# Procesadores del lenguaje

Ficheros de configuración de nginx.

**Alumnos:**

**Marcos Pena Calvar**

**David Santamaría Martín**

Contenido

[Especificación del DSL 3](#_Toc502662168)

[El contexto principal 3](#_Toc502662169)

[El contexto “events” 3](#_Toc502662170)

[El contexto “http” 4](#_Toc502662171)

[El contexto “server” 5](#_Toc502662172)

[El contexto “location” 5](#_Toc502662173)

[Otros contextos: 6](#_Toc502662174)

[Formato del fichero: 6](#_Toc502662175)

[Acerca del analizador léxico y su código: 7](#_Toc502662176)

# Especificación del DSL

La mayor parte de esta especificación está extraída de <https://www.digitalocean.com/community/tutorials/understanding-the-nginx-configuration-file-structure-and-configuration-contexts>.

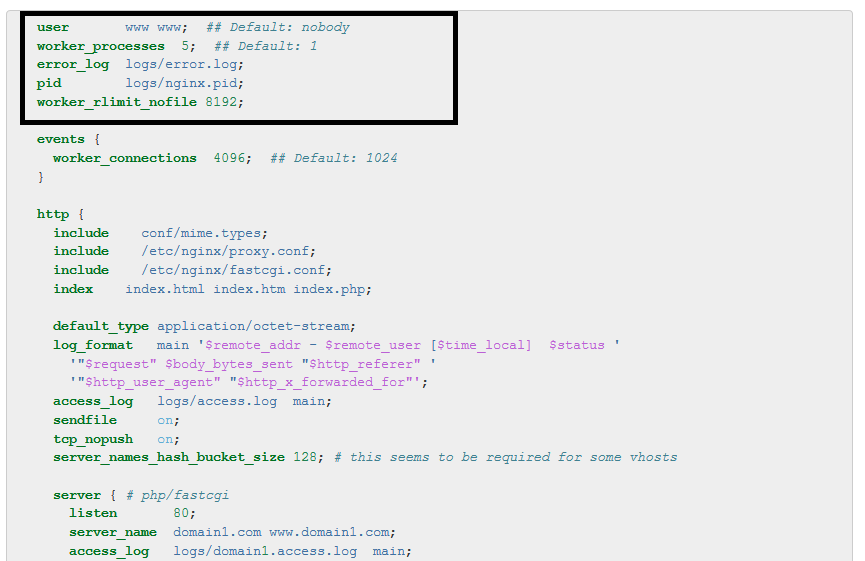
Nginx es un servidor web que destaca por su gran rendimiento con múltiples conexiones concurrentes y por ser especialmente bueno sirviendo contenido estático. Gran parte de la flexibilidad de esta herramienta radica en sus ficheros de configuración, que se generan en base a su propio DSL.

Este DSL se divide mayoritariamente en dos partes:

* Contextos: contienen configuraciones para ciertas partes del servidor.
* Parejas clave-valor: todas las configuraciones se escriben en este formato. Siempre existe un espacio entre la clave y el valor (como mínimo). La clave consta de letras, guiones, números y guiones bajos.

## El contexto principal

Es el contexto más exterior.



En este contexto se ubican configuraciones que afectan a todas las partes de la aplicación. Dos ejemplos característicos de qué se puede colocar aquí son el nombre de usuario que tendrá acceso al servidor mediante una Shell y la ubicación de los ficheros log.

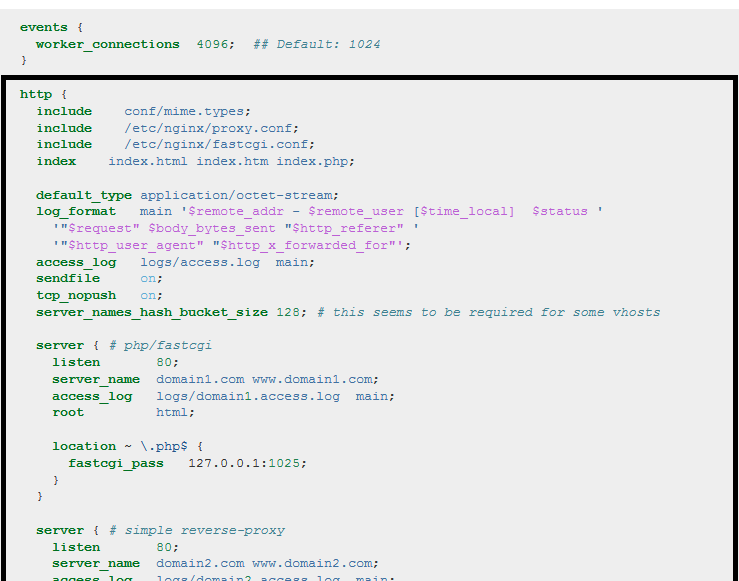
## El contexto “events”

Este contexto siempre queda dentro del contexto principal.



