Registro de eventos

Tiempo estimado: 15min

Uno de los aspectos más importantes de una aplicación web es su registro de eventos. A través de él, podemos saber qué operaciones han generado error e incluso qué están solicitando los usuarios. Y a partir de ahí, analizar la información para mejorar la aplicación y el contenido del sitio web.

Comenzamos la lección introduciendo qué es el registro de eventos y su arquitectura básica. A continuación, presentamos Morgan, el registro de eventos oficial desarrollado por el equipo de Express. Después, presentamos los tipos de registro usados más habitualmente en las aplicaciones webs, los archivos access.log y error.log. Finalmente, explicamos la rotación y el purgado de registros.

Al finalizar la lección, el estudiante sabrá:

- Qué es un registro de eventos.
- Para qué sirve un registro de eventos.
- Cómo usar el registro de eventos Morgan.
- Qué diferencia hay entre los archivos de registro access.log y error.log.
- Por qué se usa la rotación y el purgado de registros.

Introducción

El registro de eventos (*event log*), también conocido como registro de alertas (*alert log*) o registro de errores (*error log*), es un dispositivo como la consola o un archivo en el que la aplicación registra mensajes de información relacionados con su actividad como, por ejemplo, qué peticiones ha recibido, en qué dirección de IP y puerto está escuchando, qué errores se han producido, etc. Se utiliza como punto de partida cuando se detecta una incidencia.

El registro es muy útil cuando la aplicación está en producción, así como durante la fase de desarrollo. Por lo que aprender a configurar uno es de vital importancia.

Cada mensaje escrito en el registro de la instancia se conoce formalmente como entrada de registro (*log entry*) y representa un evento acaecido en la aplicación. Por lo general, estas entradas se mantienen en disco, en un archivo de registro (*log file*), el cual se recomienda mantener un tiempo como, por ejemplo, dos semanas o un mes, por si posteriormente fuera necesario para abrir un caso de soporte con el fabricante.

Arquitectura de registro de eventos

Casi todos los componentes de registro de eventos utilizan una arquitectura básica. Cada uno puede hacerlo usando una jerga distinta, pero lo importante es comprender la idea general que se esconde tras esta arquitectura.

Los componentes son básicamente dos: los escritores y los formateadores. Los escritores (*appenders* o *writers*) son los encargados de escribir las entradas de registro en un destino. Generalmente, se puede configurar cualquier número de escritores. Por otra parte, los formateadores (*layouts*) son los componentes utilizados por los escritores para obtener el texto a escribir como representación de cada evento, es decir, el escritor le pasa un evento al formateador, éste genera el texto que lo representa, siendo este texto lo que finalmente escribe el escritor.

escritores

Un escritor (*appender*) es un componente que se encarga de escribir las entradas de registro en un destino como, por ejemplo, un archivo, una base de datos o la consola. El registro de eventos delega la escritura de sus mensajes en los escritores, pudiéndose configurar tantos escritores como sea necesario. Se distingue varios tipos

de escritores:

- El escritor de consola (*console appender*), aquel que escribe en la consola.
- El escritor del registro de sistema (*syslog appender*), aquel que escribe en el registro del sistema. Específico de sistemas Linux.

  Se utiliza cuando la aplicación se ejecuta en un sistema Linux y se desea añadir las entradas al registro del sistema, junto al resto de entradas de otras aplicaciones que se están ejecutando en el sistema.
- El escritor del registro de eventos de Windows (*Windows event log appender*), aquel que escribe en el registro de eventos Application. Específico de sistemas Windows.

  Se utiliza cuando la instancia se ejecuta en un sistema Windows y se desea añadir las entradas al registro de eventos Application del servidor, junto al resto de entradas de otras aplicaciones que se están ejecutando en el sistema.
- El escritor de archivo de registro (*log file appender*), aquel que escribe en un archivo de disco.

Formateador

Un formateador (*layout*) es el componente que se encarga de generar el texto de la entrada a registrar. Para cada evento, el escritor le pasa el evento al formateador para que le devuelva el mensaje que debe escribir.

La idea es indicar qué campos del evento hay que mostrar en el mensaje y la posición de cada uno de ellos. Cada campo se conoce formalmente como modificador (*modifier*) o palabra de conversión (*conversion word*). El patrón (*pattern*), pues, es la cadena de texto que contiene el formato del mensaje de los eventos, es decir, el formato de las entradas del registro.

Morgan

El equipo de Express mantiene oficialmente un componente de *middleware* para el registro de eventos, conocido como morgan.

Sus principales características son:

- Tiene soporte oficial de Express.
- Permite escribir entradas en cualquier flujo de Node como, por ejemplo, la consola o un archivo.
- Es fácil de usar.
- Es fácil de aprender.
- Es fácil de configurar.

Para usar morgan, hay que añadir el paquete homónimo a las dependencias de la aplicación. Esto es, a la propiedad dependencies del archivo package.json.

