Exercise 20: Decoupling Route

Objectives and Outcomes

Decoupling route declarations can help organize your codebase and make it more maintainable. This exercise is a demo that showcases how to decouple route declarations in React using react-router-dom.

Exercise

Create a new file called routes.js

import React from 'react';

import { Route } from 'react-router-dom';

export const routes = [

{

path: '/',

component: () => <h1>Home Page</h1>,

exact: true,

},

{

path: '/products',

component: () => <h1>Products Page</h1>,

},

{

path: '/about',

component: () => <h1>About Page</h1>,

},

{

path: '/contact',

component: () => <h1>Contact Page</h1>,

},

{

path: '/users/:userId?',

component: ({ params }) => (

<h1>User Profile: {params.userId}</h1>

),

},

];

export const renderRoutes = () => {

return routes.map((route, index) => (

<Route

key={index}

path={route.path}

element={route.component}

exact={route.exact}

/>

));

};

In this file, we define an array of route objects called routes. Each route object contains the path for the route, the component to render, and an optional exact property to indicate an exact match.

We also export a function called renderRoutes that maps over the routes array and generates a Route component for each route. The path, element, and exact props are passed to the Route component using the properties of each route object.

Now, let's update our App component to use the decoupled route declarations:

import React from 'react';

import { BrowserRouter as Router, Routes } from 'react-router-dom';

import { renderRoutes } from './routes';

import NavigationMenu from './NavigationMenu';

const App = () => {

return (

<Router>

<div>

<NavigationMenu />

<Routes>

{renderRoutes()}

</Routes>

</div>

</Router>

);

};

export default App;

In this updated App component, we import the renderRoutes function from the routes.js file. We use the Routes component and call renderRoutes() to render the routes dynamically. This separates the route declarations from the main component, making it easier to manage and maintain.

Conclusion

This demo showcases how to decouple route declarations in React using react-router-dom. It helps organize your codebase and makes it more maintainable.

By using React Router, you can create a seamless navigation experience within your React application, rendering different components based on the current URL. It helps in organizing your application into distinct views/pages and provides a clean and structured way to handle routing.