main.dart

핵심 구조

```
runApp
 └ ProviderScope (Riverpod)
    └ EasyLocalization
       └ MyApp
          └ MaterialApp.router
             └ builder
                ├ Scaffold (Floating Debug Button)
                └ Overlay
                   ├ App Router UI
                   └ CustomToast
```

소스 코드

전체 코드

```dart
import 'package:buyoil/router.dart';
import 'package:buyoil/view/widget/debug_buttons.dart';
import 'package:easy_localization/easy_localization.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_riverpod/flutter_riverpod.dart';
import 'package:fluttertoast/fluttertoast.dart';

import 'common/utils/toast/custom_toast.dart';
import 'config.dart';

// ★ 애플리케이션 시작 지점
Future<void> main() async {
  // 전역 Config 설정 (디버그 모드 활성화)
  Config(isDebugMode: true);

  // Flutter 엔진 초기화 (서비스 등 호출 전 반드시 필요)
  WidgetsFlutterBinding.ensureInitialized();

  // 상단/하단 시스템 UI(상태바, 네비게이션바) 표시 설정
  await SystemChrome.setEnabledSystemUIMode(
    SystemUiMode.manual,
    overlays: [SystemUiOverlay.top, SystemUiOverlay.bottom],
  );

  // immersive 모드 → 전체 화면 몰입 모드 (탭하면 UI 등장)
  await SystemChrome.setEnabledSystemUIMode(SystemUiMode.immersive);

  // 화면 방향을 가로 고정
  await SystemChrome.setPreferredOrientations([
    DeviceOrientation.landscapeLeft,
```

```dart
    DeviceOrientation.landscapeRight,
  ]);

  // Riverpod ProviderScope + 다국어 지원 + 라우팅 MyApp 적용
  runApp(
    ProviderScope(
      child: EasyLocalization(
        supportedLocales: [
          Locale('en'),
          Locale('ko'),
          Locale('vi'),
          Locale('ja'),
        ],
        fallbackLocale: Locale('en'), // 설정 불가능 시 기본 언어
        startLocale: Locale('en'),
        path: 'assets/translations', // 번역 파일 경로
        child: const MyApp(),
      ),
    ),
  );
}

// ConsumerStatefulWidget → Riverpod State 관리 가능
class MyApp extends ConsumerStatefulWidget {
  const MyApp({super.key});
  @override
  createState() => _MyAppState();
}

class _MyAppState extends ConsumerState<MyApp> {
  // Debug 버튼 활성화 여부 상태
  bool isDialogShown = false;

  @override
  Widget build(BuildContext context) {
    return AnnotatedRegion<SystemUiOverlayStyle>(
      // 상/하단 UI 색상, 밝기 지정 (투명 처리)
      value: const SystemUiOverlayStyle(
        statusBarColor: Colors.transparent,
        statusBarIconBrightness: Brightness.dark,
        statusBarBrightness: Brightness.light,

        systemNavigationBarColor: Colors.transparent,
        systemNavigationBarDividerColor: Colors.transparent,
        systemNavigationBarIconBrightness: Brightness.dark,
      ),

      child: DefaultTextHeightBehavior(
        // 텍스트 렌더링 높이 관련 전역 설정
        textHeightBehavior: const TextHeightBehavior(
          applyHeightToFirstAscent: false,
          applyHeightToLastDescent: false,
          leadingDistribution: TextLeadingDistribution.proportional,
```

```
        ),

        child: MaterialApp.router(
          // EasyLocalization Delegates 설정
          localizationsDelegates: context.localizationDelegates,
          supportedLocales: context.supportedLocales,
          locale: context.locale,

          title: 'BuyOil',

          // Riverpod router provider 적용
          routerConfig: ref.watch(routerProvider),

          // Overlay 기반 커스텀 토스트 및 디버그 버튼 표시를 위해 builder 재정의
          builder: (context, child) {
            // Navigator가 빌드한 현재 화면 위젯
            final appContent = child;

            // Debug FloatingActionButton 추가를 위해 Scaffold 감쌈
            final appWithScaffold = Scaffold(
              backgroundColor: Colors.transparent,
              body: appContent,

              // 디버그 모드일 때만 FAB 표시
              floatingActionButton: Config.instance.isDebugMode
                  ? FloatingActionButton(
                      onPressed: () {
                        // 이미 표시중이면 닫기
                        if (isDialogShown) {
                          Navigator.of(rootNavigatorKey.currentContext!).pop();
                          setState(() => isDialogShown = false);
                        }
                        // 아니면 다이얼로그 열기
                        else {
                          setState(() => isDialogShown = true);
                          showDialog(
                            context: rootNavigatorKey.currentContext!,
                            barrierColor: Colors.transparent, // 뒷배경 투명
                            builder: (BuildContext dialogContext) {
                              return Column(
                                mainAxisAlignment: MainAxisAlignment.start,
                                children: [
                                  Padding(
                                    padding: EdgeInsets.only(left: 100),
                                    child: DebugButtons(), // 디버그 패널
                                  ),
                                ],
                              );
                            },
                          ).then((_) {
                            // 외부 탭으로 닫힌 경우 상태 복원
                            setState(() => isDialogShown = false);
                          });
```

```
                        }
                      },
                      backgroundColor: Colors.blueAccent.withOpacity(0.5),
                      child: Icon(
                        // 토글: 열려있으면 Close 아이콘
                        isDialogShown ? Icons.close : Icons.bug_report,
                        color: Colors.white,
                      ),
                    )
                  : null,
              floatingActionButtonLocation:
                  FloatingActionButtonLocation.startTop,
            );

            // ★ Overlay 구조 위에
            // 1) 라우터 화면
            // 2) CustomToast 위젯
            return Overlay(
              initialEntries: [
                OverlayEntry(
                  builder: (overlayContext) {
                    return Stack(
                      children: [
                        Positioned.fill(child: appWithScaffold),
                        const CustomToast(), // Global Toast (최상단 고정)
                      ],
                    );
                  },
                ),
              ],
            );
          },
        ),
      ),
    );
  }
}
```

