# 配置路由：

## 配置pages

## 配置路由(router config)

const routes: RouteObject[] = [

  {

    path: "/",

    element: <Home />,

    children: [],

  },

]

export default routes;

## 引入Browser Router

* 1. 將APP內使用useRoutes 初始化路由

import { useRoutes } from "react-router-dom";  
import routes from "./routes";

const App: React.FC = () => {

  const element = useRoutes(routes);

  return element;

};

export default App;

* 1. 在index.tsx中包裹

import React from "react";

import ReactDOM from "react-dom";

import { BrowserRouter } from "react-router-dom";

import "./index.css";

import App from "./App";

ReactDOM.render(

  <React.StrictMode>

    <BrowserRouter>

      <App />

    </BrowserRouter>

  </React.StrictMode>,

  document.getElementById("root")

);

# 串接API：

## Mark Data：建立假後端與資料做測試

Json-server：

* 啟動：npm run server
* Package.json

"scripts": {

    …

    "server": "json-server --watch db.json --port 3004 -d 1500"

…

},

--watch 檔案名 => 要啟動哪隻檔案的內容

--port 3004 => 將服務架設在哪個port 上面

-d 1500 => 延遲1.5s

* 更多資訊可以google json-server 進GitHub看

## 步驟：

1. 建立services(API方法的位置)

export const homeApi = createApi({

    // API的名稱

    reducerPath: "homeApi",

    // 連結的API基本路徑

    baseQuery: fetchBaseQuery({ baseUrl: "http://localhost:3004/" }),

    // 方法

    endpoints: (builder) => ({

        // query 需要傳兩個泛型的結果

        // 1. IGStory：response 的結果

        // 2. number | "all"：獲取資料時要使用的參數

        getIGPosts: builder.query<IGPost[], number | "all">({

            query: (id) => {

                if (id !== "all") {

                    return `posts/${id}`;

                }

                return "posts";

            },

        }),

        getIGStories: builder.query<IGStory[], number | "all">({

            query: (id) => {

                if (id !== "all") {

                    return `stories/${id}`;

                }

                return "stories";

            },

        }),

    }),

});

// Export hooks for usage in functional components, which are

// auto-generated based on the defined endpoints

export const { useGetIGPostsQuery, useGetIGStoriesQuery } = homeApi;

export const homeApiReducer = homeApi.reducer

1. 建立store

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

import { homeApi, homeApiReducer } from "services/homeServices";

export const store = configureStore({

    reducer: {

        [homeApi.reducerPath]:homeApiReducer,

    },

    // 將API的service綁定到store上

    middleware:(getDefaultMiddleware) =>

        getDefaultMiddleware().concat(homeApi.middleware),

});

// 以 Type 輸出 hook

export type RootState = ReturnType<typeof store.getState>;

export type AppDispatch = typeof store.dispatch;

export default store;

1. 在根目錄(index.tsx)引用Provider與store

ReactDOM.render(

  <React.StrictMode>

    <Provider store={store}>

      <BrowserRouter>

        <App />

      </BrowserRouter>

    </Provider>

  </React.StrictMode>,

  document.getElementById("root")

);

1. 使用hook去連接後端資料

const IGStory: React.FC = () => {

    // 在想要串連的組件中使用 Hook 拿到資料

    const { data, isLoading } = useGetIGStoriesQuery("all");

// 後續只要使用 data 就可以取得後端資料

# Redux

1. 建立Slice

export const friendSlice = createSlice({

    // Slice的名稱

    name: "friendsList",

    // 初始值

    initialState,

    // reducer 的動作

    reducers: {

        // 回傳兩個參數

        // 1. state = initialState 內的資料

        // 2. action:PayloadAction<number> = 動作取得的資料

        follow: (state, action: PayloadAction<number>) => {

            const friends = state.friends;

            for (let i = 0; i < friends.length; i++) {

                const friend = friends[i];

                if (friend.id === action.payload) {

                    friend.isFollowing = true;

                }

            }

        },

        unFollow: (state, action: PayloadAction<number>) => {

            const friends = state.friends;

            for (let i = 0; i < friends.length; i++) {

                const friend = friends[i];

                if (friend.id === action.payload) {

                    friend.isFollowing = false;

                }

            }

        },

    },

});

// 建立組件內要使用的 function (即 reducer:{...} 中的內容)

export const { follow, unFollow } = friendSlice.actions;

// 導出 slice 的 Reducer

export default friendSlice.reducer;

1. 在store中接入reducer

export const store = configureStore({

    reducer: {

        [homeApi.reducerPath]:homeApiReducer,

        friendReducer,

    },

    // 將API的service綁定到store上

    middleware:(getDefaultMiddleware) =>

        getDefaultMiddleware().concat(homeApi.middleware),

});

// 以 Type 輸出 hook

export type RootState = ReturnType<typeof store.getState>;

export type AppDispatch = typeof store.dispatch;

export default store;

1. 建立共用的Hook(必須)

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

import { AppDispatch, RootState } from "store";

// 更改 Data 內容時用

export const useAppDispatch = () => useDispatch<AppDispatch>();

// 獲取 Data 用

export const useAppSelector: TypedUseSelectorHook<RootState> = useSelector;