Cómo usar el archivo de configuración .htaccess

Cardenal Gardoki, 1
48008 BILBAO (Vizcaya)
Teléfono: 902 012 199
www.hostalia.com

HOSTALIA .com

A la hora de desarrollar una página web, nos podemos encontrar varios tipos de archivos, que van desde los archivos correspondientes al lenguaje de programación utilizado para el desarrollo, hasta archivos de imágenes. Pero en ocasiones también forma parte de ellos otro tipo de archivos que nos permiten modificar ciertos parámetros del servidor donde está alojada nuestra web. Es el caso del archivo .htaccess, un archivo de configuración que se utiliza en el servidor web Apache.

A lo largo de este White Paper haremos un recorrido sobre todo lo que nos ofrece este archivo y las cosas que podemos llegar hacer por medio de él.

Qué es un archivo .htaccess

```
php_value post_max_size 500M
php_value upload_max_filesize 500M
php_value request_order "GPC"
php_value session.gc_probability 0
php_value date.timezone "Europe/London"
php_value phar.readonly Off

RewriteEngine On
RewriteCond %{REQUEST_FILENAME} -s [OR]
RewriteCond %{REQUEST_FILENAME} -l [OR]
RewriteCond %{REQUEST_FILENAME} -d
RewriteRule ^.*$ - [NC,L]
RewriteRule ^.*$ index.php [NC,L]
RewriteBase /
```

Los archivos .htaccess (conocidos también como acceso de hipertexto) son archivos de configuración de directorios y proporcionan los mecanismos necesarios para realizar cambios en la configuración del servidor web Apache, que afectarán al directorio donde esté colocado (con sus respectivos subdirectorios) sin necesidad de tener que editar el archivo de configuración de Apache.

Todo aquello que podemos modificar mediante este tipo de archivos viene determinado por la directiva "AllowOverride" utilizada en el archivo de configuración de Apache "httpd.conf", y que es distinto dependiendo del proveedor de alojamiento web.

La característica AllowOverride especifica, por categorías, lo que harán las directivas que se encuentren en un archivo .htaccess. De este modo, el funcionamiento de las directivas que se utilicen en un archivo .htaccess dependerá en gran medida de cómo haya sido configurado el servidor por su proveedor de hosting. La mayoría de los hostings permiten directivas .htaccess.

Son muchas las funcionalidades que nos ofrecen este tipo de archivos, pero de forma habitual son utilizados para especificar ciertas restricciones de seguridad para un determinado directorio, reescribir URLs largas y complejas en otras más sencilla o bloquear a usuarios por su dirección IP, tal y como veremos a lo largo de este libro blanco.

Antes de seguir nos gustaría destacar que los archivos .htaccess no utilizan una extensión como tienen

cualquier otro archivo. Programas como el blog de notas tienden a agregar automáticamente la extensión .txt, por lo que para poder hacer uso de ellos deberemos quitarla.

Otra cosa a la hora de manejar este archivo de configuración de directorios que debemos tener presente es el tema de permisos que debe tener el archivo para poder editarlo y que se ejecuten todas las directivas que incluyamos en él, de ahí que tenga que tener asignados permisos de lectura, escritura y ejecución.

El tema de permisos también es importante para que puedan ser editados de forma automática por otras aplicaciones como WordPress o Prestashop, aplicaciones de software libre que utilizan este archivo de configuración para su funcionamiento y que pueden editar el contenido del .htaccess sin que intervenga el usuario, por tanto hay que tenerlo en cuenta.

Manera en la que trabaja un archivo .htaccess

Como ya hemos comentado anteriormente, el uso de los archivos .htaccess nos permite modificar parámetros de configuración del servidor, pero el uso de estos archivos trae consigo una reducción del rendimiento del servidor, ya que Apache buscará en cada uno de los directorios que forman parte de la web la presencia de archivos .htaccess para determinar las directivas que se utilizan en él.

