DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Front End III

Enrutando la aplicación y agregando las rutas de navegación

Hasta acá tenemos seis componentes que se renderizarán en solo cuatro páginas porque tendremos dos páginas anidadas.

Enrutando la aplicación

Ahora escribiremos nuestro componente principal que es el que contendrá a la aplicación. Será otro componente funcional, al que llamaremos **<App />**.

En las primeras líneas importamos la biblioteca React, luego nuestros componentes que funcionarán como páginas, y finalmente los componentes de React Router: BrowserRouter, Route, Switch y Link. La aplicación debe estar contenida dentro de BrowserRouter.

Veamos la estructura inicial de nuestro **componente <App />**:

App.jsx

```
import React from 'react';
import Home from './Home.jsx';
import Main from './Main.jsx';
import About from './About.jsx';
import Faq from './Faq.jsx';
import { BrowserRouter, Route, Switch, Link } from 'react-router-dom';


function App () {
       return (
           <BrowserRouter>
               {null}
           </BrowserRouter>
       );
}
export default App;
```

React Router requiere que la aplicación esté contenida o bien dentro del componente llamado <BrowserRouter /> o dentro de un componente llamado <HashRouter />. Aunque es posible utilizar el componente <HashRouter /> en lugar de <BrowserRouter />, a menos que haya alguna razón muy particular, lo mejor es trabajar con BrowserRouter.

La principal diferencia entre ellos son las URLs que crean:

- BrowserRouter usa la API del historial de HTML5 (pushState, replaceState y popstate event. Es preferible porque crea URLs ordinarias, como por ejemplo: http://localhost/8080/main.
- HashRouter usa la parte del hash de la URL. Creará URLs con hashes, como por ejemplo: http://localhost/8080/#/main.

Ahora pasemos a escribir la **lógica de la aplicación**. Como mencionamos más arriba,

React Router requiere que metamos la lógica de la aplicación dentro de un par <BrowserRouter></BrowserRouter>. Además, debemos encerrar cada componente que representa a una página dentro de un componente Route.

Cada componente Route tiene un atributo llamado path, y cuando el atributo path coincide con la URL de la página entonces se renderiza el componente que está adentro.

Esto es lo que pondremos dentro de BrowserRouter por ahora:

```
<BrowserRouter>
    <Route exact path="/"><Home /></Route>
    <Route path="/main"><Main /></Route>
    <Route path="/about"><About /></Route>
    <Route path="/faq"><Faq /></Route>
</BrowserRouter>
```

Notemos que en el código de arriba hemos utilizado el atributo exact. **El atributo exact le indica a Route que la URL debe coincidir exactamente con el path**. Por ejemplo, en el código de arriba, de no utilizar exact, se renderizará al mismo tiempo tanto <Home /> como el componente de la página que estemos visitando porque todos empiezan con "/". Así, si la URL fuera http://localhost/8080/about, la aplicación renderizaría tanto Home como About al mismo tiempo.

Es importante entender que solo porque una URL coincidió con el path de un Route, no significa que los otros Route cuyos atributos paths coincidan también con la URL dejarán de renderizar a sus componentes. Recordemos esto: **Route siempre renderizará algo, ya sea un componente, si la URL coincide con su path, o null**. Debemos pensar en los componentes Route anidados de la misma forma en que pensamos en otros componentes anidados en React.

Veamos otro componente esencial para el manejo correcto de rutas con React Router: el **componente Switch**. Como ya sabemos, cuando una URL coincide con el path de algún componente Route, el componente encerrado por Route será renderizado. Sin embargo, también sabemos que si no usamos exact, podríamos renderizar más de un componente al mismo tiempo. Pero ¿podemos resolver esto siempre con exact? La respuesta es que tendremos casos en los que no podremos usar exact para obtener el resultado correcto. Acá es donde el componente Switch es esencial.

Switch garantiza que solo el primer componente cuyo path coincida con la URL será renderizado, mientras que todos los demás cuyos paths también coincidan con la URL —si los hubiera— serán ignorados.
La forma de usar Switch es envolviendo a los componentes Route dentro de un par <Switch></Switch>:

```
<BrowserRouter>
    <Switch>
        <Route exact path="/"><Home /></Route>
        <Route path="/main"><Main /></Route>
        <Route path="/about"><About /></Route>
        <Route path="/faq"><Faq /></Route>
    </Switch>
</BrowserRouter>
```

Nuestro componente App ahora es completamente funcional, y luce así:

App.jsx

```
import React from 'react';
import Home from './Home.jsx';
import Main from './Main.jsx';
import About from './About.jsx';
import Faq from './Faq.jsx';
import { BrowserRouter, Switch, Route, Link } from 'react-router-dom';
```

```
function App () {
       return (
           <BrowserRouter>
               <Switch>
                   <Route exact path="/"><Home /></Route>
                   <Route path="/main"><Main /></Route>
                   <Route path="/about"><About /></Route>
                   <Route path="/faq"><Faq /></Route>
               </Switch>
           </BrowserRouter>
       );
}
export default App;
```

Agregando las rutas de navegación

Si bien nuestro código está completo desde el punto de vista funcional y ya podemos navegar a través de la barra de direcciones, aún podríamos facilitar la navegación agregando vínculos (links) a las páginas que creamos.

Normalmente en una aplicación multipágina usaríamos los componentes ancla (<a>) con su atributo href apuntando a la URL, pero esto nos resulta inconveniente para una SPA porque las anclas disparan una recarga de página por defecto. En su lugar, utilizaremos el componente **Link** y su atributo **to** que React Router introdujo para estos casos.

Utilizaremos una lista desordenada (
), ítems de listas (
), y el componente Link de React Router para crear los vínculos a las páginas de esta manera:

```
               <ul>
                   <li>
                       <Link to="/">Home</Link>
                   </li>
                   <li>
                       <Link to="/main">Main</Link>
                   </li>
```

```
                <li>
                    <Link to="/about">About</Link>
                </li>
                <li>
                    <Link to="/faq">FAQ</Link>
                </li>
            </ul>
```

Nuestro código ahora luce así:

App.jsx

```
import React from 'react';
import Home from './Home.jsx';
import Main from './Main.jsx';
import About from './About.jsx';
import Faq from './Faq.jsx';
import { BrowserRouter, Switch, Route, Link } from 'react-router-dom';

function App () {
       return (
           <BrowserRouter>
               <ul>
                   <li>
                       <Link to="/">Home</Link>
                   </li>
                   <li>
                       <Link to="/main">Main</Link>
                   </li>
                   <li>
                       <Link to="/about">About</Link>
                   </li>
                   <li>
                       <Link to="/faq">FAQ</Link>
                   </li>
               </ul>
```

```
            <Switch>
                <Route exact path="/"><Home /></Route>
                <Route path="/main"><Main /></Route>
                <Route path="/about"><About /></Route>
                <Route path="/faq"><Faq /></Route>
            </Switch>
        </BrowserRouter>
    );
}
export default App;
```

Ahora tenemos links de navegación en nuestras páginas.

¡Hasta la próxima!