

custom_test_input.dart

소스 코드

전체 코드

```
1 import 'package:flutter/material.dart';
2 import 'package:flutter/services.dart';
3
4 import ' ../../common/app_colors.dart';
5 import ' ../../common/app_styles.dart';
6
7 class CustomTextInputField extends StatefulWidget {
8   final String text;
9
10  const CustomTextInputField({
11    super.key,
12    required this.text,
13  });
14
15  @override
16  TextInputFieldState createState() => TextInputFieldState();
17 }
18
19 class TextInputFieldState extends State<CustomTextInputField> {
20   late TextEditingController controller;
21
22   @override
23   void initState() {
24     super.initState();
25     controller = TextEditingController(text: widget.text);
26   }
27
28   @override
29   void didUpdateWidget(covariant CustomTextInputField oldWidget) {
30     super.didUpdateWidget(oldWidget);
31
32     if (widget.text != oldWidget.text) {
33       controller.text = widget.text;
34     }
35   }
36
37   @override
38   void dispose() {
39     controller.dispose();
40     super.dispose();
41   }
42
43   @override
44   Widget build(BuildContext context) {
45     return SizedBox(
```

```

46     width: 370,
47     height: 70,
48     child: TextField(
49       controller: controller,
50       style: AppStyles.tsTextField,
51       cursorColor: AppColors.PRIMARY,
52       enabled: false,
53
54       inputFormatters: [
55         FilteringTextInputFormatter.digitsOnly,
56         LengthLimitingTextInputFormatter(11),
57       ],
58
59       decoration: InputDecoration(
60         contentPadding:
61         const EdgeInsets.symmetric(horizontal: 15.0, vertical: 10.0),
62
63         border: outlineInputBorder(
64           borderRadius: BorderRadius.circular(14),
65           borderSide: const BorderSide(color: AppColors.PRIMARY, width: 1),
66         ),
67         enabledBorder: OutlineInputBorder(
68           borderRadius: BorderRadius.circular(14),
69           borderSide: const BorderSide(color: AppColors.PRIMARY, width: 1),
70         ),
71         disabledBorder: outlineInputBorder(
72           borderRadius: BorderRadius.circular(14),
73           borderSide: const BorderSide(color: AppColors.PRIMARY, width: 1),
74         ),
75         focusedBorder: outlineInputBorder(
76           borderRadius: BorderRadius.circular(14),
77           borderSide: const BorderSide(color: AppColors.PRIMARY, width: 1),
78         ),
79       ),
80
81       keyboardType: TextInputType.text,
82     ),
83   );
84 }
85 }
```

1 파일 개요

파일명: lib/view/widget/step1/custom_test_input.dart

역할: Flutter에서 사용하는 커스텀 텍스트 입력 위젯 정의

주요 목적:

- 고정 크기와 스타일을 가진 숫자 입력 필드(**TextField**) 제공
- 부모 위젯에서 전달한 초기 텍스트를 표시
- 현재는 읽기 전용으로, 사용자 입력은 막혀 있음
- 입력 제한(숫자만, 최대 11자리)을 적용할 수 있는 구조

사용 맥락 예시:

- 전화번호, 주민등록번호, 숫자 기반 ID 등 읽기/표시 전용 입력 필드
 - UI 스타일 통일과 상태 관리 용이
-

2 주요 기능

1. 텍스트 초기화 및 동기화

- 부모 위젯에서 전달한 `text` 값을 `TextEditingController`에 초기화
- 부모에서 `text` 변경 시 자동 갱신

2. 입력 제한

- 숫자만 허용 (`FilteringTextInputFormatter.digitsOnly`)
- 최대 11자리 제한 (`LengthLimitingTextInputFormatter`)
- 현재 `enabled: false` 상태라 제한은 실제로 적용되지 않음

3. UI 스타일

- 고정 크기: `width: 370, height: 70`
- 라운드 테두리 적용(`borderRadius: 14`)
- 테두리 색상 및 두께 통일(`AppColors.PRIMARY`, `width: 1`)
- 커서 색상과 텍스트 스타일 적용 (`Appstyles.tsTextField`)

4. 상태 관리

- `TextEditingController`을 사용하여 텍스트 상태 관리
 - `initState` → 초기화, `didUpdateWidget` → 값 변경 시 갱신, `dispose` → 메모리 해제
-

3 구조 분석

```
1 | customTextField (StatefulWidget)
2 | └── TextEditingController (State)
3 |     └── controller: TextEditingController
4 |     └── initState(): controller 초기화
5 |     └── didUpdateWidget(): 부모 텍스트 변경 시 갱신
6 |     └── dispose(): controller 해제
7 |     └── build(): TextField 위젯 생성
8 |         └── sizedBox (370x70)
9 |             └── TextField
10 |                 └── controller
11 |                 └── style, cursorColor
12 |                 └── enabled: false
13 |                 └── inputFormatters (digitsOnly, length 11)
14 |                     └── InputDecoration (border, padding)
```

특징:

- 상태 관리가 깔끔하게 분리됨

- UI 속성과 로직이 `build()`에 집중되어 있음
 - 입력 필드가 읽기 전용이라 현재는 데이터 표시용
-

4 동작 흐름

1. 부모 위젯에서 `CustomTextField(text: "01012345678")` 생성
 2. `TextFieldState`의 `initState`에서 `controller` 초기화
 3. 화면 빌드 시 `TextField` 가 `controller`의 값으로 초기화
 4. 부모가 `text` 값을 변경하면 `didUpdateWidget` 이 호출되어 `controller.text` 갱신
 5. 사용자는 입력 불가(`enabled: false`)
 6. 메모리 해제 시 `dispose`에서 `controller` 해제
-

5 장점

- 부모-자식 데이터 동기화 잘 처리됨
 - UI 일관성: 테두리, 색상, 스타일 통일
 - 메모리 안전: controller dispose 처리
 - 입력 제한 기능 포함: 나중에 활성화 가능
-

6 단점 / 개선점

1. 읽기 전용 상태(`enabled: false`)
 - 현재 숫자 제한과 입력 포맷터가 의미 없음
 - 개선: 필요 시 활성화 가능하도록 옵션화 (`enabled`를 매개변수로)
 2. 고정 크기
 - `SizedBox(width: 370, height: 70)` → 화면 크기나 반응형 디자인에 취약
 - 개선: `width: double.infinity` 또는 부모 constraints 기반
 3. 재사용성
 - 현재 `text` 와 `enabled` 외에는 옵션이 없음
 - 개선: `placeholder`, `maxLength`, `inputType`, `style` 등 매개변수 추가
 4. 테스트 / 검증
 - 유닛 테스트나 위젯 테스트가 없음
 - 개선: 텍스트 변경, 포맷터, UI 상태 테스트 작성
 5. 기능 확장
 - 실시간 입력 시 `onChanged` 콜백 없음
 - 개선: 콜백을 추가하여 부모에서 입력 반응 가능
-

7 개선된 구조 예시 (개념)

```
1 CustomTextField(  
2   text: "01012345678",  
3   enabled: true,  
4   maxLength: 11,  
5   keyboardType: TextInputType.number,  
6   onChanged: (value) { ... },  
7 )
```

- 이렇게 하면 읽기/쓰기 모드 전환, 입력 제한, 콜백 모두 지원 가능
- 반응형 화면에도 대응 가능