Despachador y acciones

Tiempo estimado: 20min

Para comenzar, vamos a presentar los dos elementos principales de los inicios de flujo de datos en las aplicaciones React desarrolladas mediante una arquitectura Flux: el despachador y las acciones.

La lección la comenzamos haciendo una introducción de estos dos elementos y del paquete flux de Facebook que permite reutilizar clases que implementan conceptos de la arquitectura Flux como, por ejemplo, el despachador y los almacenes. A continuación, entramos en detalle con el despachador. Después, presentamos las acciones y los creadores de acciones. Finalmente, mostramos qué comandos del generador de Justo para React podemos utilizar para facilitar la creación de acciones y de los creadores de acciones.

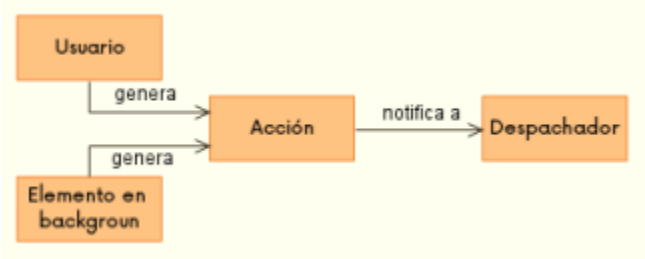
Al finalizar la lección, el estudiante sabrá:

- Qué es una acción.
- Qué es un creador de acciones.
- Cómo generar acciones.
- Quiénes generan las acciones.
- Qué es el despachador.
- Cómo utilizar el despachador de Facebook.
- Cómo utilizar el generador de Justo.js para facilitar la creación de acciones.

Introducción

El despachador (*dispatcher*) es el elemento de la aplicación que se encarga de invocar las operaciones que deben ejecutar los almacenes. Recordemos que los almacenes se encargan de acceder al servicio de datos de la aplicación, ya sea un servicio REST o una base de datos directamente.

Los componentes de React generan acciones, las cuales se comunican al despachador, el cual, entonces, invocará a los almacenes que tengan esa acción en su lista de atención. Una acción (*action*), entonces, no es más que un evento, acontecimiento o algo que necesitamos sea atendiendo por cero, uno o más almacenes. Las acciones las generan los componentes React de la aplicación cuando el usuario interactúa con ellos o bien mediante un elemento de la aplicación en *background*. Para ello, utilizan lo que se conoce como creadores de acciones (*action creators*), funciones específicas que generan las acciones y se las hacen llegar al despachador para que así éste pueda hacer su trabajo de director de orquesta.



Las acciones son eventos que originan el flujo de datos en una aplicación React, desarrollada usando la arquitectura Flux de Facebook. Una aplicación puede tener una o más acciones, pero en todo momento sólo dispondrá de un único despachador.

Paquete flux

El paquete flux, desarrollado por Facebook y disponible mediante NPM, contiene una colección de

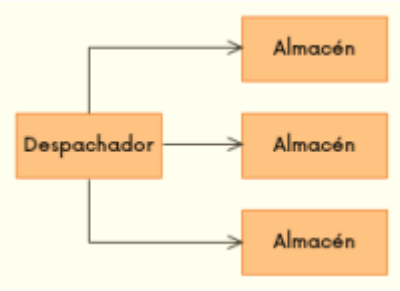
Copyright © 2016 nodemy.com. Reservados todos los derechos.

clases JavaScript que podemos utilizar gratuitamente en nuestras aplicaciones Flux. Para poder utilizarlo, hay que incorporarlo al proyecto, mediante la propiedad dependencies del archivo package.json:

```
"dependencies": {
  "flux": "*"
}
```

