Usa React Router en tus aplicaciones de Reactjs

juandc 37531

React Router es una herramienta que nos permite configurar **rutas dinámicas**; para así decidir qué parte de nuestra aplicación se mostrara a los usuario. En este tutorial crearemos la aplicación para una **tienda de libros**.

> *Components are the heart of React's powerful, declarative programming model. React Router is a collection of navigational components that compose declaratively with your application. Whether you want to have bookmarkable URLs for your web app or a composable way to navigate in React Native, React Router works wherever React is rendering–so take your pick!*
>
> *React Training Website*

La utilidad de React Router

Nuestro trabajo desarrolladores Frontend es *guiar* a los usuarios por medio de nuestras aplicaciones, para que gracias al contenido que creamos, podamos *transmitir* esta información. **React Router nos ayuda a cumplir esta tarea.** Dándonos las herramientas necesarias para dividir nuestra aplicación, y navegar por ella.
En este tutorial usaremos la **versión 4** de React Router. No es la versión oficial, pero resulta mucho más cómoda y útil de usar en nuestra aplicación.

Instalación

Si bien React Router (4) trae paquetes de soporte para aplicaciones mobiles o de escritorio, nuestra aplicación no necesitará de este soporte especial. Así que instalando la versión **web** de React Router sera suficiente:

```
npm install react-router-dom
# o usando yarn
yarn add react-router-dom
```

Tipos de Routers

Para usar React Router necesitamos envolver nuestra aplicación dentro de un **componente contenedor**. Este le dará a nuestra aplicación la capacidad de navegar entre rutas y componentes. **Pero existen varios Routers**:
Hash Router
Hace uso de la parte de *hash* de la URL (http://aplicación.com **/#/** {el resto de la URL}). Este tipo de router es muy útil cuando creamos aplicaciones estáticas. Osea, en las que no contamos con un servidor de desarrollo para nuestra aplicación, solo un archivo HTML y el código JavaScript con Reactjs.
Browser Router
Usa la API de navegación de HTML5. Gracias a esto tenemos acceso a herramientas que nos ayudaran a mantener sincronizada nuestra aplicación con la URL. Este Router necesita un servidor de desarrollo.
Memory Router
Guarda las rutas de nuestra aplicación en memoria, así que no tiene ningún tipo de sincronización con la URL del navegador. Es muy útil en aplicaciones de *React Native* o *Electron*.
Static Router
Existen momentos en que nuestra aplicación no tiene ningún tipo de interacción con los usuarios; como por ejemplo cuando se realiza un renderizado desde el Servidor, un método conocido como *Server Side Rendering*. De esta funcionalidad hablaremos más adelante. Pero recuerda que este Router lo usaremos cuando renderizamos nuestra aplicación desde el servidor.

-

Para nuestra tienda de libros, usaremos en un principio el **Hash Router**. Pero si tu aplicación necesita usar *Server Side Rendering*, lo mejor seria usar **Static Router** para renderizar las rutas desde el servidor y **Browser Router** para que tus usuarios puedan navegar usando el historial del navegador.
Tutorial relacionado: Server side Rendering by React Training

Una API demostrativa

Para construir nuestra aplicación, necesitamos una lista de libros. Aunque en una aplicación consumiríamos una API de GraphQL por ejemplo, nos enfocaremos en el uso de React Router y crearemos una lista de libros a mano:

```
// books.json.js

export default {
  books: [
    {
      id: 1,
      name: "Los detectives salvajes",
      description: "Los detectives salvajes es la quinta novela del
escritor chileno ...",
      likes: "91%",
      author: "Roberto Bolaño",
      year: 1998
    },
    {
      id: 2,
      name: "Cien años de soledad",
      description:
        "Cien años de soledad es una novela del escritor colombiano
Gabriel García ...",
      likes: "88%",
      author: "Gabriel García Márquez",
      year: 1967
    },
    // ...
]
```

Creación de rutas

Cómo aprendimos hace un momento, necesitamos un Router que envuelva nuestra aplicación. Así que usando el Hash Router vamos a preparar nuestra aplicación para recibir diferentes rutas.
Además, necesitamos un componente llamado `Route`, al que le diremos por medio de props la ruta que va a controlar y qué componente queremos renderizar.
Por último, también vamos a usar un componente llamado `Switch`, sera el encargado de decidir qué componente se renderizara, dependiendo las rutas que hallamos dado nuestros componentes *Route*.

```
// index.js

import React from "react"
import { render } from "react-dom"
import { HashRouter, Switch, Route } from "react-router-dom"
import Home from "./Home"
import BookList from "./BookList"
import BookDetail from "./BookDetail"

const App = () => (
  <HashRouter> /{/* envolvemos nuestra aplicación en el Router  */}
    <Switch> {/* también la envolvemos en el componente Switch */}
      <Route path="/" component={Home} exact /> {/* y creamos nuestras
rutas */}
      <Route path="/books" component={BookList} exact />
      <Route path="/books/:bookId" component={BookDetail} exact />
    </Switch>
  </HashRouter>
)

render(<App />, document.getElementById("root"))
```

