Aplicaciones React

Tiempo estimado: 10min

Una vez presentada la herramienta Browserify en familia, ya podemos adentrarnos en React. Para comenzar, vamos a introducir el concepto de aplicación React.

La lección comienza introduciendo el concepto de aplicación React. Después, presentamos el directorio de aplicación. Aunque no se utiliza ninguna estructura particular, React es flexible en este aspecto, se recomienda seguir algún convenio. Acabamos la lección con los módulos que implementan la funcionalidad de React.

Al finalizar la lección, el estudiante sabrá:

- Qué es una aplicación React.
- Cómo usar React.
- Cuál es la estructura de directorios de una aplicación React.

Introducción

Una aplicación React (*React application*) es aquella que se encuentra escrita mediante React. Se suelen diseñar mediante una arquitectura de acceso a datos como Flux o Redux, ampliamente aceptadas por la comunidad. Se encuentran fuera del ámbito de este curso.

Generalmente, se desarrolla la aplicación React mediante su propio proyecto. Este proyecto contendrá el lado cliente de la aplicación. Mientras que el lado servidor se implementará en otro proyecto. Es muy común, utilizar Node y el *framework* Express para definir el servicio REST que atiende las solicitudes de datos de la aplicación React.

Lo importante es recordar que cada lado de la aplicación se recomienda desarrollarlo como un proyecto independiente. De esta manera, se puede delegar el desarrollo de la aplicación React a un programador *frontend* especializado, por ejemplo, en React y Bootstrap; y el lado servidor de la aplicación a otro en aplicaciones *backend*, por ejemplo, a uno especializado en Node, el cual no tiene por qué saber nada de React ni de Bootstrap.

Directorio de aplicación

El directorio de aplicación (*application folder* o *application directory*) es aquel que contiene el código de la aplicación React. React no requiere ninguna estructura de directorios específica aunque, como casi todo en programación, hay convenios. Para evitar que cada aplicación tenga una estructura particular, según la manera de pensar o concebir las cosas de cada programador, se utilizan los convenios. Con los convenios, se busca que todos parezcamos uno.

En el caso de React, por lo general, el código de la aplicación se ubica en el directorio app, el cual contiene las siguientes entradas:

- index.html. Página HTML principal de la aplicación.

  En ella, se importa el archivo empaquetado generado por Browserify, mediante un elemento <script>.
- index.jsx. Archivo inicial con la lógica de la aplicación React.

  En este archivo, se indica el componente raíz de la aplicación React.
- robots.txt. Archivo para los robots de los motores de indexación o búsqueda.
- sitemap.xml. Archivo de mapa de sitio para los motores de indexación o búsqueda.
- components/. Directorio con los componentes específicos de la aplicación.
- conf/. Directorio con las configuraciones de la aplicación para los distintos entornos.

Por ejemplo, el archivo production.js contendrá la específica del entorno de producción, mientras que development.js, la de desarrollo. Generalmente, cuando se genere el archivo empaquetado, se determina qué archivo de configuración usar.

- images/. Las imágenes estáticas de la aplicación.
- lib/. Directorio con código JavaScript específico de la aplicación.
- routes/. Las rutas de la aplicación.
- scripts/. *Scripts* de JavaScript específicos de navegador.
- sitemaps/. Si el archivo sitemap.xml es de indexación, los distintos mapas de sitio que referencia.
- stylesheets/. Hojas de estilo CSS, utilizadas en la aplicación.

En el directorio de la aplicación, además del directorio app, también suele haber otras entradas como, por ejemplo:

- .eslintrc. Archivo con las reglas ESLint para los archivos JavaScript.
- .eslintignore. Archivo con los archivos y directorios que debe ignorar ESLint.
- .gitignore. Archivo que contiene los archivos y directorios a ignorar de cara al control de versiones de Git.
- Justo.js. Catálogo de tareas automatizadas mediante Justo.js.

  Este archivo se utiliza cuando se usa Justo como herramienta de automatización, por ejemplo, para generar el archivo empaquetado de React mediante Browserify o la versión desplegable de la aplicación.
- package.json. Archivo con los metadatos de la aplicación.