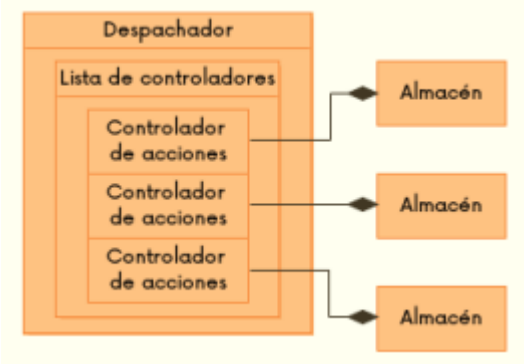
Despachador

Toda aplicación Flux debe tener un despachador (*dispatcher*). No es más que un objeto que se encarga de recibir las acciones que deben atender y procesar los almacenes. La aplicación sólo debe tener una instancia despachadora.



Para poder hacer su trabajo, el despachador contiene una lista de controladores de acción (*action handler list*), funciones que se encargan de atender las acciones recibidas por el despachador. Cada vez que el despachador recibe una acción, recorre esta lista y ejecuta el controlador. Entonces el controlador determina si la acción la debe atender, pues no todo controlador debe atender todas las acciones generadas. Si es así, la atenderá; en otro caso, no hará nada.

Los controladores tienen como objeto principal realizar operaciones de escritura sobre los orígenes de datos, generalmente bases de datos, los cuales pueden ser accesibles directamente o indirectamente, por ejemplo, mediante un servicio REST. Así pues, cada controlador está asociado a un único almacén. Por convenio y buenas prácticas, los almacenes registran un único controlador de acciones, el cual atenderá todas aquellas acciones que le conciernen. Pero ojo, pueden registrar tantos como sea necesario. Se prefiere un único controlador por almacén que dada una acción compruebe si es de su incumbencia y, si lo es, entonces haga lo que tenga que hacer.



No confundir los conceptos de controlador y creador de acciones. El creador genera la acción y se la notifica al despachador para que la atienda. Mientras que el controlador lo que hace es procesar una determinada acción comunicada por el despachador.

El despachador no es más que el objeto que controla el flujo de datos de la aplicación. Lo podemos ver como una tabla que registra los controladores de acciones que tiene la aplicación. Cada vez que se genera una acción, el despachador ejecuta los controladores asociados a la acción, consultando para ello la tabla. En todo momento, el despachador sólo puede estar atendiendo una única acción. Las restantes se encontrarán encoladas en espera.

Es importante recordar que el despachador no accede a los datos ni procesa datos. Su responsabilidad se reduce a recibir las acciones generadas y notificar a los almacenes de que han ocurrido para que puedan hacer su trabajo asociado.

Copyright © 2016 nodemy.com. Reservados todos los derechos.

Implementación de Facebook

Facebook proporciona una implementación del despachador, en el paquete flux, para que la usemos en nuestras propias aplicaciones. Esta implementación registra simplemente controladores de acción, sin indicar la acción asociada. Cada vez que se genera una acción, el despachador recorre *todos* los controladores registrados por los almacenes y los va ejecutando uno a uno. Bajo esta implementación, los controladores de acciones son los responsables de analizar la acción y determinar si deben hacer algo con ella. Bastará con comprobar la propiedad type del objeto acción que pasa el despachador como argumento a los controladores.

De manera predeterminada, el despachador se añade al archivo app/dispatcher/AppDispatcher.js como una instancia de la clase Dispatcher:

```
import {Dispatcher} from "flux";
export default new Dispatcher();
```

Esta clase proporciona un método a través del cual los almacenes deben registrar sus controladores de acción, register(), y otro para que los creadores de acciones le notifiquen las acciones, dispatch(). Cada uno de ellos lo veremos en su debido momento.

Controladores de acción

Los controladores de acción, los veremos más detenidamente en la lección de almacenes, por ahora, simplemente introducir cómo son para facilitar el aprendizaje. Sigamos. Un controlador de acciones no es más que una función que recibe como parámetro un objeto con la información de la acción generada, conocido formalmente como objeto acción (*action object*). Entonces, analiza este objeto y detecta si es de su interés y, si lo es, procesa la acción.

