< What About Coding />

Home > React Js > **Build an application using createAsyncThunk and Redux Toolkit**

React Js

Build an application using createAsyncThunk and Redux Toolkit

By Himanshu Shekhar March 24, 2023

Write a Comment

Calling an API is not so straightforward while using the redux toolkit, and that's why redux has a middleware name "createAsyncThunk()" which provided us with all the superpowers needed for handling API and response.

In this tutorial, we are going to build a complete CRUD functionality, but using mockAPI, so that you get a complete end-to-end knowledge of how to deal with API while performing Creta, Read, Update, and Delete Operations.

Table of Contents

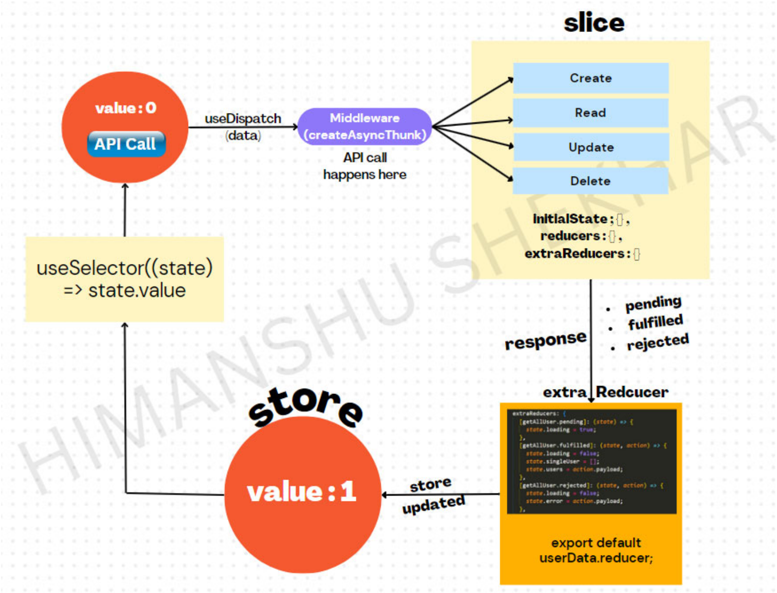
Pre-requisites

You must have an idea about redux beginner level, if not, learn from here

Basic knowlddge of HTML, CSS , Java Script , React and Redux

Diagram

Have a closer look on the diagram to understand the redux and asyncThunk flow



```
extraReducers: {
  [getAllUser.pending]: (state) => {
    state.loading = true;
  },
  [getAllUser.fulfilled]: (state, action) => {
    state.loading = false;
    state.singleUser = [];
    state.users = action.payload;
  },
  [getAllUser.rejected]: (state, action) => {
    state.loading = false;
    state.error = action.payload;
  },
```

Lets understand the flow –

Step 1 – An action is performed on the front-end (let's say a button click)

Step 2 – That action is dispatched (by using **useDispatch** hook) to the middleware "**createAsyncThunk**()" written inside slice file

Step 3 – Inside createAsyncThunk() an API is made, using fetch or Axios, depending upon the method ie. GET, POST, DELETE, OR PUT

Step 4 – Now the response from the above is handled by the **extraReducer** , written inside createSlice method

Step 5 – And finally the state (or the global **store**) is updated

Step 6 – The store data is displayed back to frontend using **useSelector** hook

Adding redux to the project

Well, I have explained this in-depth, in my preview article, do check it out. Click here

Install Package

```
npm install --save react-redux @reduxjs/toolkit
```

src/app/**store.js**

```
import { configureStore } from "@reduxjs/to
import gitUser from "../features/gitUserSlice

export const store = configureStore({
  reducer: {
    app: gitUser,
  },
});
```

Don’t forget to provide the store globally.

src/index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import { store } from "./app/store";
import { Provider } from "react-redux";
```

```

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

Let’s create Slice now

Slice is the only file that will contain all the things needed to perform our operation

