[Harshavrdhana (Harshavardhana U N) (github.com)](https://github.com/Harshavrdhana)

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Mastering Redux Toolkit and RTK Query\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**1. What is Redux?**

* Redux is an open-source JavaScript library for managing the state of an application in a predictable way.
* It is commonly used with libraries like React for building user interfaces but can be used with any JavaScript framework or library.

**2. What is Redux Toolkit?**

* The official, opinionated, batteries-included toolset for efficient Redux development.
* Toolkit makes the Redux Easy to work with.

**3. Course Setup.**

**Install Redux Toolkit and React-Redux In React Application**

* npm install @reduxjs/toolkit react-redux

**4. What is Store? How to create one?**

A store is a central place where the state of our application is stored & it can be created using the **configureStore** function, and it holds the entire state tree of our application.

// app/store.js

import { configureStore } from "@reduxjs/toolkit";

//Creating Store

export const store = configureStore({

  reducer:{}

})

// main.js

import ReactDOM from "react-dom/client";

import App from "./App.jsx";

import "./index.css";

import { store } from "./app/store.js";

import { Provider } from "react-redux";

ReactDOM.createRoot(document.getElementById("root")).render(

  <Provider store={store}>

    <App />

  </Provider>

);

**5. What is Slice? How to create one?**

A slice is a **piece of store state** and the corresponding reducer logic to update that state. Slices are a way to organize our Redux store by breaking it down into smaller, more manageable parts.

* **InitialState:** InitialState as the name suggests is going to be the ‘initial’ data of our state.
  + ex: {value = 0}
* **Reducers:** Reducers are like the instructions on what to do with each slice.
  + They define how the information in a particular slice can be changed or updated.
* **Actions:** Actions are like the requests and commands you give the slice of the cake.
  + Ex: Increment , Decrement

// app/features/counter/counterSlice.js

import { createSlice } from "@reduxjs/toolkit";

export const counterSlice = createSlice({

  name: "counter",

  initialState: { value: 0 },

  reducers: {

    //immer library: we can update state directly

    increment: (state) => {

      state.value += 1;

    },

    decrement: (state) => {

      state.value -= 1;

    },

  },

});

export const { increment, decrement } = counterSlice.actions;

export default counterSlice.reducer;

* Add reducer to our store

// app/store.js

import { configureStore } from "@reduxjs/toolkit";

import counterReducer from './features/counter/counterSlice.js'

export const store = configureStore({

  reducer:{

    counter: counterReducer,

  }

})

**6. Using Redux Toolkit in React.**

* **UseSelector() Hook:** useSelector hook allows us to read the data from the redux store.
* **useDispatch() Hook:** The dispatch hook is used to send the actions to the store, triggering updates to the application state. In simpler terms, it lets you change the state in your Redux store from within your React components.

// components/Counter.jsx

import { useSelector, useDispatch } from "react-redux";

import { increment, decrement } from "../app/features/counter/counterSlice.js";

const Counter = () => {

  //Read the data from the store

  const count = useSelector((state) => state.counter.value);

  //Changing the data by sending "actions" to store

  const dispatch = useDispatch();

  return (

    <div>

      <h1>{count}</h1>

      <button onClick={()=>dispatch(increment())}>+</button>

      <button onClick={()=>dispatch(decrement())}>-</button>

    </div>

  );

};

export default Counter;

import Counter from "./components/Counter"

const App = () => {

  return (

    <div>

      <Counter/>

    </div>

  )

}

export default App

**7. What is RTK Query? RTK Query Setup.**

Redux Toolkit Query (RTK) is specifically designed to simplify data fetching, caching, and state management for API calls in a React and Redux application.

**Install Redux Toolkit and React-Redux In React Application**

* npm install @reduxjs/toolkit react-redux

**8. Creating Service in RTK Query and Creating End Points, RTK Query Hooks Magic.**

import { createApi, fetchBaseQuery } from "@reduxjs/toolkit/query/react";

export const productsApi = createApi({

  reducerPath: "products",

  baseQuery: fetchBaseQuery({ baseUrl: "https://dummyjson.com" }),

  endpoints: (builder) => ({

    //Get All Products

    getAllProduct: builder.query({

      query: () => "/products",

    }),

  }),

});

// `use${ GetAllProduct} Query`

export const {useGetAllProductQuery} = productsApi;

* **Register Service In Store.**

import { configureStore } from "@reduxjs/toolkit";

import { setupListeners } from "@reduxjs/toolkit/query";

import { productsApi } from "./service/dummyData";

export const store = configureStore({

  reducer: {

    [productsApi.reducerPath]: productsApi.reducer

  },

  //Only helping you with the caching

  middleware:(buildGetDefaultMiddleware)=> buildGetDefaultMiddleware().concat(productsApi.middleware),

});

setupListeners(store.dispatch)

* **Get All Products**

// components/AllProducts.jsx

import React from "react";

import { useGetAllProductQuery } from "../app/service/dummyData";

const AllProducts = () => {

  const { data, isError, isLoading } = useGetAllProductQuery();

  if (isError) {

    return <h1>OhoooNoooo We got an error</h1>;

  }

  if (isLoading) {

    return <h1>Loading...</h1>;

  }

  return (

    <div>

      {data?.products.map((p) => (

        <div key={p.id}>

          <h1>{p.title}</h1>

          <p>{p.description}</p>

        </div>

      ))}

    </div>

  );