¿Qué significa que procesa la acción? Sencillo: si la acción tiene como operación asociada modificar una fila de una tabla de la base de datos, lo que hará es invocar la operación del almacén a través de la cual se realiza esta operación, pasándole los datos con los que tiene que trabajar, los cuales se encuentran en el objeto acción.

Su signatura es como sigue:

```
function handle(action)
```

Parámetro	Tipo de datos	Descripción
action	Object	Objeto acción: • type (string o number). Indica el tipo de acción. • data (object). Contiene los datos relacionados con la acción.

Como en el momento de registrar el controlador no se indica las acciones a las que se asocian, el despachador, ante toda acción, ejecutará todos los controladores registrados. Siendo obligación de las funciones controladoras las encargadas de determinar si deben procesar la acción, es decir, identificar aquellas acciones que deben atender.

He aquí un ejemplo de controlador de acción registrado ante el despachador por un almacén:

```
function handleAction(action) {
  switch (action.type) {
    case EmployeeAction.INSERT_EMPLOYEE:
      store.insertEmployee(action.data);
      break;

    case EmployeeAction.UPDATE_EMPLOYEE:
      store.updateEmployee(action.data);
      break;

    case EmployeeAction.DELETE_EMPLOYEE:
      store.deleteEmployee(action.data);
      break;

    default:
      //No hacer nada: el almacén no debe procesar la acción.
      //No es de su incumbencia.
  }
}
```

Copyright © 2016 nodemy.com. Reservados todos los derechos.

Registro de controladores de acción

El registro de un controlador de acción (*action handler register*) es la operación mediante la cual se inscribe un controlador de acción en el despachador, esto es, en la lista de controladores. Para este fin, se utiliza el método register() del despachador:

```
register(handler) : DispatchToken
```

Parámetro	Tipo de datos	Descripción
handler	Function	Función controladora: fn(action).

El método devuelve un objeto que actúa como identificador de registro y que es necesario para dar de baja el controlador del despachador.

He aquí un ejemplo ilustrativo:

```
var id = dispatcher.register((action) => {
  //…
});
```

Baja de controladores

Para suprimir un controlador de acción del despachador, hay que utilizar el método unregister() junto con el identificador que obtuvimos en el momento de registrarlo:

```
unregister(id)
```

Parámetro	Tipo de datos	Descripción
id	DispatchToken	Identificador de registro asignado al controlador.

Ejemplo:

```
dispatcher.unregister(id);
```

Método Dispatcher.isDispatching()

Si deseamos saber si el despachador se encuentra atendiendo una acción en ese mismo momento, podemos utilizar su método isDispacthing():

```
isDispatching() : boolean
```

Acciones

Una acción (*action*) representa un evento que lleva asociado algo que hacer en la aplicación, relacionado con los datos como, por ejemplo, insertar, actualizar o suprimir una fila de una tabla SQL. Concretamente, en una aplicación React diseñada usando la arquitectura Flux, el acceso a los servicios de datos no es directo. La aplicación debe crear una acción, o sea, invocar un creador de acción, solicitando así la operación de datos que desea se lleve a cabo. Por ejemplo, si queremos que se compruebe si hay correo nuevo, habrá que generar la acción correspondiente como, por ejemplo, check-email.

La acción es el punto de comienzo del flujo de datos en una aplicación Flux. Es importante tener claro que esta acción puede ser generada por un componente React como consecuencia de la interacción del usuario con la interfaz de usuario, por ejemplo, al hacer clic en un botón Enviar. O bien puede generarla un objeto interno de la aplicación, en *background*, sin intervención del usuario, por ejemplo, uno que realiza tareas automáticas de manera periódica. La cuestión es que cuando deseemos acceder a datos, no debemos acceder directamente a los almacenes, sino generar una acción que acabe atendiendo el almacén al que accederíamos directamente. Recordemos que a los almacenes se accede mediante acciones pasadas al despachador.