Estas configuraciones indican a los procesos trabajadores (nginx utiliza múltiples procesos) cómo deben de manejar las conexiones entrantes, además de algunos comportamientos específicos. Más allá de todo esto, no tiene más interés.

## El contexto “http”



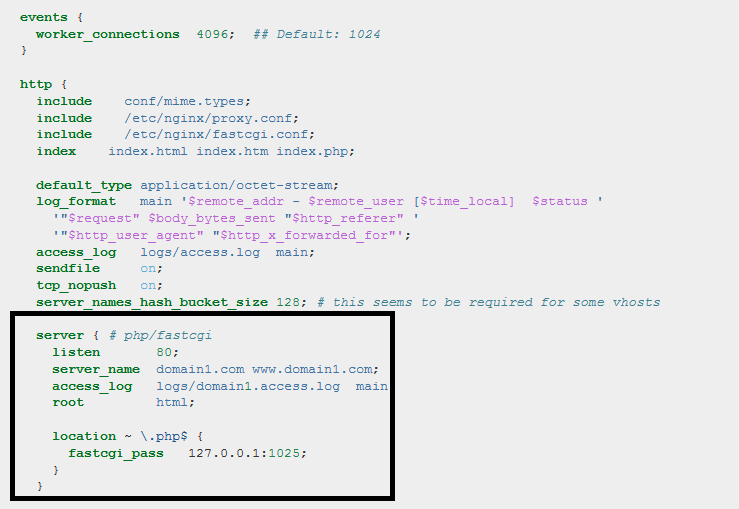
Se coloca también dentro del contexto principal. Sirve para manejar varias configuraciones que afectan a la parte de servidor web o proxy de nginx. Básicamente, las configuraciones que se coloquen van a afectar a cómo el programa maneja conexiones HTTP(S).

Ejemplo de cosas que podemos configurar: ubicación de ficheros log, comportamiento de TCP, compresión, y el uso de páginas de error para resolver ciertas peticiones.

Este contexto tiene varios subcontextos dentro.

## El contexto “server”

Este contexto siempre se ubica dentro del contexto http.



Aunque puede poseer más variables, las más importantes son dos:

* Listen: indica en qué puerto debe escuchar el servidor.
* Server\_name: indica qué página web debe responder ante esa conexión (nginx permite guardar múltiples páginas web)

## El contexto “location”

Estos bloques se colocan dentro del contexto “server” y pueden anidarse entre sí.



Este contexto se encarga de decidir cómo se sirve la parte estática de la URI de la petición (por ejemplo, en la web <http://ejemplonodarclic.com/blog>, el contexto “server” se encargaría de localizar los ficheros para ejemplonodarclic.com, y el contexto “location” localizaría los ficheros para la parte de /blog). Se puede usar para contenidos estáticos o dinámicos, pasándole el control en ese caso al servidor php, rails, o similares.

## Otros contextos:

* Contexto if: permite definir condiciones de reescritura. No debe usarse para otros fines.
* Contexto upstream: define servidores a los cuales nginx puede enviar peticiones proxy. Dentro de este contexto se utiliza la palabra “server”.
* Contexto “mail”: define el comportamiento del servidor de correo de nginx.