Función morgan

El paquete morgan consiste en una función cuya invocación devuelve una función de *middleware*:

```
function morgan() : function
function morgan(format) : function
function morgan(format, options) : function
```

Parámetro	Tipos de datos	Descripción
format	string	Patrón de formato. Formatos predefinidos: • combined. Formato estándar del servidor web Apache. • common. Formato común del servidor web Apache. • dev. Formato corto a usar generalmente en fase de desarrollo. • short. Formato corto. • tiny. Formato muy corto.

options	object	Opciones de registro: • immediate (boolean). ¿Escribir la entrada en la solicitud HTTP, en vez de en la respuesta? • skip (function). Función que indica si morgan debe omitir el registro de la entrada: fn(req, res) : boolean. • stream (Stream). *Stream* de salida en el que escribir la entrada.

Registro de la función de middleware

Recordemos que morgan no es una función de *middleware*, sí lo es la función que devuelve como resultado, lo que finalmente registraremos en la pila. He aquí un ejemplo ilustrativo:

```
app.use(morgan("combined"));
```

Se recomienda añadir el registro al comienzo de la pila de *middleware*. Así , nos aseguramos que siempre procesa la solicitud en curso y, por lo tanto, lleva a cabo el registro de su entrada.

Formato

El formato (*format*) es un texto patrón que indica qué información registrar en cada entrada. Puede ser un patrón predefinido como, por ejemplo, combined, common, dev, short o tiny, o bien uno personalizado de la aplicación.

Cuando no se utiliza un formato predefinido, se utiliza modificadores para definir qué información y en qué posición de la entrada registrarla. Los modificadores representan determinadas propiedades de la solicitud y/o respuesta en procesamiento. Veamos las más utilizadas:

- :date o :date[formato]. Fecha y hora actuales. Los posibles formatos son clf (17/Jun/2016:08:55:46 +0000), iso (2016-06-17T08:55:46.000Z) o web (Fri, 17 Jun 2016 08:55:46 GMT).

  Si no se especifica ninguno, se usa web de manera predeterminada.
- :http-version. Versión HTTP de la petición HTTP en curso.
- :method. Método HTTP de la petición HTTP en curso.
- :referrer. Cabecera Referrer de la petición en curso.
- :remote-addr. Dirección de IP del cliente que envío la petición HTTP.
- :req[cabecera]. Valor de la cabecera HTTP indicada de la petición.
- :res[cabecera]. Valor de la cabecera HTTP indicada de la respuesta.
- :response-time o :response-time[dígitos]. Milisegundos transcurridos desde que Morgan recibe la petición y se escribe alguna cabecera en la respuesta.
- :status. Código de estado de la respuesta HTTP.
- :url. URL de la solicitud HTTP.
- :user-agent. Agente de usuario que remitió la solicitud HTTP.

A continuación se presenta los patrones de los formatos predefinidos:

Nombre	Patrón
combined	:remote-addr - :remote-user [:date[clf]] ":method :url HTTP/:http-version" :status :res[content-length] ":referrer" ":user-agent"
common	:remote-addr - :remote-user [:date[clf]] ":method :url HTTP/:http-version" :status :res[content-length]
dev	:method :url :status :response-time ms - :res[content-length]
short	:remote-addr :remote-user :method :url HTTP/:http-version :status :res[content-length] - :response-time ms
tiny	:method :url :status :res[content-length] - :response-time ms

Flujos

Una vez tenemos claro el formato de las entradas, lo siguientes es determinar dónde registrarlas. Los dos

dispositivos más utilizados son la consola y/o un archivo. Para ello, hay que utilizar la opción stream.

consola

Para indicar la consola, podemos omitir la opción stream o bien asignarle el valor de process.stdout. Ejemplo:

```
app.use(morgan("combined"));
app.use(morgan("combined", {stream: process.stdout}));
```

archivo

Para indicar un archivo, generalmente se usa un flujo de archivo devuelto por la función createWriteStream() del módulo fs. Veamos un ejemplo ilustrativo:

```
app.use(morgan(
  "combined",
  {
    stream: fs.createWriteStream(path.join(__dirname, "logs/access.log"), {flags: "a"})
  }
));
```

tipos de registro

Una aplicación puede tener cero, uno o más registros. Generalmente, en las aplicaciones webs se distingue dos tipos de registro cuando se llevan a disco: acess.log y error.log. Cada uno con su función y objetivo bien definidos.

archivo access.log

El archivo access.log, ubicado generalmente en la carpeta logs de la aplicación, tiene como objeto registrar los recursos solicitados por los clientes. La idea es disponer de un sitio a partir del cual analizar:

- Qué recursos se han solicitado y cuántas veces cada uno.
- De dónde proceden las solicitudes.
- Cuál es el patrón de acceso al sitio web.

Así pues, la idea es mantener información como, por ejemplo:

- Fecha y hora de la solicitud.
- Recurso solicitado.
- Usuario que realizó la solicitud.
- Versión y método HTTP de la solicitud.
- Campo de cabecera Referrer de la solicitud HTTP, para saber desde qué sitio web se referencia el recurso.
- Dirección de IP del cliente para saber su procedencia.
- Agente de usuario usado por el cliente.
- Código de estado de la respuesta.
- Tiempo de respuesta.