- initialState
- reducers
- extraReducers

If you want a quick overview of how API call work using createAsyncthunk and extraReducers, do watch the below video

src/features/**getUserSlice.js**

```
import { createAsyncThunk, createSlice } frc

//Get all user action
export const getAllUser = createAsyncThunl
  "getUsers",
  async (args, { rejectWithValue }) => {
    try {
      const response = await fetch(
        "https://629f5d82461f8173e4e7db69.mo
      );
      const result = await response.json();
      return result;
    } catch (err) {
      return rejectWithValue("Opps found an e
    }
  }
);

//get single user
export const getSingleUser = createAsyncTh
  "getSingleUser",
  async (id, { rejectWithValue }) => {
    const response = await fetch(
      `https://629f5d82461f8173e4e7db69.moc
    );

    try {
      const result = await response.json();
      return result;
    } catch (err) {
      return rejectWithValue(err.message);
    }
  }
);
//create action
export const createUser = createAsyncThur
  "createUser",
  async (data, { rejectWithValue }) => {
    const response = await fetch(
      "https://629f5d82461f8173e4e7db69.moc
      {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify(data),
      }
    );
    const result = await response.json();
    return result;
  }
);

//delete single user
export const deleteUser = createAsyncThun
```

```
  "deleteUser",
  async (id, { rejectWithValue }) => {
    try {
      const response = await fetch(
        `https://629f5d82461f8173e4e7db69.mo
        {
          method: "DELETE",
        }
      );
      const result = await response.json();
      return result;
    } catch (err) {
      console.log(err);
      return rejectWithValue(err.response.dat
    }
  }
);

//update user
export const updateUser = createAsyncThu
  "updateUser",
  async ({ id, name, email, age, gender }, { r

    try {
      const response = await fetch(
        `https://629f5d82461f8173e4e7db69.mo
        {
          method: "PUT",
          headers: {
            "Content-Type": "application/json",
          },
          body: JSON.stringify({ name, email, ag
        }
      );
      const result = await response.json();
      return result;
    } catch (err) {
      return rejectWithValue(err);
    }
  }
);

export const gitUser = createSlice({
  name: "gitUser",
  initialState: {
    users: [],
    loading: false,
    error: null,
    searchData: [],
  },
  reducers: {
    searchUser: (state, action) => {
      state.searchData = action.payload;
    },
  },
  extraReducers: {
    [getAllUser.pending]: (state) => {
      state.loading = true;
    },
    [getAllUser.fulfilled]: (state, action) => {
      state.loading = false;
      state.singleUser = [];
      state.users = action.payload;
    },
    [getAllUser.rejected]: (state, action) => {
      state.loading = false;
      state.error = action.payload;
    },
    [createUser.fulfilled]: (state, action) => {
      state.loading = false;
      state.users.push(action.payload);
    },
    [deleteUser.pending]: (state) => {
      state.loading = true;
    },
    [deleteUser.fulfilled]: (state, action) => {
      state.loading = false;
      const { id } = action.payload;
      if (id) {
        state.users = state.users.filter((post) =>
      }
    },
    [deleteUser.rejected]: (state, action) => {
      state.loading = false;
      state.error = action.payload.message;
    },
    [getSingleUser.pending]: (state) => {
      state.loading = true;
    },
    [getSingleUser.fulfilled]: (state, action) =>
      state.loading = false;
      state.singleUser = [action.payload];
    },
    [getSingleUser.rejected]: (state, action) =
      state.loading = false;
      state.error = action.payload.message;
    },
    [updateUser.pending]: (state) => {
      state.loading = true;
    },
    [updateUser.fulfilled]: (state, action) => {
      console.log("updated user fulfilled", actio
      state.loading = false;
      state.users = state.users.map((ele) =>
        ele.id === action.payload.id ? action.pa
      );
    },
    [updateUser.rejected]: (state, action) =>
      state.loading = false;
      state.error = action.payload.message;
    },
  },
});

export const { searchUser } = gitUser.action
export default gitUser.reducer;
```

- calling an API (Create, Read, Update, Delete)
- returning an response
- error handling
- updating the store based on the response
- exporting the reducer to the store

Adding Frontend Now

App.js

```
import { BrowserRouter, Route, Routes } from
import "./App.css";
import Create from "./components/Create";
import Read from "./components/Read";
import Edit from "./components/Edit";
import Navbar from "./components/Navbar

function App() {
  return (
    <div className="App">
      <BrowserRouter>
        <Navbar />
        <Routes>
          <Route exact path="/" element={<Rea
          <Route exact path="/create" element=
          <Route exact path="/edit/:id" element
        </Routes>
      </BrowserRouter>
    </div>
  );
}

export default App;
```

