

main.dart

핵심 구조

```
1 runApp
2   ↳ ProvidersScope (Riverpod)
3     ↳ EasyLocalization
4       ↳ MyApp
5         ↳ MaterialApp.router
6           ↳ builder
7             ↳ Scaffold (Floating Debug Button)
8             ↳ Overlay
9               ↳ App Router UI
10              ↳ CustomToast
```

소스 코드

전체 코드

```
1 import 'package:buyoil/router.dart';
2 import 'package:buyoil/view/widget/debug_buttons.dart';
3 import 'package:easy_localization/easy_localization.dart';
4 import 'package:flutter/material.dart';
5 import 'package:flutter/services.dart';
6 import 'package:flutter_riverpod/flutter_riverpod.dart';
7 import 'package:fluttertoast/fluttertoast.dart';
8
9 import 'common/utils/toast/custom_toast.dart';
10 import 'config.dart';
11
12 // ★ 애플리케이션 시작 지점
13 Future<void> main() async {
14   // 전역 Config 설정 (디버그 모드 활성화)
15   Config(isDebugMode: true);
16
17   // Flutter 엔진 초기화 (서비스 등 호출 전 반드시 필요)
18   WidgetsFlutterBinding.ensureInitialized();
19
20   // 상단/하단 시스템 UI(상태바, 네비게이션바) 표시 설정
21   await SystemChrome.setEnabledSystemUIMode(
22     SystemUiMode.manual,
23     overlays: [SystemUiOverlay.top, SystemUiOverlay.bottom],
24   );
25
26   // immersive 모드 → 전체 화면 둘입 모드 (탭하면 UI 등장)
27   await SystemChrome.setEnabledSystemUIMode(SystemUiMode.immersive);
28
29   // 화면 방향을 가로 고정
30   await SystemChrome.setPreferredOrientations([
31     DeviceOrientation.landscapeLeft,
```

```
32     DeviceOrientation.landscapeRight,
33   ]);
34
35 // Riverpod ProviderScope + 다국어 지원 + 라우팅 MyApp 적용
36 runApp(
37   ProviderScope(
38     child: EasyLocalization(
39       supportedLocales: [
40         Locale('en'),
41         Locale('ko'),
42         Locale('vi'),
43         Locale('ja'),
44       ],
45       fallbackLocale: Locale('en'), // 설정 불가능 시 기본 언어
46       startLocale: Locale('en'),
47       path: 'assets/translations', // 번역 파일 경로
48       child: const MyApp(),
49     ),
50   ),
51 );
52 }
53
54 // Consumer StatefulWidget → Riverpod State 관리 가능
55 class MyApp extends Consumer StatefulWidget {
56   const MyApp({super.key});
57   @override
58   createState() => _MyAppState();
59 }
60
61 class _MyAppState extends ConsumerState<MyApp> {
62   // Debug 버튼 활성화 여부 상태
63   bool isDialogShown = false;
64
65   @override
66   Widget build(BuildContext context) {
67     return AnnotatedRegion<SystemUiOverlayStyle>(
68       // 상/하단 UI 색상, 밝기 지정 (투명 처리)
69       value: const SystemUiOverlayStyle(
70         statusBarColor: Colors.transparent,
71         statusBarIconBrightness: Brightness.dark,
72         statusBarBrightness: Brightness.light,
73
74         systemNavigationBarColor: Colors.transparent,
75         systemNavigationBarDividerColor: Colors.transparent,
76         systemNavigationBarIconBrightness: Brightness.dark,
77       ),
78
79       child: DefaultTextHeightBehavior(
80         // 텍스트 렌더링 높이 관련 전역 설정
81         textHeightBehavior: const TextHeightBehavior(
82           applyHeightToFirstAscent: false,
83           applyHeightToLastDescent: false,
84           leadingDistribution: TextLeadingDistribution.proportional,
```

```
85 ),
86
87     child: MaterialApp.router(
88         // EasyLocalization Delegates 설정
89         localizationsDelegates: context.localizationDelegates,
90         supportedLocales: context.supportedLocales,
91         locale: context.locale,
92
93         title: 'BuyoOil',
94
95         // Riverpod router provider 적용
96         routerConfig: ref.watch(routerProvider),
97
98         // Overlay 기반 커스텀 토스트 및 디버그 버튼 표시를 위해 builder 재정의
99         builder: (context, child) {
100             // Navigator가 빌드한 현재 화면 위젯
101             final appContent = child;
102
103             // Debug FloatingActionButton 추가를 위해 scaffold 감쌈
104             final appWithScaffold = Scaffold(
105                 backgroundColor: Colors.transparent,
106                 body: appContent,
107
108             // 디버그 모드일 때만 FAB 표시
109             floatingActionButton: Config.instance.isDebugEnabled
110                 ? FloatingActionButton(
