StateProvider

Einführung

Wie bereits erwähnt, kann ein normaler Provider die Werte nicht ändern. Daher werden Tools wie der `StateProvider` benötigt.

StateProvider (Kurzdefinition):

Ein `StateProvider` ist ein spezieller Provider, der den Zustand verwalten und dynamisch ändern kann.

1. **Aufbau:**

Der Aufbau eines `StateProvider` ist ähnlich wie der eines normalen Providers, jedoch wird die Klasse `StateProvider` anstelle von `Provider` verwendet:

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

2. **Provider im GUI aufrufen:** Das Aufrufen eines Providers im GUI ist genauso einfach wie bei einem normalen Provider:

```
final value = ref.watch(providerName);
```

3. **Manipulieren im GUI durch Button**

Um einen Provider ändern zu können, benutzt man den Code:
`ref.read(providerName.notifier)`
Danach fügt man die gewünschte Funktionalität ein. Änderungen können nur durch den Notifier vorgenommen werden, sonst erhält man einen Fehler.

Watch-Read Vergleich: Wenn man einen Provider lesen möchte, benutzt man `ref.watch`, da die `watch`-Funktion den Provider beobachtet und die Seite neu baut, sobald sich der Zustand ändert. Im Gegensatz dazu macht `read` nur einen einmaligen Blick auf den Provider beim ersten Aufbau der Seite und aktualisiert die Seite nicht, egal wie sich der Zustand des Providers ändert.

Wenn man den Providerwert im GUI darstellen möchte, muss man `watch` verwenden, um die neuen Werte sichtbar zu machen. In einem Button-Widget, wo man die Funktionen schreibt, reicht jedoch `read` aus, da man nicht möchte, dass der Button bei jeder Änderung des Providers neu gebaut wird.

`Ref` Objekt

Wie zuvor erwähnt, ist `ref` ein Objekt der Klasse `WidgetRef`, das Zugriff auf alle Provider hat. `ref` bietet viele wichtige Methoden, die man kennen sollte. Neben den bekannten `read` und `watch` gibt es weitere nützliche Methoden:

1. `listen`:
   Diese Methode ermöglicht den Zugriff auf den vorherigen und aktuellen Zustand eines Providers. Sie kann verwendet werden, um Aktionen auszuführen, wenn einer der Zustände einen bestimmten Wert erreicht oder eine bestimmte Beziehung zwischen dem vorherigen und aktuellen Zustand besteht.

2. `invalidate`:
   Mit dieser Methode kann ein Provider auf seinen Standardwert zurückgesetzt werden. Sie ist besonders nützlich, wenn man mit lokalem Speicher arbeitet. Wenn der lokale Speicher als Standardwert eines Providers verwendet wird, kann man mit `invalidate` sicherstellen, dass Änderungen im lokalen Speicher berücksichtigt und im GUI angezeigt werden.

Es gibt noch viele weitere Methoden, aber hier wurden nur die wichtigsten beschrieben.

Wichtiger Hinweis Das `ref`-Objekt kann nicht nur innerhalb von `ConsumerWidget`-Widgets verwendet werden, sondern auch direkt in den Providern selbst. Aus diesem Grund verlangt die Funktion, die beim Erstellen eines Providers übergeben wird, ein `ref`-Objekt als Parameter.

```
final myProvider = Provider<int>((ref) {
   final x= ref.watch(andereProvider);
   return x;
});
```

App-Beschreibung:

Für die BMI-Berechnung soll der Benutzer folgende Werte eingeben:

- **Größe**: Eingabe über einen `StateProvider` (`heightProvider`).
- **Gewicht**: Eingabe über einen `StateProvider` (`weightProvider`).
- **Geschlecht**: Auswahl über einen `StateProvider` (`genderProvider`).

Der berechnete BMI ist ein Wert, der nicht direkt vom Benutzer eingegeben wird, sondern aus den anderen Werten berechnet wird.
Dafür wird ein normaler Provider verwendet (`bmiProvider`), der die Werte der anderen Provider (`heightProvider`, `weightProvider`) abruft und den BMI berechnet.

Erklärung:

- Jeder eingegebene oder ausgewählte Wert wird mit einem eigenen `StateProvider` verwaltet.
- Innerhalb des BMI-Providers werden die anderen Provider über `ref` aufgerufen, um ihre Werte abzurufen und die Berechnung durchzuführen.
- Wenn sich der Zustand von `Größe`, `Geschlecht` oder `Gewicht` ändert, wird diese Änderung im `BMI-Provider` automatisch registriert. Der `BMI-Provider` beobachtet die Werte der entsprechenden State-Provider und berechnet den BMI basierend auf den aktuellen Werten neu. Dies stellt sicher, dass die GUI immer die aktualisierte BMI-Werte anzeigt.Durch die Verwendung von `ref.watch` im `BMI-Provider` wird jede Änderung an den beobachteten Providern erfasst, was dazu führt, dass der `BMI-Provider` neu gebaut wird.