Generalmente, se usa el formato predefinido combined. Ejemplo:

```
app.use(morgan("combined", {
  stream: fs.createWriteStream(path.join(__dirname, "logs/access.log"), {flags: "a"})
}));
```

Lo más habitual es coger el contenido del archivo y volcarlo a una base de datos para su posterior análisis por parte del SEO.

archivo error.log

El archivo error.log, por convenio también ubicado en el directorio logs de la aplicación, mantiene información sobre el funcionamiento de la aplicación como, por ejemplo, errores que se han producido, configuración de arranque usada, procesamiento de inicio de sesión, dirección de IP y puerto en el que escucha la aplicación, etc.

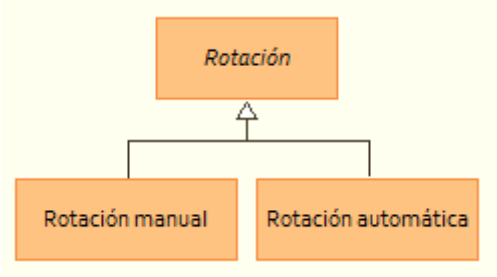
Generalmente, se registra información como:

- Fecha y hora en la que se produjo el error.
- Mensaje descriptivo.

Rotación de registros

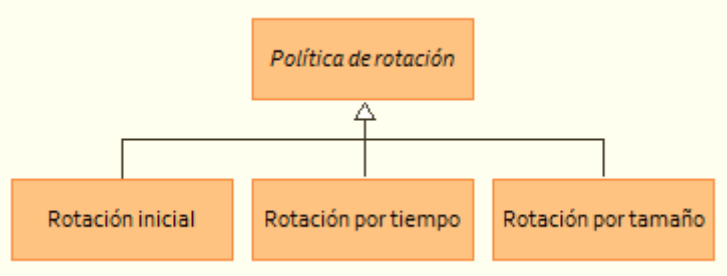
La rotación (*rotate*) es el proceso mediante el cual se hace una copia del archivo de registro y se comienza con uno nuevo. Es una característica muy valiosa que impide que un archivo crezca descontroladamente. La idea es muy sencilla, se coge el archivo de registro actual, se archiva en el directorio de registro y se comienza uno nuevo. De esta manera, se puede tener controlado el tamaño del archivo. En muchas ocasiones, el servicio de soporte de los fabricantes suele solicitar los archivos de registro para echarles un vistazo. Si rotamos los archivos, por ejemplo, cada día, y nos piden los archivos de los últimos siete días, bastará con proporcionarles el archivo actual y los de los seis últimos días.

Atendiendo a quién realice la rotación, se distingue entre manual o automática.



La rotación automática (*autorotate*) es aquella que lleva a cabo la aplicación de manera automática, implícitamente. En cambio, la rotación manual (*manual rotate*) es aquella que realiza explícitamente un *webmaster*, por ejemplo, mediante el uso de una herramienta de automatización como Justo.js o Puppet.

La política de rotación (*rotate policy*) es aquella que define el momento en el que se realiza: al inicio, pasado un tiempo o alcanzado un tamaño.



Bajo la política de rotación inicial (*start rolling policy*), la rotación se realiza en el momento de arrancar la aplicación. Cada vez que se arranca, se rota el archivo, comenzando siempre con un registro vacío. Mediante la política de rotación por tiempo (*time based rolling policy* o *age based rolling policy*) la rotación se dispara tras un determinado período de tiempo o en un determinado momento del día, por ejemplo, a las 24h. Finalmente, la política de rotación por tamaño (*size based rolling policy*) la dispara cuando el archivo alcanza un determinado tamaño.

Debido a la rotación, un aspecto a tener en cuenta es cuántos archivos mantener. Cada vez que se realiza una rotación, se coge el archivo actual, se renombra y se crea uno nuevo en el que seguir escribiendo; a esta operación se la conoce formalmente como archivado del archivo de registro (*log file archiving*) y tiene como objeto mantener los últimos *n* archivos. Se recomienda mantener tantos archivos como sea necesario para mantener la información de los últimos 15 a 31 días, según los estándares internos de la organización. Los archivos se deben de mantener para su posible consulta, por parte de los desarrolladores, SEOs o webmasters, en caso de incidencia, con objeto de comprobar si un determinado problema se lleva produciendo desde hace tiempo y por si el fabricante requiere información en caso de apertura de un caso de soporte.

purgado del registro

La información de los registros no hay que mantenerla indefinidamente. Pero sí hay que mantenerla un tiempo, conocido formalmente como período de retención (*retention period*). Transcurrido éste, hay que llevar a cabo lo que se conoce como purgado del registro (*log purging*), proceso mediante el cual se elimina los archivos del registro cuyo período de retención ha expirado. Por lo general, se recomienda utilizar un período de retención de como mínimo 15 días y como máximo uno o dos meses. Este purgado se suele hacer todos los días, en las horas de menor carga que generalmente coincide con la noche.