Navbar.js

```
import React, { useState } from "react";
import { useDispatch, useSelector } from "re
import { Link } from "react-router-dom";
import { searchUser } from "../features/gitUs

const Navbar = () => {
  const [searchData, setSearchData] = useS
  const totalCount = useSelector((state) =>

  const dispatch = useDispatch();

  dispatch(searchUser(searchData));

  return (
    <>
      <nav class="navbar navbar-expand-lg r
        <div className="container-fluid">
          <div class="collapse navbar-collapse'
            <ul class="navbar-nav">
              <li class="nav-item">
                <Link to="/create" class="nav-link">
                  Create Post
                </Link>
              </li>
              <li class="nav-item">
                <Link to="/" class="nav-link">
                  All Post ({totalCount.length})
                </Link>
              </li>
            </ul>
          </div>

          <input
            class="form-control "
            type="search"
            placeholder="Search"
            value={searchData}
            onChange={(e) =>
              dispatch(searchUser(setSearchDat
            }
          ></input>
        </div>
      </nav>
    </>
  );
};

export default Navbar;
```

Create.js

```
import React, { useState } from "react";
import { useDispatch } from "react-redux";
import { useNavigate } from "react-router-
import { createUser } from "../features/gitUs

const Create = () => {
  const [data, setData] = useState({});
  const dispatch = useDispatch();
  const navigate = useNavigate();

  const updateData = (e) => {
    setData({
      ...data,
      [e.target.name]: e.target.value,
    });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("user data...", data);
    dispatch(createUser(data));
    navigate("/");
  };

  return (
    <div>
      <h2>Enter the data</h2>

      <form onSubmit={handleSubmit}>
        <div>
          <input
            type="text"
            name="name"
            placeholder="enter name"
            onChange={updateData}
          />
        </div>
        <div>
          <input
            type="email"
            name="email"
            placeholder="enter email"
            onChange={updateData}
          />
        </div>
        <div>
          <input
            type="number"
            name="age"
            placeholder="enter age"
            onChange={updateData}
          />
        </div>
        <div>
          <input
            type="radio"
            name="gender"
            // checked={updateData.gender ==
            value="Male"
            onChange={updateData}
          />
          <label>Male</label>
          <input
            type="radio"
            name="gender"
            // checked={this.state.selectedOptio
            value="Female"
            onChange={updateData}
          />
          <label>Famale</label>
        </div>
        <div>
          <button type="submit">Submit</butto
        </div>
      </form>
    </div>
  );
};

export default Create;
```

```
import React, { useEffect, useState } from "re
import { useSelector, useDispatch } from "re
import { getAllUser, deleteUser } from "../fea
import { Link } from "react-router-dom";
import UserModal from "./UserModal";

const Read = () => {
  const dispatch = useDispatch();
  const [show, setShow] = useState(false);
  const handleClose = () => setShow(false);
  const handleShow = () => setShow(true);
  const [radioCheck, setRadioCheck] = useS

  const [id, setId] = useState();

  const data = useSelector((state) => {
    return state.app;
  });

  console.log("radio...", radioCheck);

  useEffect(() => {
    dispatch(getAllUser());
    // eslint-disable-next-line react-hooks/e
  }, []);

  if (data.loading) {
    return <h2>Loading...</h2>;
  }

  if (data.error != null) {
    return <h3>{data.error}</h3>;
  }

  console.log("final data to loop", data);

  return (
      <div>
        <UserModal
          handleShow={handleShow}
          handleClose={handleClose}
          show={show}
          setShow={setShow}
          id={id}
        />
        <div className="d-flex justify-content-
          <h1>All Users</h1>
          <div className="d-flex gap-2">
            <div>
              <input
                class="form-check-input"
                type="radio"
                name="gender"
                checked={radioCheck === ""}
                onChange={(e) => setRadioCheck("
              />
              <label class="form-check-label">All<
            </div>
            <div class="form-check">
              <input
                class="form-check-input"
                type="radio"
                name="gender"
                value="Male"
                checked={radioCheck === "Male"}
                onChange={(e) => setRadioCheck(
              />
              <label class="form-check-label">Mal
            </div>
            <div>
              <input
                class="form-check-input"
                type="radio"
                name="gender"
                value="Female"
                checked={radioCheck === "Female
                onChange={(e) => setRadioCheck(
              />

              <label class="form-check-label">Fem
            </div>
          </div>
        </div>

        {data?.users
```

```
     .filter((item) => {
      if (data.searchData.length === 0) {
       return item;
      } else {
       return item.name
        .toLowerCase()
        .includes(data.searchData.toLowerC
      }
     })
     .filter((item) => {
      if (radioCheck === "") {
       return item;
      } else if (radioCheck === "Male") {
       return item.gender === radioCheck;
      } else if (radioCheck === "Female") {
       return item.gender === radioCheck;
      }
     })
     .map((ele) => (
      <div key={ele.id} className="card w-
       <div className="card-body">
        <h5 className="card-title">{ele.na
        <h6 className="card-subtitle mb-
        <h6 className="card-subtitle mb-

        <button
         type="button"
         class="btn btn-primary"
         //onClick={() => setId(ele.id) && h
         onClick={() => [setId(ele.id), handl
        >
         View
        </button>

        <Link
         onClick={() => dispatch(deleteUse
         className="card-link mx-2"
        >
         Delete
        </Link>
        <Link to={`/edit/${ele.id}`}>
         <span className="card-link mx-2
        </Link>
       </div>
      </div>
     ))}
   </div>
  );
```

```
};

export default Read;
```

```
import React, { useEffect, useState } from "re
import { useDispatch, useSelector } from "re
import { useParams, useNavigate } from "re
import { updateUser } from "../features/gitU

const Edit = () => {
  const dispatch = useDispatch();
  const { id } = useParams();
  const navigate = useNavigate();
  const initialState = {
    name: "",
    email: "",
    age: "",
    gender: "",
  };
  const [updatedData, setUpdatedData] = u

  //get all data
  const { users, loading } = useSelector((sta

  useEffect(() => {
    //retrieving single data from user list
    if (id) {
      const singleData = users.find((user) => u
      console.log("singledata preload on edit
      setUpdatedData({ ...singleData });
    }
  }, []);

  //updating state as use changes input fiel
  const newData = (e) => {
    setUpdatedData({ ...updatedData, [e.targ
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    console.log("update data..", updatedData
    dispatch(updateUser(updatedData));
    setUpdatedData(initialState);
    navigate("/");
  };

  if (loading) {
    return <h2>Loading..</h2>;
  }

  return (
    <div>
      <h2>Update the data</h2>

      {updatedData && (
        <form onSubmit={handleSubmit}>
          <div>
            <input
              type="text"
              name="name"
              placeholder="enter name"
              value={updatedData.name}
              onChange={newData}
            />
          </div>
          <div>
            <input
              type="email"
              name="email"
              placeholder="enter email"
              value={updatedData.email}
              onChange={newData}
            />
          </div>
          <div>
            <input
              type="number"
              name="age"
              placeholder="enter age"
              value={updatedData.age}
              onChange={newData}
            />
          </div>
          <div>
            <input
              type="radio"
              name="gender"
              checked={updatedData.gender ==
              value="Male"
              onChange={newData}
            />
            <label>Male</label>
            <input
```

```
              type="radio"
              name="gender"
              checked={updatedData.gender ==
              value="Female"
              onChange={newData}
            />
            <label>Famale</label>
          </div>
          <div>
            <button type="submit">Submit</butt
          </div>
        </form>
      )}
    </div>
  );
};

export default Edit;
```

