



**How to use htaccess to secure,  
manage and optimise your website.**

## Introduction

Craig Parker our resident SEO Expert has put together a detailed document on using htaccess to how to manipulate htaccess to help secure and manage your site.

Craig discusses the three major uses for htaccess, security, redirects and optimisation and gives you the code examples to implement them on your site as well as explaining how each one works so you have a better idea of how to manage your site.

The guide also comes with an example htaccess file, it contains all the code samples that are discussed in the document so all you need to do is customise and enable commands to use on your site.

One of the hardest things about htaccess is finding reliable information, there are so many code samples out there, often bogged down inside complicated questions or multilevel rules that simply don't apply to most sites, our collect features easy to follow code examples and rules which should be useful on a day to day basis for all your affiliate marketing sites.

All this code has been tested multiple times but please keep in mind some hosts place specific restrictions on htaccess access and functionality, we can't possibly account for all these variations but on the whole these rules should work with your hosting provider, if you find specific problems don't hesitate to try contacting your hosting company and see how they can help.

## Why You Should Learn About .htaccess

A lot of people ask me about htaccess, they ask me what it's for, they ask me for code snippets and they ask me how it can help their SEO. In this document we are going to look at how you can use htaccess to tweak your website to be more user and SEO friendly and I'm going to give you some code snippets to use as well as explain when you should be using them.

The most important thing to remember about redirects and htaccess is this, it's not a catchall solution, if you build things right in the first place you won't need to use htaccess to "fix up" the problems, using CMS like Wordpress will dramatically reduce you're involvement with this pesky little file.

That being said nothing in life is perfect and tightening up your site can help improve SEO and usability.

You can find a template htaccess file containing all the commands in the document below but if you're not familiar with how to use htaccess I strongly recommend you read through this paper.

<http://www.moreniche.com/tutorials/example-htaccess.zip>



## What is .htaccess?

**Hypertext Access** or **htaccess** is an Apache configuration web server configuration file. It is loaded by the server and allows you to significantly modify its behaviour by specifying redirects, turning features on and off or protecting special sections of your site. In a most literal way htaccess is simple a small text file with the filetype of .htaccess and no name.

Due to its strange file type naming structure you may not always be able to see htaccess, if you find this to be a problem you may need to enable “dot files” or “.files” in your FTP software or OS viewing options and if you are really struggling you can create and edit the file as a .txt and simple change the extension and name when you upload it.

You may also find that some hosts block access to htaccess, this is because it can used to drastically modify server behaviour and if you’re on a shared server solution this may not always be in the hosting companies best interests.

Although not written in a specific coding language htaccess makes frequent use of regular expression for matching strings of text and creating rules. Regular expression (or regex for short) can be very hard to understand for people not used to looking at code but is essentially quite simple, it’s main problem is that it’s incredibly *strict* when it comes to making the odd mistake, a misplaced / or . can completely break htaccess and bring your site down.

You should always be careful when modifying your htaccess file, keep backups in case you make untraceable mistakes but don’t be scared to try and learn, if you do make a mistake and bring the site down with a 500 error then just revert the last change you did or comment the line out (with a #) and re-upload and 99.9% of the time your site will be fine.

A few basic things to know about htaccess;

- You can comment links in htaccess with # it doesn’t matter how many you use so I tend to use ## or ### for descriptive comments and single # for disable/enabling code
- Use notepad to edit in Windows, it’s quick and easy!
- ALWAYS keep a backup
- Htaccess applies itself to the directory it’s in and all lower directories therefore putting on in the root will mean it can usually control your whole site
- You **can** use htaccess on IIS too, you need <http://www.isapirewrite.com/>
- Mistakes in htaccess will cause a 500 internal server error bringing down your entire site, **don’t panic** just revert to your backup version or comment out the latest code.

