Chapter 3: React Basics

Redux Logger and Async Thunk

Sesi 3

PERHATIAN

Jangan lupa untuk isi form absensi. Bagi instruktur mohon mengisi form absensi yang telah diinfokan dan konfirmasi nomor urut peserta yang hadir.

Untuk student harap mengisi form absensi **di sini** sebelum kelas dimulai. Untuk kode peserta dapat ditanyakan kepada instruktur dan jangan lupa mencantumkan urutan pertemuannya.

Redux Logger

Kita bisa menambahkan fungsi tambahan seperti extensions atau plugin diantara proses dispatch sebuah action hingga pada saat mencapai reducer. Kita bisa menggunakan Redux middleware untuk logging, crash reporting, proses async ke API, routing dan lain sebagainya.

Redux Logger

Sudah cukup jelas ya, sebuah logger untuk Redux. Setiap terjadi perubahan state, action di panggil dan berbagai hal yang berhubungan dengan Redux, akan muncul informasi di console browser.

1. Instalasi

```
> npm install redux-logger
```

1. Import dan implementasikan logger dalam store

Lakukan import ini di file tempat kita melakukan configureStore.

```
import { configureStore } from "@reduxjs/toolkit";
import usersReducer from '../features/users/usersSlice';
import logger from 'redux-logger'

export const store = configureStore({
    reducer: {
        users: usersReducer
    },
    middleware: (getDefaultMiddleware) => getDefaultMiddleware().concat(logger),
})
```

2. Run the application

Silakan langsung jalankan aplikasinya dan saksikan aksi dari redux-logger nya pada developer console di browser kita. Recommended untuk menggunakan Chrome atau Firefox

Async Thunk

Redux Toolkit menyediakan function createAsyncThunk untuk memproses async.

THUNK

- Nama ini sebenarnya secara humor merupakan bentuk past-tense dari “think”
- Adalah sebuah fungsi yang membungkus 1 atau beberapa expresi yang akan di evaluasi (baca: dikerjakan)
- Mengadopsi konsep thunk ke dalam redux

```
// calculation of 1 + 2 is immediate
// x === 3
let x = 1 + 2;

// calculation of 1 + 2 is delayed
// foo can be called later to perform the calculation
// foo is a thunk!
let foo = () => 1 + 2;
```

Introduction

Async Thunk

Kita langsung saja manfaatkan aplikasi kita untuk implementasi fetch data user dari API.

1. Install library axios

(*Asumsi melanjutkan dari aplikasi Counter. Silakan install redux toolkit dan react-redux, jika belum)

> npm install axios

2. Tambahkan users features

> mkdir -p src/features/users

> touch src/features/users/usersSlice.js

3. Daftarkan dalam app/store.js

Silakan diikuti dulu, kita akan tambahkan userReducer() nya

```
import { configureStore } from "@reduxjs/toolkit";
import usersReducer from '../features/users/usersSlice';

export const store = configureStore({
    reducer: {
        users: usersReducer
    }
})
```

JS usersSlice.js U

async-thunk > src > features > users > JS usersSlice.js > ...

```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import axios from "axios";

const USERS_URL = 'https://jsonplaceholder.typicode.com/users';

const initialState = {
    entities: [],
}

export const fetchUsers = createAsyncThunk('users/fetchUsers', async () => {
    const response = await axios.get(USERS_URL);
    return response.data
})

const usersSlice = createSlice({
    name: 'users',
    initialState,
    reducers: {},
    extraReducers(builder) {
        builder.addCase(fetchUsers.fulfilled, (state, action) => {
            state.entities.push(...action.payload)
        })
    }
})

export default usersSlice.reducer
```

Asynchronous

Thunk in Action

HACKTIV8

4. Membuat komponen src/features/UsersList.js

Buat komponen usersLists, seperti yang terlampir disamping.

5. Cantumkan UsersList dalam App.js

```
import './App.css';
import UsersList from "./features/users/UsersList"

function App() {
  return (
    <div className="App container">
     <UsersList />
    </div>
  );
}
```

```
import { useDispatch, useSelector } from "react-redux";
import { fetchUsers } from "./usersSlice";

const UsersList = () => {
    const allUsers = useSelector((state) => state.users.entities)
    const dispatch = useDispatch()

    const doFetchUsers = () => {
      dispatch(fetchUsers())
    }

    return (
      <div className="container">

        <h1>Users data</h1>
        <button className="btn btn-primary" onClick={doFetchUsers}>Get Users</button>
        <table className="table table-striped table-bordered">
          <thead className="table-dark">
            <tr>
              <th>Name</th>
              <th>Username</th>
              <th>Email</th>
            </tr>
          </thead>
          <tbody>
            {
            allUsers.map((user => (
                <tr key={user.id}>
                  <td>{user.name}</td>
                  <td>{user.username}</td>
                  <td>{user.email}</td>
                </tr>
              )))
            }
          </tbody>
        </table>
      </div>
    );
}
export default UsersList
```