111                     onPressed: () {
112                         // 이미 표시중이면 닫기
113                         if (isDialogShown) {
114                             Navigator.of(rootNavigatorKey.currentContext!).pop();
115                             setState(() => isDialogShown = false);
116                         }
117                         // 아니면ダイ얼로그 열기
118                         else {
119                             setState(() => isDialogShown = true);
120                             showDialog(
121                                 context: rootNavigatorKey.currentContext!,
122                                 barrierColor: Colors.transparent, // 뒷배경 투명
123                                 builder: (BuildContext dialogContext) {
124                                     return Column(
125                                         mainAxisAlignment: MainAxisAlignment.start,
126                                         children: [
127                                             Padding(
128                                                 padding: EdgeInsets.only(left: 100),
129                                                 child: DebugButtons(), // 디버그 패널
130                                             ),
131                                         ],
132                                         );
133                                     },
134                                 ).then((_) {
135                                     // 외부 탭으로 닫힌 경우 상태 복원
136                                     setState(() => isDialogShown = false);
137                                 });
138
139             );
140         );
141     );
142
143     // 외부 탭으로 닫힌 경우 상태 복원
144     setState(() => isDialogShown = false);
145   );
146
147   // 외부 탭으로 닫힌 경우 상태 복원
148   setState(() => isDialogShown = false);
149
150   // 외부 탭으로 닫힌 경우 상태 복원
151   setState(() => isDialogShown = false);
152
153   // 외부 탭으로 닫힌 경우 상태 복원
154   setState(() => isDialogShown = false);
155
156   // 외부 탭으로 닫힌 경우 상태 복원
157   setState(() => isDialogShown = false);
158
159   // 외부 탭으로 닫힌 경우 상태 복원
160   setState(() => isDialogShown = false);
161
162   // 외부 탭으로 닫힌 경우 상태 복원
163   setState(() => isDialogShown = false);
164
165   // 외부 탭으로 닫힌 경우 상태 복원
166   setState(() => isDialogShown = false);
167
168   // 외부 탭으로 닫힌 경우 상태 복원
169   setState(() => isDialogShown = false);
170
171   // 외부 탭으로 닫힌 경우 상태 복원
172   setState(() => isDialogShown = false);
173
174   // 외부 탭으로 닫힌 경우 상태 복원
175   setState(() => isDialogShown = false);
176
177   // 외부 탭으로 닫힌 경우 상태 복원
178   setState(() => isDialogShown = false);
179
180   // 외부 탭으로 닫힌 경우 상태 복원
181   setState(() => isDialogShown = false);
182
183   // 외부 탭으로 닫힌 경우 상태 복원
184   setState(() => isDialogShown = false);
185
186   // 외부 탭으로 닫힌 경우 상태 복원
187   setState(() => isDialogShown = false);
188
189   // 외부 탭으로 닫힌 경우 상태 복원
190   setState(() => isDialogShown = false);
191
192   // 외부 탭으로 닫힌 경우 상태 복원
193   setState(() => isDialogShown = false);
194
195   // 외부 탭으로 닫힌 경우 상태 복원
196   setState(() => isDialogShown = false);
197
198   // 외부 탭으로 닫힌 경우 상태 복원
199   setState(() => isDialogShown = false);
200
201   // 외부 탭으로 닫힌 경우 상태 복원
202   setState(() => isDialogShown = false);
203
204   // 외부 탭으로 닫힌 경우 상태 복원
205   setState(() => isDialogShown = false);
206
207   // 외부 탭으로 닫힌 경우 상태 복원
208   setState(() => isDialogShown = false);
209
210   // 외부 탭으로 닫힌 경우 상태 복원
211   setState(() => isDialogShown = false);
212
213   // 외부 탭으로 닫힌 경우 상태 복원
214   setState(() => isDialogShown = false);
215
216   // 외부 탭으로 닫힌 경우 상태 복원
217   setState(() => isDialogShown = false);
218
219   // 외부 탭으로 닫힌 경우 상태 복원
220   setState(() => isDialogShown = false);
221
222   // 외부 탭으로 닫힌 경우 상태 복원
223   setState(() => isDialogShown = false);
224
225   // 외부 탭으로 닫힌 경우 상태 복원
226   setState(() => isDialogShown = false);
227
228   // 외부 탭으로 닫힌 경우 상태 복원
229   setState(() => isDialogShown = false);
230
231   // 외부 탭으로 닫힌 경우 상태 복원
232   setState(() => isDialogShown = false);
233
234   // 외부 탭으로 닫힌 경우 상태 복원