## Uses for htaccess

Now we’re going to look at some of the different uses for htaccess along with some code snippets you can just copy, paste edit and go with, remember this however these code snippets work fine, **for me on my various hosts and servers.**

Your particular host may be blocking functionality or doing other things that restrict what you can and cannot do, if you have trouble remember to contact your hosting provider or read their FAQ section.

If you run into serious problems just restore from the backup you should have made. Obviously you need to keep an eye out for the placeholder text in this document and replace it with your own; placeholder text is **highlighted in green**.

## Enabling Rewrites

You need this small piece of code at the top of your htaccess to ensure the rewrites below work, make sure it's the first thing in your htaccess file.

```
##Rewrite Engine on code - MUST BE ACTIVE for rewrites##  
  
Options +FollowSymLinks  
RewriteEngine on
```

## Section 1: Security & Control

This section looks at rules that will help keep your website secure and control how people access it, while not always specifically SEO focused it is extremely important to have a secure and stable site and these code snippets should help with that.

### Error Handling

One of the most common uses for htaccess is handling header status errors, these are the numbers that come back from the server when a client makes a request, for example you will all be familiar with the error status 404, commonly called "page not found". By default your browser will show an ugly generic 404 page in the event of this error but with some htaccess magic we can customise this page!

It's a very important step for SEO for two reasons;

1. You need to report the correct status code to search engines. This means if a page has moved for good you report 301, if it's not there and has never been there it should be 404, if it's there and working it should be 200, incorrect reporting of errors can lead to duplicate crawling and indexing problems.
2. It helps your users, they will know when they are in the wrong section of a site and can easily follow some recommended links or go back, if you just bounce them back to the homepage or throw up a generic browser 404 it can shake their confidence and force them to leave the site.

You will need to create your own error landing pages, this can be done via a CMS if your CMS does not already have one or manually and just upload the files to your server.

They don't have to be in the root like this example but **they do** have to be on the same server, if you put a full url in such as:

```
ErrorDocument 404 http://www.example.com/404.html
```

It will fail to return the 404 status and likely return 301 or 200. There are ways around it but I advise you to use local files.

```
##Error Handling - Note to preserve error stratus DO NOT use full  
URLs##
```

```
ErrorDocument 401 /401.html  
ErrorDocument 403 /403.html  
ErrorDocument 404 /404.html  
ErrorDocument 400 /400.html
```

The codes in my example are the more common ones however you can use as many or few as you like, you can even send people to the same error page, a full list of error codes is available on Wikipedia here however the ones you should learn are;

- 404 – Not Found;  
The file is missing or could not be accessed
- 401/403 – Unauthorized/Forbidden;  
You are not allowed to access the content, entering the corrected details may fix a 401 but not a 403
- 400 – Bad Request;  
Something is wrong with the syntax of your request, usually a typo in the url.
- 500 – Internal Server Error;  
Frequently caused when playing around with htaccess if you are not careful, it indicates a generic server error.

Really 404 errors are way more common than any other error so if you just make an alternate page for this problem then you should be fine.

Constructing a good 404 page is a topic in itself however I would advise you to include links to the homepage, sitemap, a few recommend important pages of the site and a contact form should the user be getting the message repeatedly.

## Block Index Displays

This one is more about general security and usability than SEO, and as we all know a secure site is better than an unsecure one. The code snippet is very simple and will stop people being able to access those horrible lists of files when you don't put an index.html or php

```
##Block Index Display##  
Options All -Indexes
```

## Set Default Index File

Another less SEO focused one but useful none the less, this code will enable you to change which page is treated as the directory index. For example you may want people to temporarily land on a promotional page first or maybe you are using index.php for something else, either way this code is handy to have around.