Asynchronous

Thunk in Action

HACKTIV8

6. Update src/App.css

Tambahkan style seperti di kanan ini

```
11  table {
12    border-collapse: collapse;
13  }
14
15  th {
16    padding: 5px;
17    background-color: #3C5186;
18    color: #FFF5DE;
19  }
20
21  td {
22    padding: 5px;
23  }
```

Asynchronous

Thunk in Action

7. Jalankan aplikasi dan klik tombol Get Users

... ekspektasi kita adalah seperti tampilan di bawah ini

Asynchronous

Thunk in Action

HACKTIV8

Login with + React & Redux

Login biasanya merupakan halaman pertama yang harus kita buat di hampir semua aplikasi. Dalam sesi ini sekaligus review tentang react-redux di pembahasan sebelumnya, kita akan membuat form login yang interaktif dan intuitif dengan React & Redux.

Berikut beberapa fitur dari form login yang akan kita implementasi:

- Form memiliki field email dan kata sandi
- Ketika pengguna mengklik kirim, itu akan menampilkan pesan bahwasanya value email dan kata sandi itu dikirim sebagai permintaan ke server.
- Saat menerima respon, itu akan menunjukkan status login.

Prerequisite libraries

- **create-react-app**: I always use this library to create my React project, it will install React and some scripts with make our React app running
- **@reduxjs/toolkit**: our state management
- **react-redux**

Application state

Because we only have a login form, our application state is pretty simple, it has 3 attributes:

- **isLoginPending**: indicates when login has sent login request
- **isLoginSuccess**: indicates if login successful
- **errorMessage**: contains the error message if login fail

0. Provide store dalam index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

import { store } from './app/store';
import { Provider } from 'react-redux';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>
);
```

1. Tambahkan auth features

> mkdir -p src/features/auth

> touch src/features/users/authSlice.js

Buat dummy callLoginAPI. (dengan memasukkan kredensial secara hardcode)

Gunakan createAsyncThunk untuk fetching API Login yang dummy tersebut.

```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";

const initialState = {
  isLoginPending: false,
  isLoginSuccess: false,
  errorMessage: '',
  user: {}
}

function callLoginApi(email, password) {
  return new Promise(function(resolve, reject) {
    setTimeout(() => {
      if (email === 'admin@login.com' && password === 'admin') {
        resolve({email})
      } else {
        reject('Invalid email and password')
      }
    }, 1000)
  })
}

export const authLoginAPI = createAsyncThunk('auth/login', async ({email, password}) => {
  try {
    const response = await callLoginApi(email, password)
    return response.email
  } catch(err) {
    throw(err)
  }
})
```

2. Masih di authSlice

Tambahkan builder.addcase, untuk case pending, fulfilled dan rejected.

Bila sudah jangan untuk mendaftarkan authSlice ini ke dalam app/store.js.

```
import { configureStore } from "@reduxjs/toolkit";
import authReducer from '../features/auth/authSlice';
import logger from 'redux-logger'

export const store = configureStore({
    reducer: {
        auth: authReducer
    },
    middleware: (getDefaultMiddleware) => getDefaultMiddleware().concat(logger),
})
```

```
31  const authSlice = createSlice({
32      name: 'auth',
33      initialState,
34      reducers: {},
35      extraReducers(builder) {
36          builder
37          .addCase(authLoginAPI.pending, (state) => {
38            state.isLoginPending = true
39          })
40          .addCase(authLoginAPI.fulfilled, (state, action) => {
41            console.log('fulfilled')
42            console.log(action)
43            const { email } = action.payload
44            state.isLoginPending = false
45            state.isLoginSuccess = true
46            state.user = { email }
47          })
48          .addCase(authLoginAPI.rejected, (state, action) => {
49            console.log(action, "rejected")
50            state.isLoginPending = false
51            state.isLoginSuccess = false
52            state.errorMessage = action.error.message
53          })
54      }
55  })
56
57  export default authSlice.reducer
58
```

3. Buat komponen /src/features/AuthLoginForm

Persiapkan state dan method seperti disamping.

Method doSubmit digunakan untuk action form.

