NODEMON

Tiempo estimado: 5min

Hasta el momento, cada vez que hemos realizado un cambio y hemos deseado probarlo, en la mayoría de las ocasiones, hemos tenido que reiniciar la aplicación manualmente. Ir a la consola, detener la aplicación y volver a iniciarla. Existe una manera mucho más práctica, útil y productiva de hacer las cosas. Usar un monitor de cambios. En esta lección, vamos a presentar nodemon.

La lección comienza introduciendo el concepto de monitor de cambios. A continuación, se presenta nodemon, cómo instalarlo y cómo usarlo. Finalmente, ilustramos cómo realizar monitorizaciones parciales.

Al finalizar la lección, el estudiante sabrá:

- Qué es un monitor de cambios y para qué usarlos.
- Cómo instalar y usar nodemon.
- Cómo monitorizar sólo algunos archivos y/o directorios del proyecto.
- Cómo retrasar los reinicios para agrupar varios cambios en uno.

Introducción

Un monitor de cambios (*change monitor*) es un programa que ejecuta una aplicación, supervisando los cambios de sus archivos. De tal manera que si alguno de ellos cambia, el monitor reiniciará la aplicación automáticamente. Son muy útiles durante la fase de desarrollo. Ayudan a no tener que reiniciar manualmente nuestra aplicación web cada vez que realizamos un cambio y deseamos probarlo.

nodemon

Con más de medio millón de descargas mensuales, uno de los monitores de cambios más conocidos y utilizados es nodemon. Su sitio oficial es nodemon.io.

Sus principales características son:

- Es fácil de aprender.
- Es fácil de usar.
- Consume pocos recursos.
- Permite realizar monitorización completa y parcial.
- Es de código abierto.
- Su uso es gratuito.

nodemon no es específico de Express. Se puede utilizar con cualquier tipo de aplicación servidora desarrollada en Node.

Instalación de nodemon

nodemon se puede instalar global o localmente a la aplicación en la que deseamos usarlo.

Instalación global

Para instalar nodemon globalmente, no hay más que usar NPM como sigue:

```
npm install -g nodemon
```


Instalación local

Cuando se instala localmente, generalmente se hace como dependencia de desarrollo, mediante la propiedad devDependencies del archivo package.json de la aplicación Express:

```
"devDependencies": {
  "nodemon": "*"
}
```

Uso de nodemon

Una vez instalado, no hay más que indicar en la propiedad scripts.start del archivo package.json que la aplicación se ejecute mediante nodemon, en vez de usar node directamente. Si lo tenemos instalado globalmente:

```
"scripts": {
  "start": "node ./app.js",
  "start-dev": "nodemon ./app.js",
}
```

Mientras que si lo tenemos instalado localmente:

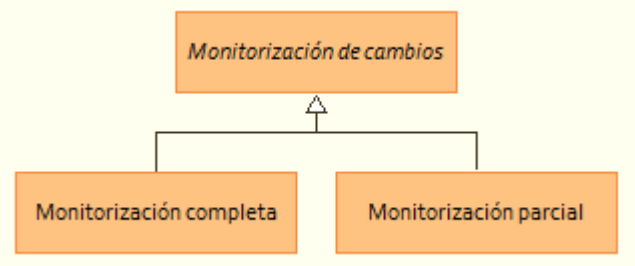
```
"scripts": {
  "start": "node ./app.js",
  "start-dev": "./node_modules/.bin/nodemon ./app.js"
}
```

Por buenas prácticas, se recomienda configurar el *script* de inicio start para que se ejecute bajo node; creando uno específico para el desarrollo denominado start-dev, el cual se ejecutará con nodemon.

A partir de ahora, cuando iniciemos la aplicación mediante npm start, estaremos ejecutándola como siempre, directamente con node. En cambio, cuando ejecutemos npm run start-dev, estaremos arrancándola con nodemon.

Monitorización parcial

nodemon puede realizar básicamente dos tipos de monitorización de archivos: completa o parcial.



Mediante la monitorización completa (*full monitor*), supervisa todos los archivos del proyecto. En cambio, mediante la monitorización parcial (*partial monitor*), sólo aquellos directorios o archivos que le indiquemos.

De manera predeterminada, realiza una monitorización completa. Para indicar una parcial, hay que usar la opción -w o --watch, una vez para cada directorio o archivo que deseemos monitorice:

```
-w dir|archivo
--watch dir|archivo
```

Por ejemplo, si sólo deseamos monitorizar los cambios del archivo app.js y del directorio routes, usaremos:

```
nodemon -w app.js -w routes app.js
```

Reinicio retardado

Se puede configurar nodemon para que haga una pequeña espera antes de reiniciar la aplicación. Esto se conoce como reinicio retardado (*delayed restart*). Si no indicamos nada explícitamente, realizará un reinicio por cada cambio detectado.

Cuando sabemos que habrá varios cambios simultáneos, es mejor intentar agrupar el mayor número de ellos, reduciendo el número de reinicios. Para ello, se puede usar la opción --delay, mediante la cual se indica el tiempo que debe pasar antes de reiniciar la aplicación y, así, agrupar varios cambios en uno. Generalmente, con un par de segundos es suficiente.

He aquí un ejemplo ilustrativo, en el que el retraso es de 2 segundos:

```
./node_modules/.bin/nodemon --delay 2s --ext hbs,js,json ./bin/www.js
```

