Recoil in Next.js-Projekt

Recoil State Management in Next.js-Projekt

Wiki für Recoil State Management in Next.js-Pro

Überblick

Diese Wiki-Seite dokumentiert den Einsatz und die Konfiguration von Recoil für das State Management in unserem Next.js-Projekt. Recoil wird verwendet, um den Zustand unserer Anwendung auf einfache und effektive Weise zwischen den Komponenten zu teilen.

Einrichtung

Installation

Um Recoil in Ihrem Projekt zu verwenden, installieren Sie es zuerst über npm:

```
npm install recoil
```

Einrichten des `RecoilRoot`

Um Recoil in Ihrer Anwendung zu nutzen, müssen Sie die gesamte Anwendung (oder zumindest den Teil, den Zustand verwaltet) mit einem `<RecoilRoot>` -Provider umschließen. Dies geschieht typischerweise in der `_app.js` -Datei von Next.js:

```
// pages/_app.js
import { RecoilRoot } from 'recoil';

function MyApp({ Component, pageProps }) {
  return (
    <RecoilRoot>
      <Component {...pageProps} />
    </RecoilRoot>
  );
}

export default MyApp;
```

Verwendung von Atomen

Atome sind die grundlegenden Einheiten des Zustands in Recoil. Jedes Atom hat einen eindeutigen Schlüs
und einen Standardwert.

Definieren eines Atoms

Definieren Sie Atome in einer zentralen Datei, um sie leicht in verschiedenen Komponenten
wiederzuverwenden:

```
// features/gisStationState.js
import { atom } from 'recoil';

export const gisStationsStaticDistrictState = atom({
  key: 'gisStationsStaticDistrict',
  default: [],
});
```

Zustand in Komponenten verwenden

Recoil bietet verschiedene Hooks, um den Zustand in Ihren Komponenten zu verwalten:

- `useRecoilState` : Gibt ein Zustandspaar zurück (ähnlich wie `useState`), das den Zustandswert und eine Setter-Funktion enthält.
- `useRecoilValue` : Gibt nur den aktuellen Zustandswert zurück, ohne die Möglichkeit zur Aktualisierung.
- `useSetRecoilState` : Gibt nur eine Funktion zum Aktualisieren des Zustands zurück, was die Komponente effizienter macht, da sie nicht bei jeder Zustandsänderung neu gerendert wird.

Beispiele

MapComponent.js

In `MapComponent.js` verwenden Sie `useRecoilState` , um den Zustand sowohl zu lesen als auch zu aktualisieren:

```
// components/MapComponent.js
import { useRecoilState } from 'recoil';
import { gisStationsStaticDistrictState } from '../features/gisStationState';

const MapComponent = () => {
  const [GisStationsStaticDistrict, setGisStationsStaticDistrict] = useRecoilSt
```

```
  const updateDistricts = (districts) => {
    setGisStationsStaticDistrict(districts);
  };

  return (
    // Komponenten-Rendering
  );
};
```

DataSheet.js

```
// components/DataSheet.js
import React from "react";
import { useRecoilValue } from 'recoil';
import { gisStationsStaticDistrictState } from '../features/gisStationState';

function DataSheet() {
  const GisStationsStaticDistrict = useRecoilValue(gisStationsStaticDistrictSta

  return (
    <div id="mainDataSheet" className="absolute top-3 right-3 w-1/6 min-w-[300p
      <ul>
        {GisStationsStaticDistrict.map(station => (
          <li key={station.id}>{station.name}</li>
        ))}
      </ul>
    </div>
  );
}

export default DataSheet;
```

Debugging

Nutzen Sie Recoil DevTools für eine effektive Zustandsverwaltung und Debugging. Diese Tools bieten Einb
in den Zustand Ihrer Anwendung und helfen bei der Fehlersuche.