```
import { useState } from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { authLoginAPI } from './authSlice';

function AuthLoginForm() {
  const authState = useSelector((state) => state.auth)
  const dispatch = useDispatch()

  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')

  const emailChange = (event) => {
    setEmail(event.target.value)
  }

  const passwordChange = (event) => {
    setPassword(event.target.value)
  }

  const doSubmit = (event) => {
    event.preventDefault()
    dispatch(authLoginAPI({email, password}))
    setEmail('')
    setPassword('')
  }
```

4. Masih dalam komponen /src/features/AuthLoginForm

Dalam fungsi return, siapkan form login yang berisi email dan password. Beserta template ketika pending, sukses login dan error login.

```
  return (
    <form name="loginForm" onSubmit={doSubmit}>
      <div className="form-group-collection">
        <div className="form-group">
          <label>Email:</label>
          <input type="email" name="email" onChange={emailChange} value={email} />
        </div>

        <div className="form-group">
          <label>Password:</label>
          <input type="password" name="password" onChange={passwordChange} value={password} />
        </div>
      </div>

      <input type="submit" value="Login" />

      <div className="message">
        { authState.isLoginPending && <div><img src="https://mir-s3-cdn-cf.behance.net/project_modules/disp/35771931234507.564a1d2403b3a.gif" alt="Loading..." /></div> }
        { authState.isLoginSuccess && <div>Success.</div> }
        { authState.errorMessage && <div>{authState.errorMessage}</div> }
      </div>
    </form>
  );
}

export default AuthLoginForm;
```

5. Cantumkan komponen di App.js

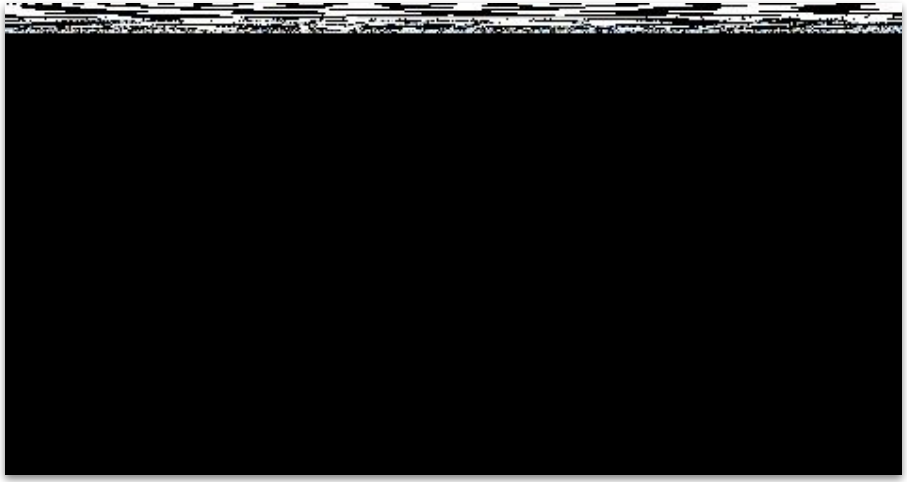
Import komponen AuthLoginForm yang sudah dibuat, dan render dalam fungsi return.

```
import './App.css';
import AuthLoginForm from "./features/auth/AuthLoginForm"

function App() {
  return (
    <div className="App">
      <AuthLoginForm />
    </div>
  );
}

export default App;
```

6. Update di App.css

```css
form[name="loginForm"] {
  display: flex;
  flex-direction: column;
  width: 450px;
  padding: 10px;
}

.form-group {
  display: flex;
  flex-direction: row;
  margin-bottom: 10px;
}

label {
  min-width: 180px;
  width: 180px;
  font-weight: bold;
  font-size: 25px;
}

input[type="email"], input[type="password"] {
  box-sizing: border-box;
  height: 40px;
  padding: 3px;
  flex-grow: 1;
  font-size: 25px;
}

input[type="submit"] {
  align-self: flex-end;
  width: 100px;
  height: 50px;
  border: 3px solid lightgray;
  border-radius: 5px;
  font-size: 20px;
  background: white;
}

.message {
  font-size: 25px;
  margin-top: 10px;
  text-align: right;
}
```

Silakan jalankan aplikasinya. Untuk melihat simulasi hasilnya, silakan lihat Lampiran 1



Implementation (Final)

Login with React & Redux

Thank You

PT Hacktivate Teknologi Indonesia
Gedung Aquarius Pondok Indah
Jalan Sultan Iskandar Muda No.7
Kebayoran Lama, Jakarta Selatan

https://hacktiv8.com

Hacktiv8id

Hacktiv8id

Hacktiv8

Hacktiv8 Indonesia