state is UIStateStep3Completed) {
      context.goNamed(RouteGroup.Step4.name);
    }
  });
}
```

🔧 closeDoor UI 개선 (Step2와 동일한 프로그레스 필요)

```
_closeDoorBody() {
  return Container(
    width: double.maxFinite,
    height: double.maxFinite,
    color: AppColors.EFFDF6,
    alignment: Alignment.center,
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        SizedBox(
          width: 110,
          height: 110,
          child: RotatingImageCircularProgressBar(
            progress: 50,
            imagePath: "${AppStrings.assetPath}img_circular.png",
          ),
```

```
        ),
        SizedBox(height: 24),
        Text(AppStrings.doorClosing.tr(),
            style: AppStyles.tsDoorOpeningText),
      ],
    ),
  );
}
```

🔧 ViewModel Step3 로직 점검

- pressedClose → state = closeDoor → 일정 시간 후 completed
  이 구조가 만들어져야 화면 이동이 실행됨.

🧩 정리 — 지금 Step3의 상태

항목	Step2	Step3
버튼 있음	✔	✔
버튼 누르면 state 변경	✔	아마 있음
state listen	✔	❌ (주석)
프로그레스 UI	✔	❌
완료 시 라우팅	✔	❌

결론:
Step3 화면은 UI 구조만 만들어놓고 **핵심 기능 70%가 비어있는 상태**입니다.

✅ 🚀 완성된 Step3 리팩토링 코드

```
import 'package:buyoil/view/widget/w_step_nav.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/material.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:go_router/go_router.dart';

import '../../common/app_colors.dart';
import '../../common/app_strings.dart';
import '../../common/app_styles.dart';
import '../../model/ui_state_step3.dart';
import '../../router.dart';
import '../../viewmodel/vm_step3.dart';
import '../widget/w_header.dart';
import '../widget/circular_prograss.dart';

class Step3Screen extends ConsumerStatefulWidget {
  const Step3Screen({Key? key}) : super(key: key);
```

```
18
19    @override
20    ConsumerState<ConsumerStatefulWidget> createState() => Step3ScreenState();
21  }
22
23  class Step3ScreenState extends ConsumerState<Step3Screen> {
24    @override
25    void initState() {
26      super.initState();
27      afterLayout();
28    }
29
30    void afterLayout() {
31      // Step3 완료 상태 감지 → Step4 이동
32      ref.listenManual(step3Provider, (_, state) {
33        if (state is UIStateStep3Completed) {
34          context.goNamed(RouteGroup.Step4.name);
35        }
36      });
37    }
38
39    @override
40    Widget build(BuildContext context) {
41      final state = ref.watch(step3Provider);
42
43      return Scaffold(
44        body: Column(
45          children: [
46            const HeaderWidget(),
47            Expanded(
48              child: state.when(
49                init: () => _initBody(),
50                closeDoor: () => _closeDoorBody(),
51                completed: () => const SizedBox(),
52              ),
53            )
54          ],
55        ),
56      );
57    }
58
59    /// ============================
60    /// UI: 초기 화면 (닫기 버튼)
61    /// ============================
62    Widget _initBody() {
63      return Row(
64        mainAxisAlignment: MainAxisAlignment.spaceBetween,
65        children: [
66          const StepNavWidget(currentStep: 3, totalSteps: 4),
67          Expanded(
68            child: Center(
69              child: _closeButton(),
70            ),
```

```
        )
      ],
    );
  }

  Widget _closeButton() {
    return InkWell(
      borderRadius: BorderRadius.circular(169),
      onTap: () {
        ref.read(step3Provider.notifier).pressedClose();
      },
      child: Container(
        width: 338,
        height: 338,
        decoration: BoxDecoration(
          image: DecorationImage(
            image: Image.asset(
              "${AppStrings.assetPath}img_close_btn.png",
              width: 338,
              height: 338,
            ).image,
          ),
        ),
        child: Center(
          child: Text(
            AppStrings.closeAction.tr(),
            style: AppStyles.tsOpenBtn,
          ),
        ),
      ),
    );
  }

  /// ============================
  /// UI: 문 닫는 중
  /// ============================
  Widget _closeDoorBody() {
    return Container(
      width: double.infinity,
      height: double.infinity,
      color: AppColors.EFFDF6,
      alignment: Alignment.center,
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          SizedBox(
            width: 110,
            height: 110,
            child: RotatingImageCircularProgressBar(
              progress: 50,
              imagePath: "${AppStrings.assetPath}img_circular.png",
            ),
          ),
```

```
            const SizedBox(height: 24),
            Text(
              AppStrings.doorClosing.tr(),
              style: AppStyles.tsDoorOpeningText,
            ),
            const SizedBox(height: 3),
          ],
        ),
      );
  }
}
```

🔥 완성된 흐름

▶️ Step3 init 상태

- “닫기 버튼” 화면 표시

▶️ pressedClose() 호출

- ViewModel에서 `UIStateStep3CloseDoor()`로 상태 변경

▶️ closeDoor 화면

- 회전 이미지 + “문 닫는 중”

▶️ ViewModel에서 문 닫기 완료 후

- 상태를 `UIStateStep3Completed()`으로 변경

▶️ Step3Screen에서 해당 상태 감지 → Step4 이동

```
context.goNamed(RouteGroup.Step4.name);
```