Las acciones tienen básicamente dos elementos:

- Un tipo (*type*). No es más que un identificador único, generalmente textual, que la distingue de las demás de la aplicación.
- Un creador de acción (*action creator*). Una función que tiene como objeto generarla y hacérsela llegar al despachador.

Cuando se desarrolla una aplicación, hay que tener clara la lista de acciones (*action list*), una

Copyright © 2016 nodemy.com. Reservados todos los derechos.

enumeración de las posibles acciones de la aplicación. Por convenio, una aplicación dispone de una o más listas de acciones, cada una de ellas relacionadas con una determinada categoría o dominio, asociadas a su correspondiente archivo de acciones (*action file*). Por convenio, se añade el sufijo Action a estos archivos; ejemplos: UserAction.js, RoleAction.js y EmployeeAction.js.

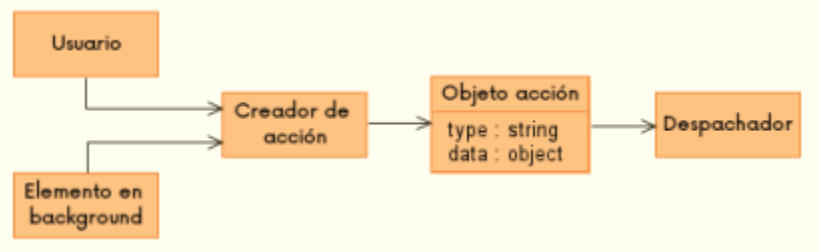
Cada categoría de acciones vendrá bajo su propio objeto JavaScript, donde cada propiedad representará una acción posible. A su vez, cada acción tendrá asociado un nombre identificativo que la describa. Por convenio, estos identificadores pueden contener varias palabras separadas por guiones (-). Veamos unos ejemplos ilustrativos: comprobar-correo, enviar-correo, registrar-empleado, suprimir-proyecto, etc.

Por convenio, las acciones se enumeran en el directorio app/actions, donde cada categoría de acciones se representa, tal como hemos indicado hace unos instantes, mediante su propio archivo. Por ejemplo, las acciones relacionadas con los empleados se registrarán en el archivo EmployeeAction.js como sigue:

```
//app/actions/EmployeeAction.js
export default {
  INSERT_EMPLOYEE: "insert-employee",
  UPDATE_EMPLOYEE: "update-employee",
  DELETE_EMPLOYEE: "delete-employee"
};
```

Creadores de acciones

Las acciones se deben generar mediante lo que se conoce formalmente como creadores de acciones (*action creators*), funciones que reciben los datos asociados a la acción a generar, crean el objeto acción y notifican o pasan la acción al despachador, el cual, recordemos, invocará a los almacenes que están asociados a la acción para su procesamiento o tratamiento.



Un objeto acción (*action object*) representa una acción generada. Tiene básicamente las siguientes propiedades:

- type (string o number). El tipo o nombre identificativo de la acción.

  Lo usan los controladores para determinar ante qué acción se encuentran y, así, determinar si deben atenderla o pasar de ella.

- data (object). Objeto de datos asociado a la instancia de la acción.

  Por ejemplo, si la acción tiene asociado la actualización de un registro de la base de datos, los datos a actualizar. Así pues, es usado principalmente por los almacenes.

Cada creador de acciones está asociado a una determinada acción y recibe los datos que se asociarán a esa acción en particular. Como está asociado a una determinada acción, no necesita recibir cómo argumento el tipo de acción, queda implícito en su funcionalidad. Pero sí debe recibir los datos. Así pues, el objetivo del creador es crear el objeto acción y, a continuación, invocar al despachador para que sea procesada por los almacenes correspondientes.

