Flutter & Cursor AI

1. Riverpod이란?

Riverpod은 상태 관리를 위한 라이브러리로, **Provider**를 기반으로 작동합니다. 이는 상태를 생성하고 관리하며, 의존성을 처리하는 방식으로 작동합니다. 주요 특징은 다음과 같습니다:

- **글로벌 상태 관리**: 앱 어디에서나 동일한 상태를 공유하고 사용할 수 있습니다.
- **Provider** 기반**: 상태는 `Provider`라는 객체를 통해 생성되고, 이를 통해 상태를 읽거나 업데이트합니다.
- **명시적 의존성 관리**: 각 상태는 명확한 의존성을 가지며, 이로 인해 상태 변화에 의존하는 부분만 갱신됩니다.
- **타입 안전**: 상태를 정의할 때 타입을 지정할 수 있어, 타입 안정성이 보장됩니다.
- **간편한 테스트**: Riverpod은 의존성 주입을 통해 상태를 쉽게 테스트할 수 있게 해줍니다.

Riverpod의 핵심 개념

- **Provider**: 상태를 생성하고 제공하는 역할을 합니다. `StateProvider`, `FutureProvider`, `StreamProvider` 등 다양한 종류가 있으며, 각기 다른 상태 유형을 다룰 수 있습니다.
- **ConsumerWidget**: 상태를 구독하고, 상태 변경에 반응하여 UI를 갱신하는 위젯입니다.
- **StateNotifier**: 복잡한 상태를 관리할 때 사용되는 클래스로, 상태를 직접적으로 변경할 수 있는 메서드를 제공합니다.

예시 (Riverpod 사용)

```
dart
복사
// 상태를 제공하는 Provider
```

```
final counterProvider = StateProvider<int>((ref) => 0);

// 상태를 사용하는 위젯
class CounterPage extends ConsumerWidget {
  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final counter = ref.watch(counterProvider); // 상태 구독

    return Scaffold(
      appBar: AppBar(title: Text('Counter')),
      body: Center(
        child: Text('Counter: $counter'),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: () => ref.read(counterProvider.notifie
r).state++, // 상태 변경
        child: Icon(Icons.add),
      ),
    );
  }
}
```

2. 전역 관리가 필요한 이유

앱이 커지고 복잡해짐에 따라 **상태 관리**의 중요성이 커집니다. 전역 상태 관리가 필요한 이유는 다음과 같습니다:

- **상태 공유**: 여러 위젯에서 동일한 데이터를 공유해야 할 경우, 전역 상태 관리가 필요합니다. 예를 들어, 사용자가 로그인한 상태를 앱 내 모든 페이지에서 접근해야 한다면, 이를 전역적으로 관리해야 합니다.
- **UI와 상태 분리**: UI와 상태를 분리해 관리함으로써 더 유연하고 재사용 가능한 컴포넌트를 만들 수 있습니다.
- **상태의 일관성 유지**: 앱이 커지면, 여러 부분에서 동일한 데이터를 다룰 수 있습니다. 전역 상태 관리는 이를 중앙에서 관리하여 상태의 일관성을 유지하는 데 도움을 줍니다.
- **복잡한 앱 로직 처리**: 전역 상태 관리 라이브러리를 사용하면 앱이 복잡해질 때, 상태 변화와 의존 관계를 명확히 정의하고, 관리할 수 있습니다.

3. Riverpod과 props의 비교

`props`는 주로 React에서 사용되는 개념으로, 위젯에 외부에서 전달되는 데이터입니다. Flutter에서는 `props`에 해당하는 개념이 **위젯의 생성자**로 구현됩니다. 하지만 이 방식은 상태를 전역적으로 관리하거나 여러 위젯 간의 상태 공유에 한계가 있습니다.

props의 특징:

- **상태 전파**: 부모에서 자식 위젯으로 데이터를 전달합니다.
- **위치 기반 관리**: 데이터를 특정 위치에서만 관리할 수 있기 때문에, 복잡한 앱에서는 상태를 상위 컴포넌트에서 계속 전달해야 하는 번거로움이 생깁니다.

Riverpod의 특징:

- **전역 상태 관리**: 앱 전역에서 상태를 관리할 수 있으며, 필요할 때만 의존성을 끌어옵니다.
- **상태 캡슐화**: 상태를 외부에서 직접 수정할 수 없고, 오직 `Provider`를 통해 상태를 관리합니다. 따라서 상태 관리가 안전하고 예측 가능해집니다.
- **자동 UI 갱신**: 상태가 변경되면 그 상태에 의존하는 모든 위젯이 자동으로 갱신됩니다.
- **Testability**: 상태가 명확하게 분리되므로, 각 상태에 대해 독립적인 테스트가 용이합니다.

Cursor AI

커서 ??

Cursor AI는 **AI 기반의 개발 도구** 또는 **개발자 지원 도구**입니다. 이 툴은 주로 **코드 자동완성**과 **코딩 작업을 효율적으로 도와주는 기능**을 제공합니다. 예를 들어, **GitHub Copilot**이나 **Tabnine**과 유사한 AI 코드 보조 툴로, 개발자가 코드를 작성할 때 실시간으로 코드의 완성, 제안 및 자동화를 도와줍니다.

주요 기능

1. **코드 자동완성**: 사용자가 코드를 작성하는 동안, AI가 그 다음에 어떤 코드를 입력할지 예측하고 제안합니다.