While working on any real-life project, you have to deal with API and if you are working on redux, the middleware will definitely come into the picture.

The combination of createAsyncThunk and extraReducers makes API handling easy and simple.

Frankly, it more of a process, than the logic, so once you are comfortable with the flow, it will be a piece of cake for you

I have tried to break it down and explain you as simply as I can. Don't forget to watch my tutorial on the same.

Help Others

DOM and Window Object in JavaScript | JavaScript Interview series

Previous

Himanshu Shekhar

Hey. I am a software engineer with 3+ years. Along with my job, I create content regarding coding and software engineering in general. I would request you to follow my blog and my video, which will help you in your software engineering journey. Follow me on Youtube →

View All Articles

Related Posts

useRef in React Js | React Js for Beginners

November 23, 2022 React Hooks React Js

Revising React Basics (Fast)

June 1, 2022 React Js

Redux Toolkit Tutorial

December 9, 2022 React Js

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

□ Save my name, email, and website in this browser for the next time I comment.

Post Comment

Seach this blog...

Recent Posts

Build an application using createAsyncThunk and Redux Toolkit

DOM and Window Object in JavaScript | JavaScript Interview series

Build a simple MERN Stack Application | CRUD using

“this” keyword | JavaScript Interview Series

Scope , Scope chain and Lexical Scope | JavaScript Interview Series

Category

Frontend Interview Question	(5)
Java Script	(10)
JavaScript Interview Series	(5)
Jquery	(4)
Node JS	(3)
React Basics	(2)
React Hooks	(2)
React Js	(11)
Sales and Offers	(2)

Help others

Privacy Poilicy Sitemap

Terms and Conditions