## Formato del fichero:

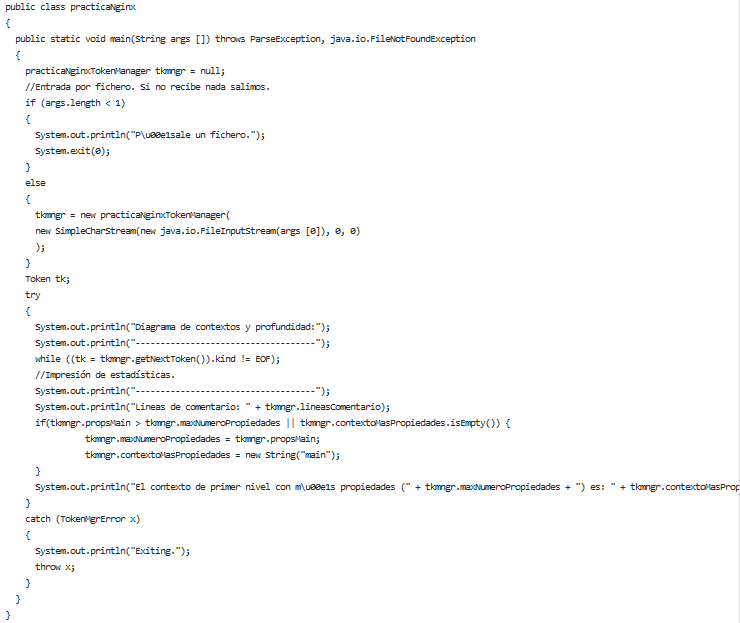
* Primeramente, se colocarán las configuraciones del contexto principal, fuera de cualquier otro contexto. Luego, se añadirán contextos como se desee, respetando la anidación. El contenido del contexto está encerrado entre llaves.
* Algunos contextos pueden tener argumentos, en cuyo caso se colocan después del nombre del contexto y antes de la llave de apertura.
* Las configuraciones siguen este formato: clave valor; donde valor puede contener cualquier carácter excepto el punto y coma (que define el final del valor).
* Los comentarios empiezan con el símbolo # y terminan con un salto de línea.

# Acerca del analizador léxico y su código:

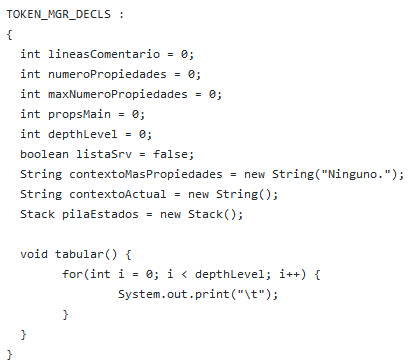
El analizador léxico realiza las siguientes tareas:

* Cuenta el número de líneas de comentario.
* Busca el contexto de primer nivel con más propiedades (los contextos que queden dentro de otros contextos los ignora)
* Muestra un diagrama de anidamiento de los contextos.

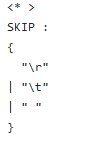
Revisando el código con detalle encontramos estas secciones:



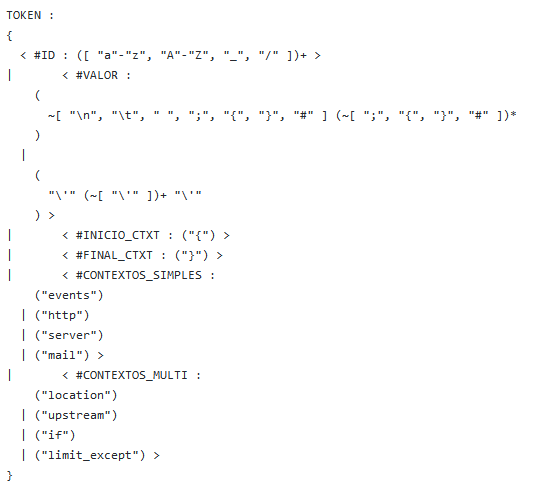
Este bloque simplemente ejecuta el TokenManager e imprime las estadísticas al final. No tiene nada especialmente interesante.



Declaraciones del TokenManager. Se incluye una función que se usa para formatear la salida por pantalla.



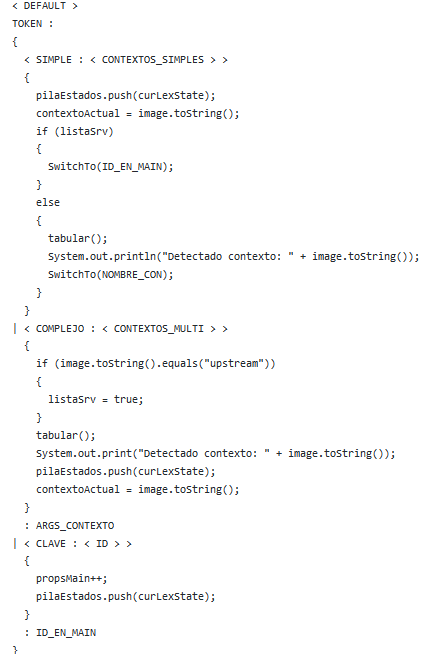
Saltar retornos de carro, tabuladores y espacios. Los saltos de línea tienen algunas propiedades especiales en nginx (por ejemplo, se pueden formatear los logs insertando saltos de línea).