};

export default AllProducts;

* **Get product by using ID.**

import { createApi, fetchBaseQuery } from "@reduxjs/toolkit/query/react";

export const productsApi = createApi({

  reducerPath: "products",

  baseQuery: fetchBaseQuery({ baseUrl: "https://dummyjson.com" }),

  endpoints: (builder) => ({

    //Get All Products

    getAllProduct: builder.query({

      query: () => "/products",

    }),

    // Get product by Id

    getProductById: builder.query({

      query: (id) => `/product/${id}`,

    }),

  }),

});

export const { useGetAllProductQuery, useGetProductByIdQuery } = productsApi;

// /components/SpecificProductById.jsx

import React from 'react'

import { useGetProductByIdQuery } from '../app/service/dummyData'

const SpecificProductById = () => {

  const {data, isError, isLoading} = useGetProductByIdQuery(2)

  if (isError) {

    return <h1>OhoooNoooo We got an error</h1>;

  }

  if (isLoading) {

    return <h1>Loading...</h1>;

  }

  return (

    <div>

      <h1>{data?.brand}</h1>

      <h1>{data?.category}</h1>

      <h1>{data?.description}</h1>

    </div>

  );

}

export default SpecificProductById ;

* **Add New Product.**

// service/dummyData.js

import { createApi, fetchBaseQuery } from "@reduxjs/toolkit/query/react";

export const productsApi = createApi({

  reducerPath: "products",

  baseQuery: fetchBaseQuery({ baseUrl: "https://dummyjson.com" }),

  endpoints: (builder) => ({

    //Get All Products

    getAllProduct: builder.query({

      query: () => "/products",

    }),

    // Get product by Id

    getProductById: builder.query({

      query: (id) => `/product/${id}`,

    }),

    //Add new Product

    addNewProduct: builder.mutation({

      query: (newProduct) => ({

        url: `/products/add`,

        method: "POST",

        headers: { "content-Type": "application/json" },

        body: newProduct,

      }),

    }),

  }),

});

// `use${ GetAllProduct} Query`

export const {

  useGetAllProductQuery,

  useGetProductByIdQuery,

  useAddNewProductMutation,

} = productsApi;

// components/AddNewProduct.jsx

import React from "react";

import { useAddNewProductMutation } from "../app/service/dummyData";

const AddNewProduct = () => {

  const [addNewProduct, { data, error, isLoading }] =

    useAddNewProductMutation();

  if (error) {

    return <h1>Error....!</h1>;

  }

  if (isLoading) {

    return <h1>Loading.......!</h1>;

  }

  const handleAddProduct = async () => {

    try {

      const newProductData = {

        id: 1,

        title: "Amazing T-shirt",

        description: "This is the best T shirt from One 8 brand",

      };

      await addNewProduct(newProductData);

    } catch (error) {

      console.error("Error adding new product", error);

    }

  };

  return (

    <div>

      <h1>{data?.id}</h1>

      <h1>{data?.title}</h1>

      <h1>{data?.description}</h1>

      <button onClick={handleAddProduct} disabled={isLoading}>

        Add New Product

      </button>

    </div>

  );

};

export default AddNewProduct;

* **Update Specific Product.**

// dummyData.js

//Update specific product

    updateProduct: builder.mutation({

      query: ({id, updatedProduct})=>({

        url:`/products/${id}`,

        method:"PUT",

        headers:{'Content-Type': "application/json"},

        body: updatedProduct

      })

    })

// components/UpdateProduct.js

import React from 'react'

import { useUpdateProductMutation } from '../app/service/dummyData'

const UpdateProduct = (prop) => {

  const [updateProduct, {data,error,isLoading}]= useUpdateProductMutation();

  if(error){

    <h1>Error.......!</h1>

  }

  if(isLoading){

    <h1>Loading.......!</h1>

  }

  const handleUpdateProduct = async()=>{

    try{

      const updatedProductData={

        title: "Title updated",

      };

      await updateProduct({

        id: prop.productId,

        updatedProduct: updatedProductData

      })

    }catch(error){

      console.error("Error updating product:", error );

    }

  }

  return (

    <div>

      <h1>{data?.title}</h1>

      <button onClick={handleUpdateProduct} disabled={isLoading}> Update Product</button>

    </div>

  )

}

export default UpdateProduct

* **Delete Product.**

//Delete Product

    deleteProduct: builder.mutation({

      query: (id) => ({

        url: `/products/${id}`,

        method: "DELETE",

      }),

    }),

import React from "react";

import { useDeleteProductMutation } from "../app/service/dummyData";

const DeleteProduct = (prop ) => {

  const [deleteProduct, { data, isError, isLoading }] =

    useDeleteProductMutation();

    if (isError) {

      return <h1>OhoooNoooo We got an error</h1>;

    }

    if (isLoading) {

      return <h1>Loading...</h1>;

    }

    const handleDeleteProduct=async()=>{

      try{

        await deleteProduct(prop.productId)

      }catch(error){

        console.error("Error deleting product:", error)

      }

    }

  return (

    <div>

      <h1>{data?.title? `${data.title} successfully deleted`: ""}</h1>

      <button onClick={handleDeleteProduct}>Delete Product</button>

    </div>

  );

};

export default DeleteProduct;