Por convenio y buenas prácticas, los creadores se definen en el mismo archivo que sus acciones asociadas. Algunas organizaciones prefieren separar los creadores de las acciones. Esto tiene un pequeño problema. Si en algún momento deseamos suprimir una determinada acción, por la razón que sea, tendremos que actualizar dos archivos. Si los ubicamos en el mismo archivo, la supresión de la acción y de su función creadora estarán en el mismo punto. Uno debajo del otro. Es menos probable olvidar algo si tenemos ambas cosas, todo hay que decirlo muy relacionadas entre sí, en el mismo archivo.

En nuestro caso, usaremos los archivos de acciones, ubicados en el directorio app/actions. He aquí unos

Copyright © 2016 nodemy.com. Reservados todos los derechos.

ejemplos de acciones y sus creadores asociados:

```
export default {
  INSERT_EMPLOYEE: "insert-employee",  //acción
  insertEmployee(data) {               //creador de acción
    dispatcher.dispatch({
      type: this.INSERT_EMPLOYEE,
      data: data
    });
  },

  UPDATE_EMPLOYEE: "update-employee",  //acción
  updateEmployee(data) {               //creador de acción
    dispatcher.dispatch({
      type: this.UPDATE_EMPLOYEE,
      data: data
    });
  }
};
```

El desacoplamiento de las acciones y los almacenes es bastante grande. Las acciones no saben, ni desean saber, qué almacenes las atenderán. Mientras que cada almacén sólo necesita saber qué acciones debe procesar; para así, atender sólo aquellas que son de su interés, descartando todas las demás.

Por otra parte, el desacoplamiento entre las acciones y el despachador es total. Los creadores de acciones invocan el método dispatch() del despachador. No interactúan de ninguna otra manera con él. Y al despachador no le importa en absoluto cuáles son las acciones que tiene la aplicación. Cada vez que reciba una acción, invocará todos los controladores de acciones que han registrado los almacenes de la aplicación y éstos serán los encargados de determinar si las acciones generadas son de su interés.

Gracias a este desacoplamiento, el mantenimiento de la aplicación es más sencillo.

Generador de Justo

Para ayudar a crear los archivos de acciones, las acciones y los creadores de acciones, podemos utilizar el generador de Justo para React.

Generación del directorio del proyecto

Tal como vimos en el curso Desarrollo de aplicaciones webs con React, el generador de Justo se puede utilizar para crear la estructura de directorios de un proyecto de aplicación React. Recordemos de las prácticas que cada vez que invocamos el generador sin ningún comando, el generador recolecta información para crear la mejor estructura de directorios según nuestras necesidades. Entre las preguntas que hace se encuentra si la aplicación usará el paquete flux. Si así es, lo seleccionaremos y entonces será el propio generador el que registre el paquete en la propiedad dependencies del archivo package.json.

Pero como las aplicaciones React se pueden implementar bajo otras arquitecturas como, por ejemplo, Redux, el generador no añade ningún directorio específico de la arquitectura Flux. Para hacerlo, hay que invocar el comando flux explícitamente antes de comenzar a trabajar con los aspectos específicos de esta arquitectura:

```
justo -g react flux
```

Entre otras cosas, el comando creará:

- El directorio app/actions, donde registrar los archivos relacionados con las acciones.
- El archivo app/dispatcher/AppDispatcher.js, donde se encuentra el despachador de la aplicación. Este despachador es una instancia de la clase Dispatcher del paquete flux de Facebook.
- El directorio app/stores, donde se implementarán los almacenes de la aplicación.

Copyright © 2016 nodemy.com. Reservados todos los derechos.

Comando flux action file

Mediante el comando flux action file, se crea un archivo de acciones en el directorio app/actions:

```
justo -g react flux action file
```

Recordemos que generalmente se dedica un archivo a cada dominio de la aplicación que podría ser implementado de manera aislada y por miembros distintos del equipo de desarrollo.

Comando flux action

Mediante el comando flux action se actualiza un archivo de acciones, añadiendo una acción y su creador:

```
justo -g react flux action
```

Copyright © 2016 nodemy.com. Reservados todos los derechos.