

app_strings.dart

소스 코드

전체 코드

```
1 class AppStrings {
2   static const String assetPath = "assets/images/";
3
4   static const String startButton = "startButton";
5   static const String exitApp = "exitApp";
6   static const String enterPhoneNumber = "enterPhoneNumber";
7   static const String enterButton = "enterButton";
8   static const String doorOpening = "doorOpening";
9   static const String doorClosing = "doorOpening";
10  static const String openAction = "openAction";
11  static const String closeAction = "closeAction";
12  static const String okayButton = "okayButton";
13  static const String recheckButton = "recheckButton";
14  static const String settingsMenu = "settingsMenu";
15  static const String oilContainerLabel = "oilContainerLabel";
16  static const String measurementLabel = "measurementLabel";
17  static const String motorLabel = "motorLabel";
18  static const String valveLabel = "valveLabel";
19  static const String stopButton = "stopButton";
20
21  static const String step = "step";
22  static const String checkValidationToast = "checkValidationToast";
23  static const String showTheProverDriverCode = "showTheProverDriverCode";
24
25  static const String tryToConnect = "Trying to connect;";
26  static String cannotCreatePort = "Failed to create port";
27  static String cannotOpenPort = "Failed to open port";
28  static String trySleep = "120s / Write [CMD]SLEEP#";
29  static String noConnectableDevicesFound = "No connectable devices found";
30  static String deviceSelectionCanceled = "Device selection canceled";
31
32  static String fullContainer = "The container is full. Please contact staff";
33  static String accessDenied = "Access denied. Please check your information.";
34  static String systemError = "A system error has occurred. Please try again later.";
35  static String notAuthorized = "Not Authorized.";
36 }
```

1 파일 개요

파일명: lib/common/app_strings.dart

역할: Flutter 앱에서 사용하는 문자열 상수(Strings) 관리

주요 목적:

- UI, 버튼, 라벨, 토스트, 상태 메시지 등 앱 전반에서 반복되는 문자열 관리

- 코드 내 하드코딩 방지
- 다국어 대응(i18n) 전 단계에서 문자열 중앙화

사용 맥락 예시:

- 버튼 텍스트: `AppStrings.startButton`
- 안내 메시지/토스트: `AppStrings.checkValidationToast`
- 장치 연결 상태 표시: `AppStrings.tryToConnect`
- 라벨/스텝 표시: `AppStrings.step`

2 주요 기능

1. 이미지/자산 경로 관리

- `assetPath = "assets/images/"`
- 이미지 파일 경로를 문자열 상수로 정의, 추후 로딩 시 재사용

2. UI 요소 문자열

- 버튼: `startButton`, `exitApp`, `enterButton`, `okayButton`, `recheckButton`, `stopButton`
- 라벨: `oilContainerLabel`, `measurementLabel`, `motorLabel`, `valveLabel`
- 메뉴: `settingsMenu`

3. 상태/행동 관련 문자열

- 도어 상태: `doorOpening`, `doorClosing`
- 동작: `openAction`, `closeAction`

4. 단계/검증 관련

- 단계 표시: `step`
- 검증 토스트 메시지: `checkValidationToast`
- 운전자 코드 표시: `showTheProverDriverCode`

5. 장치/포트 상태 메시지

- 연결 시도: `tryToConnect`
- 포트 생성 실패: `cannotCreatePort`
- 포트 열기 실패: `cannotOpenPort`
- 장치 없음/취소: `noConnectableDevicesFound`, `deviceSelectionCanceled`
- 장치 절전 모드: `trySleep`

6. 오류/권한 메시지

- 컨테이너 가득 참: `fullContainer`
 - 접근 거부: `accessDenied`
 - 시스템 오류: `systemError`
 - 인증 실패: `notAuthorized`
-

3 구조 분석

```
1 AppStrings (class)
2   └─ 자산 경로
3     └─ assetPath
4   └─ 버튼 문자열
5     └─ startButton, exitApp, enterButton, okayButton, recheckButton, stopButton
6   └─ 라벨/메뉴
7     └─ oilContainerLabel, measurementLabel, motorLabel, valveLabel
8     └─ settingsMenu
9   └─ 단계/검증
10    └─ step, checkValidationToast, showTheProverDriverCode
11   └─ 상태/동작
12     └─ doorOpening, doorClosing, openAction, closeAction
13   └─ 장치/포트 및 오류 메시지
14     └─ tryToConnect, cannotCreatePort, cannotOpenPort, trySleep
15     └─ noConnectableDevicesFound, deviceSelectionCanceled
16     └─ fullContainer, accessDenied, systemError, notAuthorized
```

특징:

- 문자열을 중앙화하여 재사용 가능
- 앱 전반에서 일관된 메시지/텍스트 유지
- 포트/장치 관련 메시지, UI 버튼/라벨 문자열 모두 관리

4 동작 흐름 예시

1. 버튼 생성 시:

```
1 ElevatedButton(
2   onPressed: () {},
3   child: Text(AppStrings.startButton),
4 );
```

1. 토스트 메시지 표시 시:

```
1 ScaffoldMessenger.of(context).showSnackBar(
2   SnackBar(content: Text(AppStrings.checkValidationToast)),
3 );
```

1. 장치 연결 상태 표시:

```
1 Text(AppStrings.tryToConnect)
```

1. 오류 메시지 처리:

```

1  if (!authorized) {
2      showDialog(
3          context: context,
4          builder: (_) => AlertDialog(
5              title: Text("Error"),
6              content: Text(AppStrings.accessDenied),
7          ),
8      );
9  }

```

5 장점

- **중앙 관리:** 문자열을 한 곳에서 관리해 변경 용이
- **코드 가독성 향상:** `AppStrings.startButton` → 하드코딩 "startButton" 사용 대비 명확
- **재사용성:** UI, 토스트, 다이얼로그 등 다양한 곳에서 재사용 가능
- **국제화(i18n) 확장 용이:** 추후 `AppStrings` 를 로컬라이즈 구조로 교체 가능

6 단점 / 개선점

1. 중복 문자열

- `doorOpening` 과 `doorClosing` 이 둘 다 `"doorOpening"` 으로 설정됨
- 개선: `doorClosing = "doorClosing"` 으로 수정 필요

2. 상수/변수 혼합

- 일부는 `static const`, 일부는 `static String`
- 개선: 변경 불가능한 문자열은 `const` 로 통일

3. 국제화(i18n) 미적용

- 현재 문자열은 단일 언어 고정
- 개선: Flutter `intl` 패키지와 연동해 다국어 지원

4. 사용자 친화적 메시지 부족

- 예: `cannotOpenPort = "Failed to open port"` → UI 레벨에서 좀 더 안내 메시지 가능

7 개선된 사용 예시 (개념)

```

1  // 버튼 텍스트
2  TextButton(
3      onPressed: () {},
4      child: Text(AppStrings.startButton),
5  );
6
7  // 도어 상태 표시
8  Text(isOpening ? AppStrings.doorOpening : AppStrings.doorClosing),
9

```

```
10 // 장치 연결 상태
11 Text(AppStrings.tryToConnect),
12
13 // 오류 메시지
14 showDialog(
15   context: context,
16   builder: (_) => AlertDialog(
17     title: Text("Error"),
18     content: Text(AppStrings.systemError),
19   ),
20 );
```

- 문자열 변경 시 **앱 전역에 자동 반영**
- 오류/상태 메시지 통일 가능