La manera en que trabaja se puede resumir de la siguiente forma:

- Ponemos un archivo .htaccess en la raíz de nuestro sitio y otro en un directorio que forma parte de la web. Cuando se hace una llamada a este directorio, Apache busca en él el archivo y también lo hace en un nivel más arriba.
- Las directivas son aplicadas en el orden en que son encontradas por Apache. De esta forma, un archivo .htaccess de un directorio puede sobrescribir las directivas encontradas en un archivo situado en un nivel superior, y así sucesivamente.
- Apache busca directivas en cualquier archivo .htaccess del directorio particular que ha sido llamado y en los que están más arriba, para determinar las directivas que utilizará cuando procese la petición de llamada de la página de su sitio.

Usando un .htaccess

Veamos a continuación algunos usos del fichero .htaccess.

Control de acceso a carpetas

Podríamos querer desactivar el acceso al contenido dentro de una carpeta vía web, pero sí que se pudiera acceder a ellos mediante el sistema de archivos. Veamos varios ejemplos.

En el caso que no queramos que nadie acceda a ella vía web, utilizaríamos el siguiente código.

```
deny from all
```

Si queremos denegar el acceso excepto a una dirección IP, deberíamos utilizar lo siguiente.

```
deny from all
allow from DIRECCIÓN_IP
```

También podemos bloquear el acceso a un único archivo. En este caso el código sería el siguiente.

```
<Files privado.html>
Order allow,deny
Deny from all
```

Listado de carpetas

Se puede dar el caso que queramos mostrar el contenido de una carpeta en la estructura de directorios. Para eso utilizaremos lo siguiente.

```
Options +Indexes
```

Si por el contrario queremos evitar el listado de carpetas, lo que podemos utilizar es lo siguiente.

```
IndexIgnore *
```

Activar compresión GZIP

Hay ocasiones en las que son interesantes activar la compresión de datos para mejorar la velocidad de carga de la web.

Para activar la compresión GZIP, lo primero que necesitamos saber es si el servidor apache donde está alojado nuestro sitio permite la compresión. Para ello, en el archivo .htaccess hacemos una comprobación para detectar si existe el módulo mod_gzip, y si es así, entonces que realice las tareas de compresión.

El código que deberíamos añadir a nuestro fichero sería el siguiente.

```
# Compresión GZIP con mod_gzip
<IfModule mod_gzip.c>
 mod_gzip_on Yes
```

```
  mod_gzip_dechunk Yes
  mod_gzip_item_include file \.(html?|txt|css|js|php|pl)$
  mod_gzip_item_include handler ^cgi-script$
  mod_gzip_item_include mime ^text/.*
  mod_gzip_item_include mime ^application/x-javascript.*
  mod_gzip_item_exclude mime ^image/.*
  mod_gzip_item_exclude rspheader ^Content-Encoding:.*gzip.*
</ifModule>
```

Páginas de error personalizadas

Mediante un fichero .htaccess podemos indicar que cada vez que se produzca algún tipo de error, se redirija hacia un determinado fichero, lo que nos permitirá personalizar esos errores en vez de que aparezcan los típicos mensajes de Apache.

Para lograr esto debemos utilizar la siguiente instrucción.

```
ErrorDocument XXX /ruta_al_fichero_de_error_personalizado
```

En el siguiente código sustituiríamos XXX por el error en cuestión (404, 403, 500…) y /ruta_al_fichero_de_error_personalizado por la ruta completa hasta llegar al archivo.

Esta instrucción puede aparecer varias veces en un archivo .htaccess, pero cada aparición debe hacer referencia a un código de error diferente.

No mostrar las “www”

Hay ocasiones en las que por tema de posicionamiento, no nos interesa que los buscadores nos posicionen con la “www”. Para lograr que nuestro sitio no las muestre, podemos hacer uso de las reglas de reescritura para indicarle que toda petición que venga con las “www”, sea redirigido al dominio sin las “www”. Para ello podemos utilizar el siguiente código.

```
RewriteEngine on
RewriteCond %{http_host} ^www\.example\.com[nc]
RewriteRule ^(.*)$ http://example.com/$1 [r=301,nc]
```

El número 301 corresponde con el tipo de redirección que estamos llevando a cabo.

