Rakamin Academy

Redux State Management

Async Logic dan Data Fetching

Raih Mimpi #TanpaBatas

Outline Pembelajaran

Rakamin Academy

Async Logic dan Data Fetching

- [ ] Async Logic dan Data Fetching
- [ ] Redux Debugging

Objektif sesi

- Memahami dan mengimplementasikan Data Fetching
- Memahami proses Redux Debugging

Data Fetching using Redux: Preparation

Sebelum memulai mencoba Data Fetching dengan Redux, Anda perlu menyediakan beberapa dependensi dan mengatur konfigurasi awal.
Berikut adalah beberapa langkah yang perlu diambil:

1. Install Dependensi Redux: Anda perlu menginstal dependensi Redux dan React-Redux di proyek Anda. Pastikan untuk menjalankan perintah berikut:

```
npm install redux react-redux
```

Data Fetching using Redux: Preparation

2. Buat Store Redux: Di proyek React Native Anda, buat file **store.js** untuk mengkonfigurasi Redux store:

```
// store.js
import { createStore, applyMiddleware } from 'redux';
import thunk from 'redux-thunk'; // Untuk mendukung aksi asynchronous
import rootReducer from './reducers'; // Kustom kombinasi reducer

const store = createStore(rootReducer, applyMiddleware(thunk));

export default store;
```

Data Fetching using Redux: Preparation

3. Konfigurasi Root Reducer: Anda perlu membuat file reducers/index.js untuk menggabungkan semua reducer:

```
// reducers/index.js
import { combineReducers } from 'redux';
import { dataReducer } from './dataReducer';
import { formReducer } from './formReducer';
import { authReducer } from './authReducer';

const rootReducer = combineReducers({
  data: dataReducer,
  form: formReducer,
  auth: authReducer,
  // ... tambahkan reducer lain jika ada
});

export default rootReducer;
```

Data Fetching using Redux: Preparation

4. Integrasikan Redux Provider ke dalam Aplikasi: Di file tempat Anda menginisialisasi aplikasi React Native (biasanya index.js), impor **Provider** dari **react-redux** dan pasang store Redux ke dalamnya:

```
// index.js
import React from 'react';
import { AppRegistry } from 'react-native';
import { Provider } from 'react-redux';
import App from './App'; // Komponen utama aplikasi
import store from './store';

const ReduxApp = () => (
  <Provider store={store}>
    <App />
  </Provider>
);

AppRegistry.registerComponent('YourAppName', () => ReduxApp);
```

Data Fetching using Redux

```
// actions/dataActions.js
export const fetchDataSuccess = (data) => ({
  type: 'FETCH_DATA_SUCCESS',
  payload: data,
});

export const fetchDataError = (error) => ({
  type: 'FETCH_DATA_ERROR',
  payload: error,
});

export const fetchData = () => {
  return async (dispatch) => {
    try {
      const response = await fetch('https://api.example.com/data');
      const data = await response.json();

      dispatch(fetchDataSuccess(data));
    } catch (error) {
      dispatch(fetchDataError(error.message));
    }
  };
};
```

Data Fetching using Redux

```
// reducers/dataReducer.js
const initialState = {
  data: [],
  error: null,
};

export const dataReducer = (state = initialState, action) => {
  switch (action.type) {
    case 'FETCH_DATA_SUCCESS':
      return { ...state, data: action.payload, error: null };
    case 'FETCH_DATA_ERROR':
      return { ...state, error: action.payload };
    default:
      return state;
  }
};
```

Data Fetching using Redux

```
// components/DataComponent.js
import React, { useEffect } from 'react';
import { View, Text } from 'react-native';
import { useDispatch, useSelector } from 'react-redux';
import { fetchData } from '../actions/dataActions';

const DataComponent = () => {
  const dispatch = useDispatch();
  const data = useSelector((state) => state.data.data);
  const error = useSelector((state) => state.data.error);

  useEffect(() => {
    dispatch(fetchData());
  }, [dispatch]);

  return (
    <View>
      {error && <Text>Error: {error}</Text>}
      {data.length > 0 ? (
        <View>
          {data.map((item) => (
            <Text key={item.id}>{item.name}</Text>
          ))}
        </View>
      ) : (
        <Text>Loading data...</Text>
      )}
    </View>
  );
};

export default DataComponent;
```

Redux Debugging

Redux Debugging

Debugging redux menggunakan Reactotron, Berikut adalah langkah-langkah untuk penggunaan Reactotron :

Langkah 1: Instalasi Reactotron

- **Unduh Reactotron:**
  Kunjungi situs resmi Reactotron di https://github.com/infinitered/reactotron.
  Pilih versi yang sesuai dengan sistem operasi Anda (Windows, macOS, atau Linux) dan unduh installer atau arsip ZIP.
- **Instal Reactotron:**
  Ikuti petunjuk instalasi sesuai dengan sistem operasi yang Anda gunakan.

Redux Debugging

Langkah 2: Persiapkan Proyek React Native

- **Instal Reactotron Package**

```
npm install --save-dev reactotron-react-native
```

- **Instal Reactotron Core Package:** Jika Anda ingin menggunakan Reactotron untuk debugging Redux atau pengembangan React.

```
npm install --save-dev reactotron
```

Redux Debugging

Langkah 3: Konfigurasi React Native

- **Tambahkan Skrip di package.json:**
  Buka file package.json proyek Anda.
  Tambahkan script berikut di bagian "**scripts**":

```
"scripts": {
  "android": "react-native run-android",
  "ios": "react-native run-ios",
  "start": "react-native start",
  "reactotron": "reactotron"
}
```

Redux Debugging

Langkah 4: Gunakan Reactotron dalam Proyek Anda

- **Inisialisasi Reactotron di index.js:**

```
import Reactotron from 'reactotron';

if (__DEV__) {
  Reactotron.configure({ host: 'your-host-ip' }) // Gantilah 'your-hos
    .useReactNative() // Tambahkan plugin untuk React Native
    .connect(); // Hubungkan ke aplikasi desktop Reactotron

  // Opsional: Tambahkan plugin Redux dan lainnya
  Reactotron.use(require('reactotron-redux')()); // Untuk integrasi Re
  Reactotron.useReactNative({
    asyncStorage: { ignore: ['secret'] },
  });

  // Anda bisa menambahkan opsi konfigurasi tambahan di sini
}
```

- **Gunakan Reactotron di Komponen**

```
Reactotron.log('Hello Reactotron!');
```

Rakamin Academy

Referensi

Reactotron


Rakamin
Academy
Thank you
Raih Mimpi #TanpaBatas