Import Library

📦 1. Pub.dev 외부 패키지

✅ (1) easy_localization

```
import 'package:easy_localization/easy_localization.dart';
```

- 다국어(로컬라이제이션) 지원 패키지
- `supportedLocales`, `locale`, `tr()` 등을 제공

✅ (2) flutter_riverpod

```
import 'package:flutter_riverpod/flutter_riverpod.dart';
```

- 상태관리 패키지
- Provider, StateNotifier, FutureProvider 등 컴포지션 기반의 상태 관리

✅ (3) fluttertoast

```
import 'package:fluttertoast/fluttertoast.dart';
```

- OS 레벨 Toast 메시지 표시 라이브러리
- iOS/Android 기본 Toast 메시지 사용 시 필요
  (하지만 여기서는 CustomToast도 별도 사용 중)

📱 2. Flutter SDK 기본 패키지

🔹 (1) Material UI

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

- Flutter 기본 Material 디자인 위젯

🔹 (2) System UI/Orientation 제어

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

- `SystemChrome`, 화면 방향, 시스템 UI 오버레이 설정 등 시스템 관련 API

🗂️ 3. 프로젝트 내부 파일 (Local Project Files)

📌 (1) 라우팅 관련

```
import 'package:buyoil/router.dart';
```

- buyoil 프로젝트 내부의 앱 라우팅(Router, GoRouter 등)

📌 (2) 디버그 버튼 위젯

```
import 'package:buyoil/view/widget/debug_buttons.dart';
```

- 개발 모드에서 FloatingActionButton 클릭 시 나타나는 디버그 도구 UI

📌 (3) Custom Toast

```
import 'common/utils/toast/custom_toast.dart';
```

- 프로젝트 자체 구현 커스텀 Overlay Toast

📌 (4) Config

```
import 'config.dart';
```

- 앱 설정(디버그 모드 여부 등)을 관리하는 프로젝트 내부 파일

📚 정리 표

라이브러리 종류	패키지	비고
Flutter SDK	flutter/material.dart	UI
Flutter SDK	flutter/services.dart	시스템 UI/방향
Pub.dev 패키지	easy_localization	다국어
Pub.dev 패키지	flutter_riverpod	상태관리
Pub.dev 패키지	fluttertoast	Toast
프로젝트 내부	router.dart	라우팅
프로젝트 내부	debug_buttons.dart	디버그 UI
프로젝트 내부	custom_toast.dart	커스텀 토스트
프로젝트 내부	config.dart	설정 클래스

주요 특징

기능	구현 방식
전체 화면 몰입 UI	`SystemChrome.setEnabledSystemUIMode(SystemUiMode.immersive)`
가로 전용 앱	`setPreferredOrientations()`
다국어 지원	`EasyLocalization`
DI/상태 관리	Riverpod `ProviderScope()`
Router 설정	`MaterialApp.router()` & `routerProvider`
전역 Overlay Toast	`OverlayEntry + CustomToast()`
디버그 기능 토글 패널	FAB + `DebugButtons()` 다이얼로그