Evitar el HotLinking

El hotlinking consiste en realizar un enlace directo a ficheros que pertenecen a otro sitio web. Con esto estamos consiguiendo mostrar en nuestra web una imagen o vídeo que no está subido en nuestro servidor, lo que consume la transferencia del sitio donde está subido ese fichero. Si queremos evitar que nos "roben" nuestra transferencia o simplemente no queremos que nadie enlace archivos subidos en nuestro servidor, lo podemos hacer mediante el siguiente código.

```
RewriteEngine on
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !^http://([-a-z0-9]+\.)?example\.com[NC]
RewriteRule .*\.(zip|mp3|avi|wmv|mpg|mpeg)$
http://www.example.com/images/nohotlink.gif [R,NC,L]
```

Con este código lo que estamos indicando es que si el visitante no proviene de nuestro sitio y hace una petición de un archivo cuya extensión es zip, mp3, avi, wmv, mpg o mpeg, se redireccione hacia la imagen nohotlink.gif.

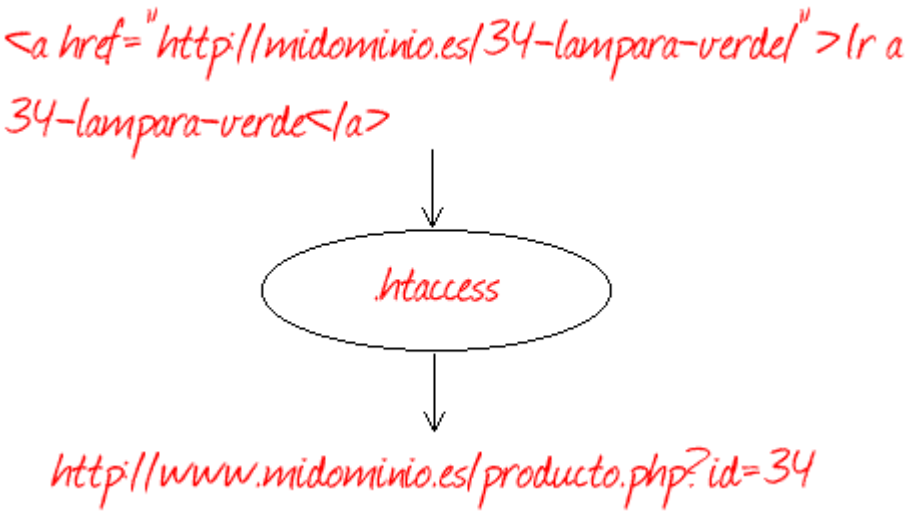
Cambiar la página por defecto

De forma habitual, los servidores web están configurados para que el primer archivo que ejecuten lleve por nombre index.html, index.htm o index.php, pero puede haber ocasiones en las que nos interese que el primer archivo que se ejecute cuando alguien entre al dominio sea otro distinto. Esto se puede lograr mediante la ejecución de la siguiente instrucción.

```
DirectoryIndex about.html
```

Este caso le estamos indicando que sea about.html el primer archivo en ejecutarse.

Crear URLs amigables



Una URL amigable consiste en "disfrazar" una url que contenga parámetros para que quede más elegante y sea más fácil de recordar para los usuarios, además de ayudarnos a mejorar nuestra posición en los buscadores.

Por ejemplo, esto no es una url amigable:

clientes/usuario.php?id=1350&ciudad=madrid

Esto si es una url amigable:

clientes/usuario/1350/madrid

Las urls amigables son las que tendremos que añadir en los enlaces que forman parte de nuestra web, pero estas urls amigables no son entendidas por el servidor web si no lo "explicamos" en .htaccess para que sepa cómo actuar.

RewriteRule ^clientes/usuario/([0-9]+)/([a-zA-Z-]+)$ clientes/usuario.php?id=$1&ciudad=$2

Expliquemos un poco qué significa cada cosa que hemos utilizado.

- RewriteRule: significa que vamos a escribir una regla para reescribir una dirección.
- El carácter ^ significa el principio de la expresión.
- El signo $ significa el fin de la expresión.
- El valor ([0-9]+) significa que ahí va el primer valor que pasamos a la url. En este caso estamos indicando que ahí puede ir cualquier número del 0 al 9 pudiéndose repetir todas las veces que sean necesarias. Este será la primera variable que pasamos y que es identificada con el nombre de $1.

- El valor ([a-zA-Z-]+) es lo mismo que el caso anterior, pero en esta ocasión el valor podrá estar formado por letras y el guión medio. En este caso corresponde con la segunda variable que es identificada con el nombre $2.
- La expresión clientes/usuario.php?id=$1&ciudad=$2, corresponde con el script que se ejecutarán y donde $1 y $2 tomarán los valores pasados en la URL.