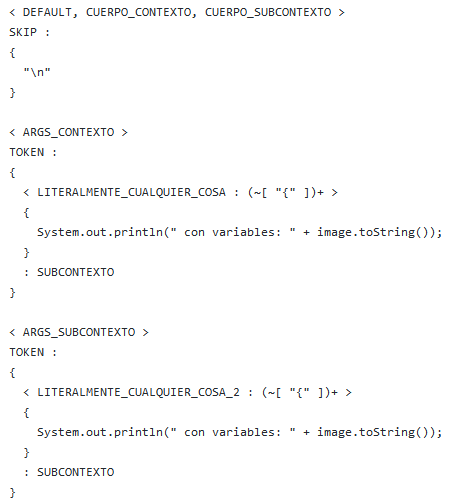
Se definen algunos tokens para ser reutilizados a lo largo del código.



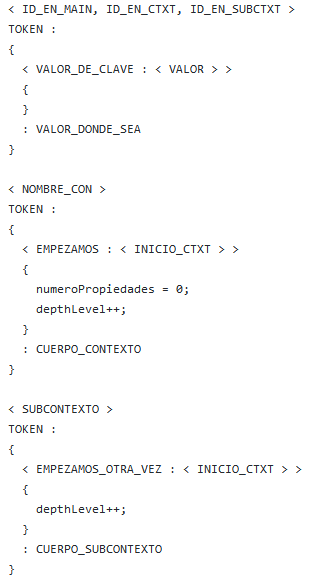
Filtrado de comentarios. Cuando se acaba el comentario, se cuenta una línea más y se vuelve a donde estaba antes el programa.



Este bloque de código controla el contexto principal. Si se encuentra algún subcontexto, pasa a él y lo imprime por pantalla. Si el contexto es “upstream” debemos permitir que la palabra reservada “server”, que corresponde a otro contexto, se interprete de una manera distinta (lo hacemos con un flag).

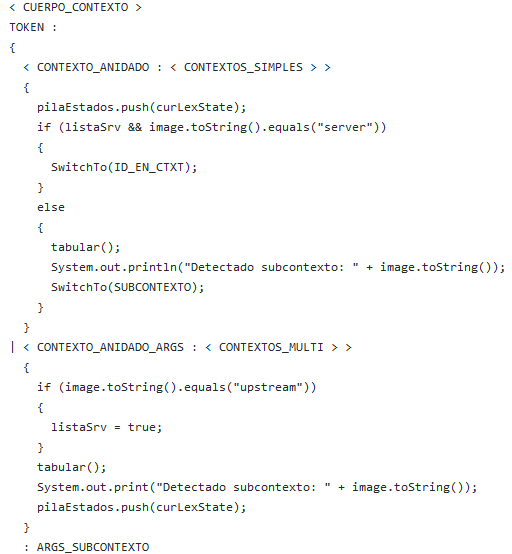


Omitir los saltos de línea en ciertos estados donde no son interesantes. Además, si el contexto detectado posee argumentos, se capturan y se pasan por pantalla.

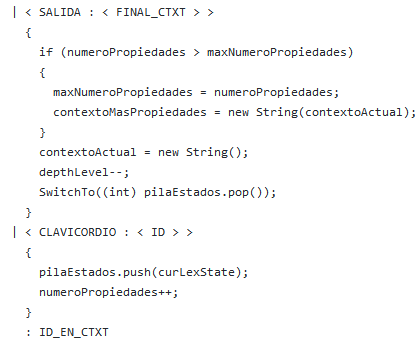


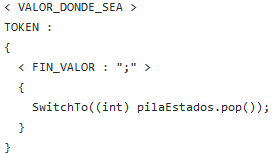
En el caso de que encontremos un ID de una configuración, pasaremos a capturar su valor. Como la expresión regular de ese valor es peligrosa (captura muchas cosas), debemos minimizar su uso, y para ello lo mandamos a otro estado.

A la hora de analizar un contexto o subcontexto, cuando encontremos la llave de apertura debemos incrementar el nivel de profundidad y, si el contexto es de primer nivel, reiniciar el contador de propiedades.



A la hora de encontrar un subcontexto, se guarda el estado actual y se procesa el subcontexto. También se imprime por pantalla su nombre.



Sin embargo, si encontramos el final del subcontexto comparamos el número de propiedades, lo actualizamos, reducimos el nivel de profundidad y volvemos al estado anterior. Al encontrar una propiedad, simplemente la contamos. El código para procesar los subcontextos es muy similar a este, así que no será necesario explicarlo nuevamente. 

Al terminar de capturar el calor, volvemos al estado anterior. Esto lo hacemos para unificar la captura de valores en todo el código.