  Igual que el de las aplicaciones Node.
- dist/. Directorio en el que depositar la versión de la aplicación a desplegar en el servidor web.

ESLint es una herramienta de análisis de código estático (*static code analysis*), o sea, un programa que lee código JavaScript, sin llegar a ejecutarlo, con objeto de detectar errores y patrones poco recomendables. Lo que se busca es que el código cumpla con los estándares y convenciones definidos por la organización. Por ejemplo, algunas organizaciones no desean que las expresiones condicionales de las sentencias ifs dispongan de asignaciones, sólo está permitida la comparación, o que en el código no aparezca el objeto console y cosas así. Son las herramientas de análisis de código las encargadas de detectar si alguna de las reglas definidas por la organización se está violando.

En aplicaciones React, se prefiere el uso de ESLint, por delante de JSHint, porque puede analizar código JSX, una extensión de JavaScript, usada por React, que permite añadir código HTML o XML de manera muy sencilla dentro de archivos JavaScript.

Generador justo-generator-react

Para facilitar el arranque del proyecto, podemos utilizar el generador de React de Justo.js. Justo es una herramienta de automatización que ayuda a automatizar tareas manuales para mejorar principalmente el rendimiento y la productividad. Resumiendo, un generador (*generator*) no es más que un componente que crea o produce algo, en nuestro caso, la estructura inicial de un proyecto React. Además, nos asiste en el desarrollo de aplicaciones React mediante la creación de archivos de componentes o de rutas, tal como veremos a lo largo del curso.

Para utilizarlo, es necesario tener instalado la aplicación CLI de Justo.js y el generador de React. Ambas cosas, se instalan fácilmente mediante NPM y se recomienda hacerlo globalmente:

```
npm install -g justo-cli justo-generator-react
```

Una vez instalados Justo y el generador, no tenemos más que pedirle a Justo que ejecute el generador de React, lo que desplegará una pequeña batería de preguntas y, con la información recopilada del usuario, generará la estructura inicial del proyecto:

```
justo -g react
```

Una vez ejecutado el generador, podremos consultar la estructura de directorios en el directorio actual.

Implementación de React

React se implementa en varios paquetes:

- react. Contiene el *kernel* de React.
- react-dom. Contiene el transformador DOM de React.

Por convenio, en nuestras aplicaciones, importaremos el módulo react como React; y react-dom como ReactDOM. Ejemplo:

```
import React from "react";
import ReactDOM from "react-dom";
```

Las APIs de estos módulos las desglosaremos detalladamente a lo largo del curso.

Resumen de dependencias

En resumen, para desarrollar aplicaciones React, debemos de utilizar:

- El empaquetador de aplicaciones Browserify, aunque también se puede utilizar webapp.
- El transformador babelify.

  Utiliza Babel para compilar código JSX y ES2015 o superior que finalmente empaquetará Browserify. Babel es el *transpiler* recomendado por el equipo de React. Es más, en el momento de escribir estas líneas, su desarrollador trabaja en Facebook que, recordemos, es la creadora de React.
- Una herramienta de automatización de tareas de desarrollo como, por ejemplo, Grunt, Gulp o Justo.js, así como los *plugins* correspondientes. A lo largo del curso, utilizaremos Justo.js.
- El *framework* React para desarrollar aplicaciones React: los paquetes react y react-dom.

El uso de una herramienta de automatización es opcional, pero muy recomendado. Si decidimos no usar ninguna, habrá que instalar globalmente Browserify. En otro caso, generalmente no hace falta porque los *plugins* de las herramientas suelen instalar Browserify localmente al proyecto. Lo que permite utilizar distintas versiones de Browserify según el proyecto. En nuestro caso, al utilizar Justo.js y el *plugin* justo-plugin-browserify, no hará falta instalar Browserify globalmente en la máquina, porque el *plugin* viene con una copia interna de Browserify.

Los paquetes react y react-dom hay que instalarlos localmente a cada proyecto. Se recomienda añadirlos a la propiedad dependencies del archivo package.json del proyecto. Si se utiliza el generador de Justo, no hay que hacerlo explícitamente, porque el generador lo hace por nosotros.

