toast_provider.dart

소스 코드

전체 코드

```dart
import 'dart:async';

import 'package:riverpod_annotation/riverpod_annotation.dart';
import 'toast_state.dart'; // ToastState 임포트

part 'toast_provider.g.dart'; // 생성될 파일, `flutter pub run build_runner build` 실행 필요

@riverpod
class Toast extends _$Toast { // 클래스 이름은 원하는 대로 (예: Toast, GlobalToast 등)
  // _$ 접두사가 붙은 생성된 클래스를 상속
  Timer? _timer; // 타이머를 Notifier 내에서 관리

  @override
  ToastState build() {
    ref.onDispose(() {
      _timer?.cancel();
    });

    return const ToastState();
  }

  void showToast(String message, {Duration duration = const Duration(seconds: 2)}) {
    _timer?.cancel(); // 이전 타이머가 있다면 취소

    state = state.copyWith(message: message, isVisible: true);

    _timer = Timer(duration, () {
      // 현재 상태의 메시지와 비교하여 의도치 않은 hide 방지
      if (state.message == message && state.isVisible) {
        hideToast();
      }
    });
  }

  void hideToast() {
    state = state.copyWith(isVisible: false, message: null);
    _timer?.cancel(); // hide가 명시적으로 호출될 때도 타이머 취소
    _timer = null;
  }

}
```

1️⃣ 파일 개요

파일명: `lib/common/utils/toast_provider.dart`
역할: 토스트 메시지 상태 관리(Riverpod Notifier) 정의
주요 목적:

- 앱 전체에서 사용하는 **토스트 메시지**의 상태를 중앙에서 관리
- 메시지 표시(`showToast`)와 숨김(`hideToast`) 기능 제공
- 일정 시간 후 자동으로 토스트를 숨기도록 **타이머**를 관리

사용 맥락 예시:

- 화면 상단/중앙에 `CustomToast` 위젯과 연동하여 토스트 표시
- 글로벌 상태 관리로 여러 위젯에서 접근 가능
- 자동 숨김, 중복 메시지 처리 등 안정적 메시지 표시

2️⃣ 주요 기능

1. **초기 상태 생성** (`build()`)
    - `ToastState` 초기화 (`message: null`, `isVisible: false`)
    - Notifier가 dispose될 때 타이머 자동 해제 (`ref.onDispose`)
2. **토스트 표시** (`showToast`)
    - 이전 타이머가 있다면 취소 (`_timer?.cancel()`)
    - `state`를 `message`와 `isVisible: true`로 갱신
    - 주어진 `duration` 후 토스트 자동 숨김
    - 현재 메시지와 상태를 비교해 의도치 않은 hide 방지
3. **토스트 숨김** (`hideToast`)
    - `state`를 `isVisible: false`, `message: null`로 갱신
    - 타이머 취소 및 null 처리
4. **타이머 관리**
    - `_timer` 변수로 단일 인스턴스 관리
    - 새로운 토스트가 표시되면 기존 타이머 취소
    - hide 호출 시에도 타이머 취소

---

3 구조 분석

```
Toast (Riverpod Notifier)
 ├─ _timer: Timer?          // 메시지 자동 숨김용
 ├─ build(): ToastState     // 초기 상태 생성
 |     └─ ref.onDispose(): 타이머 해제
 ├─ showToast(String message, {Duration duration})
 |     ├─ 이전 타이머 취소
 |     ├─ state 갱신: message, isVisible
 |     └─ Timer(duration): hideToast() 호출
 └─ hideToast()
       ├─ state 갱신: isVisible=false, message=null
       └─ 타이머 취소 및 null 처리
```

특징:

- **단일 소스 진실(Single Source of Truth)** 방식
- 자동 숨김 + 명시적 hide 모두 지원
- 중복 호출 및 이전 토스트 겹침 방지

---

4 동작 흐름

1. `showToast("Hello")` 호출
2. `_timer`가 존재하면 취소 후 새 타이머 생성
3. `state` 갱신 → `CustomToast` UI에서 표시
4. 지정된 시간(`duration`) 후 타이머 콜백 실행
5. 현재 메시지와 상태가 일치하면 `hideToast()` 호출
6. `hideToast()`에서 상태 초기화 및 타이머 해제

---

5 장점

- **글로벌 상태 관리**로 앱 전체에서 재사용 가능
- **자동 숨김** 기능 내장
- **중복 호출 방지**: 동일 메시지 중복 hide 방지
- Riverpod `ref.onDispose`로 메모리 안전

---

6 단점 / 개선점

1. **단일 메시지 제한**
    - 한 번에 하나의 토스트만 표시 가능
    - 개선: 큐(queue)를 만들어 다중 메시지 순차 표시
2. **UI 분리 의존**

- 토스트 표시 UI(`CustomToast`)와 직접 연결되지 않음
- 개선: 상태 변화 감지 후 애니메이션 적용 가능

3. **재사용성**
   - 현재 메시지, duration만 변경 가능
   - 개선: 배경색, 텍스트 스타일, 위치 등 옵션화
4. **타이머 관리 단순화 필요**
   - `_timer` 단일 변수 사용 → 복잡한 큐/애니메이션과 병행 시 한계

7️⃣ 개선 예시 (개념)

```
showToast(
  "Hello World",
  duration: Duration(seconds: 3),
  backgroundColor: Colors.black,
  textStyle: TextStyle(color: Colors.white, fontSize: 16),
);
```

- 메시지 스타일, 위치, 애니메이션 옵션 추가
- 여러 메시지를 큐로 관리하여 순차 표시 가능