235   setState(() => isDialogShown = false);
236
237   // 외부 탭으로 닫힌 경우 상태 복원
238   setState(() => isDialogShown = false);
239
240   // 외부 탭으로 닫힌 경우 상태 복원
241   setState(() => isDialogShown = false);
242
243   // 외부 탭으로 닫힌 경우 상태 복원
244   setState(() => isDialogShown = false);
245
246   // 외부 탭으로 닫힌 경우 상태 복원
247   setState(() => isDialogShown = false);
248
249   // 외부 탭으로 닫힌 경우 상태 복원
250   setState(() => isDialogShown = false);
251
252   // 외부 탭으로 닫힌 경우 상태 복원
253   setState(() => isDialogShown = false);
254
255   // 외부 탭으로 닫힌 경우 상태 복원
256   setState(() => isDialogShown = false);
257
258   // 외부 탭으로 닫힌 경우 상태 복원
259   setState(() => isDialogShown = false);
260
261   // 외부 탭으로 닫힌 경우 상태 복원
262   setState(() => isDialogShown = false);
263
264   // 외부 탭으로 닫힌 경우 상태 복원
265   setState(() => isDialogShown = false);
266
267   // 외부 탭으로 닫힌 경우 상태 복원
268   setState(() => isDialogShown = false);
269
270   // 외부 탭으로 닫힌 경우 상태 복원
271   setState(() => isDialogShown = false);
272
273   // 외부 탭으로 닫힌 경우 상태 복원
274   setState(() => isDialogShown = false);
275
276   // 외부 탭으로 닫힌 경우 상태 복원
277   setState(() => isDialogShown = false);
278
279   // 외부 탭으로 닫힌 경우 상태 복원
280   setState(() => isDialogShown = false);
281
282   // 외부 탭으로 닫힌 경우 상태 복원
283   setState(() => isDialogShown = false);
284
285   // 외부 탭으로 닫힌 경우 상태 복원
286   setState(() => isDialogShown = false);
287
288   // 외부 탭으로 닫힌 경우 상태 복원
289   setState(() => isDialogShown = false);
290
291   // 외부 탭으로 닫힌 경우 상태 복원
292   setState(() => isDialogShown = false);
293
294   // 외부 탭으로 닫힌 경우 상태 복원
295   setState(() => isDialogShown = false);
296
297   // 외부 탭으로 닫힌 경우 상태 복원
298   setState(() => isDialogShown = false);
299
299 );
```

```

138         }
139     },
140     backgroundColor: colors.blueAccent.withOpacity(0.5),
141     child: Icon(
142         // 토글: 열려있으면 close 아이콘
143         isDialogShown ? Icons.close : Icons.bug_report,
144         color: colors.white,
145     ),
146     ),
147     : null,
148     floatingActionButtonLocation:
149         FloatingActionButtonLocation.startTop,
150 );
151
152 // ★ overlay 구조 위에
153 // 1) 라우터 화면
154 // 2) CustomToast 위젯
155 return Overlay(
156     initialEntries: [
157         OverlayEntry(
158             builder: (overlayContext) {
159                 return Stack(
160                     children: [
161                         Positioned.fill(child: appWithScaffold),
162                         const CustomToast(), // Global Toast (최상단 고정)
163                     ],
164                 );
165             },
166         ),
167     ],
168 );
169 },
170 ),
171 );
172 );
173 }
174 }

```

Import Library

📦 1. Pub.dev 외부 패키지

(1) easy_localization

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

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

(2) flutter_riverpod

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

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

(3) fluttertoast

```
1 | 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

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

- Flutter 기본 Material 디자인 위젯

◆ (2) System UI/Orientation 제어

```
1 | 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

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

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

(2) 디버그 버튼 위젯

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

- 개발 모드에서 FloatingActionButton 클릭 시 나타나는 디버그 도구 UI

▶ ⑤ Custom Toast

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

- 프로젝트 자체 구현 커스텀 Overlay Toast

▶ (4) Config

```
1 | 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	flutertoast	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	<code>SystemChrome.setEnabledSystemUIMode(SystemUiMode.immersive)</code>
가로 전용 앱	<code>setPreferredOrientations()</code>
다국어 지원	<code>EasyLocalization</code>
DI/상태 관리	Riverpod <code>ProviderScope()</code>
Router 설정	<code>MaterialApp.router()</code> & <code>routerProvider</code>
전역 Overlay Toast	<code>OverlayEntry + CustomToast()</code>
디버그 기능 툴 패널	FAB + <code>DebugButtons()</code> 다이얼로그