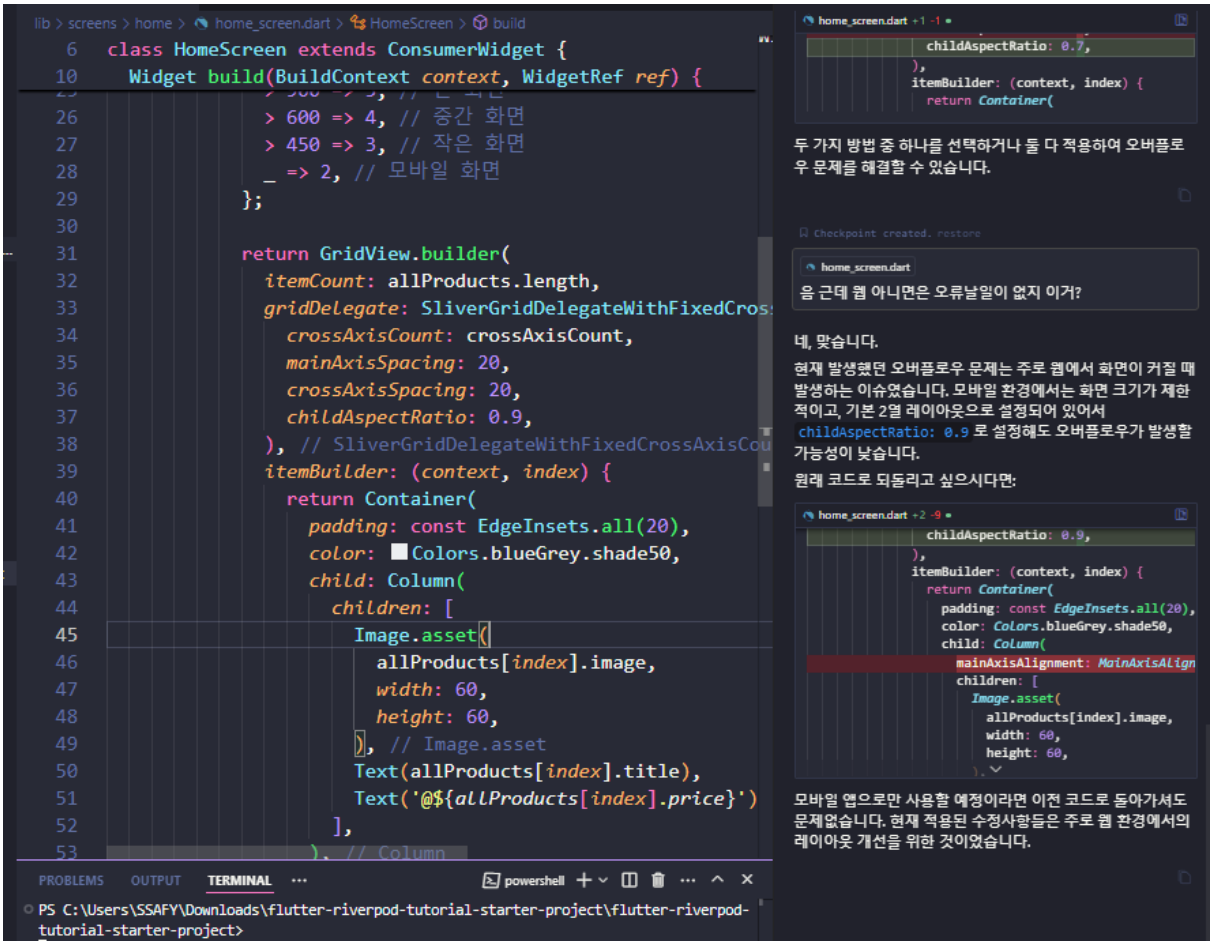
2. **코드 제안 및 리팩토링**: 반복되는 코드 패턴이나 개선이 필요한 코드를 AI가 분석하여 더 효율적이거나 간결한 코드로 제안할 수 있습니다.
3. **문제 해결**: 코드를 작성할 때 발생할 수 있는 오류나 문제를 예측하고 해결 방법을 제시합니다.
4. **설명 및 문서화**: 코드에 대한 설명을 자동으로 생성하거나, 코드의 목적과 기능을 간단하게 설명해줄 수 있습니다.

Cursor AI의 장점

- **생산성 향상**: 개발자는 반복적인 작업에서 벗어나 더 창의적이고 고차원적인 부분에 집중할 수 있습니다.
- **효율적인 코드 작성**: 코드 작성 시 AI가 실시간으로 제안해주기 때문에 작성 시간이 단축될 수 있습니다.
- **버그 감소**: 코드에서 발생할 수 있는 잠재적인 오류나 버그를 AI가 자동으로 인지하고 수정 제안을 하기도 합니다.

Cursor AI 를 이용한 Flutter 개발

- Cursor 는 VScode 와 비슷한 IDE 로 IDE 형 AI .
- 기본적으로 Cursor IDE 에서 작업을 하는데 Ctrl + L 을 누르면 Cursor AI 가 나온다.
- 아래 사진처럼 현재 내 코드를 기준으로 바로 업데이트를 해주기 때문에 작업속도의 효율이 엄청나게 증가할 수 있는 IDE 형 AI 이다.
- Cursor 와 VSCODE 에 같은 폴더를 오픈한 후 작업을 하면 Cursor 의 변경 후 저장을 누르면 VSCODE 에서도 바로 수정이 되도록 Files : auto 를 active 설정을 해주면 작업이 더 편해진다.
- Cursor 는 특성상 에뮬레이터 등 Flutter 의 코드 실행이 안되므로 vscode 와 연동해서 작업을 하면 효율이 증가한다.



결론

- Cursor AI 를 잘 활용한다면 ChatGPT 를 번거롭게 왔다갔다 할 필요없이 현재 내 코드 기준으로 좀 더 좋은 코드를 추천해주고 어디가 수정되는 지 초록색으로 표시를 해주기 때문에 ChatGPT 보다 효율이 좋은 AI 인 거 같다.