Redirigir a usuarios móviles

En la actualidad la mayoría de los usuarios disponemos de dispositivos móviles capaces de navegar por la red. Ante esta situación, es muy importante asegurarnos de que nuestros portales se visualicen bien en el tamaño de pantallas de estos dispositivos. Para ello, una buena opción es desarrollar una versión para que cuando el visitante acceda desde un dispositivo móvil, se redirija hacia la carpeta que contenga la versión móvil.

Para lograr esto de forma automática, podemos hacer uso del siguiente código:

```
RewriteCond %{HTTP_USER_AGENT} ^.*iPad.*$

RewriteRule ^(.*)$ http://tudominio.com/movil [R=301]
```

En este caso le estamos indicando que si se visita desde un iPad, se redirija a la carpeta “movil”. De forma similar se puede actuar con el resto de dispositivos móviles que hay en el mercado.

Redirigir visitantes a una nueva página

En muchas ocasiones ocurre que cuando rediseñamos una página web, las URLs cambian, lo que puede afectar a nuestro posicionamiento en los buscadores.

Para evitar esto, es recomendable hacer uso de las redirecciones 301 para indicar a los navegadores que la url antigua ya no existe y que ha sido sustituida por la que le indiquemos.

Esto se consigue con el siguiente código.

```
redirect 301 /pagina_vieja.html http://www.sudominio.com/pagina_nueva.html
```

Si lo que hemos hecho ha sido renombrar un directorio, se puede utilizar este tipo de redirecciones que afectará a todas las páginas que se encuentran dentro de él.

```
redirect 301 /directorio_viejo http://www.sudominio.com/directorio_nuevo
```

Configurar la caché de los ficheros

El fichero .htaccess también nos brinda la oportunidad de poder cachear ciertos ficheros en indicar el tiempo que deben estar en la caché antes de volver a ser consultados.

```
<FilesMatch "\.(html|htm|xml|txt|xsl)$">

Header set Cache-Control "max-age=7200, must-revalidate"

</FilesMatch>
```

En este caso estamos indicando que para los ficheros con extensión .html, .htm, .xml, .txt o xsl, estén en caché durante dos horas (7.200 segundos).

Cambiar valores de configuración de Apache

Como explicamos anteriormente, los ficheros .htaccess permiten cambiar ciertos parámetros de la configuración de Apache, siempre y cuando esté habilitada esta opción en el servidor.

Para cambiar estos parámetros se utiliza la instrucción "php_value" seguida del nombre de la directiva y el valor que le asignamos. Veamos algunos ejemplos.

- Cambiar el tiempo de ejecución de script: **php_value max_execution_time 100**
- Cambiar el tamaño máximo de los datos enviados por POST: **php_value post_max_size 20M**.
- Cambiar el tamaño máximo permitido para los archivos subidos desde la web: **php_value upload_max_filesize 50M**

Añadir una marca de agua a todas las imágenes

Se puede meter una marca de agua a todas las imágenes en lugar de tener que editarlas una a una, en este sencillo enlace te explica cómo se hace: http://foro.elhacker.net/empty-t318024.0.html

Consejos a la hora de utilizar un archivo .htaccess

A continuación os dejamos una serie de consejos a la hora de utilizar este tipo de archivo.

- No abusar del uso de este tipo de archivo de configuración. Como ya explicamos, por cada petición, el servidor analiza los archivos .htaccess lo que aumenta la carga del servidor. Cuanto más pequeño sea este archivo, mejor rendimiento conseguiremos.
- Mantén tu archivo organizado. Utiliza para ello los comentarios con almohadilla (#) al principio de las líneas. Cuesta mucho entender un archivo .htaccess una vez que este crece demasiado.
- Cuando hagamos uso de reglas de reescritura de URLs, agrega la opción [L] a aquellas páginas finales. Con esto le estaremos indicando al servidor que no procese más reglas, mejorando el rendimiento.
- Cuidado con la herencia. Un archivo .htaccess en un directorio afecta a todos los niveles que estén por debajo de él, por lo que una regla mal indicada puede afectar al resto de la aplicación.