Ahora que creamos las rutas, podemos crear el resto de componentes que necesitamos para nuestra tienda de libros. Estos serán el componente de **Home**, el componente de **Book List** y el componente de **Book Detail**.

Funcionamiento de nuestra aplicación

Ahora que tenemos nuestra aplicación con React Router funcionando, podemos crear los componentes necesarios para que nuestra aplicación funcione.
El primer componente que vamos a crear es **Book List**. Este componente sera el encargado de mostrarle al usuario los libros disponibles. Y para que el usuario pueda ver los detalles del libro, usaremos el componente `Link` de React Router. Este nos va a permitir cambiar la ruta, y mostrarle al usuario el componente de **Book Detail** por medio del `id` del libro y las *rutas dinámicas* de React Router:

```
// BookList.js

import React from "react"
```

```
import { Link } from "react-router-dom"
import api from "./books.json.js" // traemos todos los libros

const BookList = () => (
  <div className="BookList__container">
    <h1>Book List</h1>
    <ul className="BookList__list">
      {api.books.map(book => ( // y por cada uno creamos un link
        <li className="BookList__item">
          <Link to={`/books/${book.id}`}> {/* que nos lleve al detalle
del libro pormedio del id */}
            <h3>{book.name}</h3>
          </Link>
        </li>
      ))}
    </ul>
  </div>
)

export default BookList
```

Por ultimo, vamos a construir el componente de **Book Detail**. Para esto, necesitamos saber el id del libro que ha elegido el usuario. Para suerte nuestra, **React Router** ya se ha encargado de esto, y a todos los componentes que estén dentro de un componente Route, les ha pasadado por medio de props

```
// BookDetail.js

import React from "react"
import { Link } from "react-router-dom"
import api from "./books.json.js"

const BookDetail = props => {
  const { bookId } = props.match.params // aqui vienen los parametros
  const book = FindBookById(bookId)

  if (!book) { // si no existe el id del libro, mostraremos error
    return (
      <p>
        Lo sentimos, este libro no esta disponible...
        <Link to="/books">Ir atras</Link>
      </p>
    )
  }

  // si existe el libro, tenemos lo que necesitamos!
  return (
    <div id={bookId} className="BookDetail__container">
```

```
        <Link to="/books">Ir atras</Link>
        <h1>{book.name}</h1>
        <small>
          Una novela por {book.author} en {book.year}
        </small>
        <p>{book.description}</p>
        <p>Este libro cuenta con {book.likes} de likes.</p>
      </div>
    )
}

function FindBookById(bookId) {
  // los parametros vienen por defecto en formato string,
  // así que tenemos que convertir el id a formato número
  bookId = Number(bookId)

  // buscaremos el libro con el id que recibimos, y si no existe
  // devolveremos null para mostrarle al usuario que hubo un error
  return api.books.find(book => book.id === bookId) || null
}

export default BookDetail
```

-

Eso es todo!

Ya sabes usar React Router. Te invito a que nos compartas en los comentarios qué tal te ha parecido React Router; recuerda que puedes ver el demo de esta aplicación en este Link.

2 hace 10 meses Frontend con React.JS

Escribe tu comentario +2

2

rojasleon 14246 Puntos — 10 meses

Muy bien explicado Juan, claro y simple.

1

juandc — 10 meses

Thank u

Responder +2

1

larpa 8189 Puntos — 9 meses

Muchas gracias Juan! ahora mismo lo pondré en practica. gran explicación.

Responder +2

Entradas relacionadas

1

Thinking About React, Atomically ⚛ – Noteworthy - The Journal Blog

https://blog.usejournal.com/thinking-about-react-atomically-608c865d2262

juandc

1

Tiny Components: What Can Go Wrong? – Bits and Pieces

https://blog.bitsrc.io/tiny-components-what-can-go-wrong-d6aa42d71370

juandc

1

Architecting your React application. – Noteworthy - The Journal Blog

https://blog.usejournal.com/architecting-your-react-application-5af9cd65a891

juandc