All you need to do is modify the file to be whatever your new index is, the file type doesn't have to be HTML.

```
##Set Default Index Files (Recourses)##  
DirectoryIndex newindex.htm
```

## Protect Htaccess

Your htaccess file is important but also potentially vulnerable; this bit of code will stop people accessing it.

```
##Secure htaccess file ##  
<Files .htaccess>  
Order Allow,Deny  
Deny from all  
</Files>
```

Additionally if you're using Wordpress then this will protect your wp-config file

```
##Secure wpconfig.php ###  
<Files wp-config.php>  
Order Allow,Deny  
Deny from all  
</Files>
```

## Password Protect a Location

You may wish to protect certain segments of your site, while this can easily be achieved inside a CMS static user may struggle installing a script but fear not, htaccess can help.

There is actually a very cool little tool to create these files which can be found here <http://www.tools.dynamicdrive.com/password/> however I am going to through the code for you to be able to do it yourself in case any problems occur.

Using htaccess and a similar file called htpasswd you can block public access to any file or folder on your website. Let's look at the htpasswd file first.

Using the same techniques as creating htaccess make a text file with no name and the htpasswd filetype. This file will house a list of usernames and passwords that can access the secure sections, they are formatted like this.

```
MyUsername:MyPassword
```

Remember that CaSe does matter so you will need to remember what you use and it's also a very good idea to encrypt the password, the link I gave you above can do this.

You need to upload the htpasswd file to a relevant place on the a server with an FTP client, it should be a non public accessible folder (so don't put it in /www/ or /htdocs/) and you'll need to make a note of the path from the root so it may be something like "/private/script/.htpasswd".

Once this is uploaded you're ready to add a new section to htaccess.

```
AuthName "Restricted Area"  
AuthType Basic  
AuthUserFile /home/mysite/.htpasswd  
<Files secretstuff />  
require valid-user  
</Files>
```

## Block Hotlinking

As an affiliate your hosting and bandwidth are precious, the last thing you need is someone taking your site down because they are copy and pasting an image to a 1000 locations. In the early days of the net this used to be a major problem and was known as “Hotlinking”.

It’s not such a serious problem these days but for some hosting packages it’s still a threat, if you want to stop people linking to resources on your site then you can use the following code, you’ll need to add your own domain as an exception of course and remember that this will block *all* other sites so there may be the odd extra site you need to add as an exception (i.e. forums).

```
## Block Hotlinking ##
##From all Sites Except Mine##
RewriteEngine On
RewriteCond %{HTTP_REFERER} !^http://(.+\.)?mysite\.com/ [NC]
RewriteCond %{HTTP_REFERER} !^$
RewriteRule .*\.(\.pe?g|.gif|.bmp|.png)$ - [F]
```

Unlike other code you may come across this accounts for subdomains of your own domain, very important for affiliates running multilingual sites!

If it’s just one or two specific sites (usually social ones) that are causing the problem then this code is a little friendlier

```
## Block Hotlinking ##
##From Specific Sites##
RewriteEngine On
RewriteCond %{HTTP_REFERER} ^http://(.+\.)?myspace\.com/ [NC,OR]
RewriteCond %{HTTP_REFERER} ^http://(.+\.)?friendfeed\.com/ [NC,OR]
RewriteCond %{HTTP_REFERER} ^http://(.+\.)?livejournal\.com/ [NC]
RewriteRule .*\.(\.pe?g|.gif|.bmp|.png)$ - [F]
```

Generally I don’t recommend you using this code unless you have actually trouble with frequent hot linkers but it’s a useful one to have in emergencies.

## Force a File to “Save As…”

This turns out to be surprisingly useful if you are running your own blog or content site, it will force certain file types to bring up the “save as” dialogue box instead of letting the browser load them, perfect for resource directories.

```
## Force a file to download with a "Save As" ##  
AddType application/octet-stream doc .mov .avi .pdf .xls .mp4
```

Just remember it will apply globally if you put it in your root htaccess so this one is probably better on its own in a subdirectory.

## Redirect Everyone but Selected IPs

This is a slightly slapdash way to do it but this will allow you to redirect all your visitors bar specific IPs to your site. It's perfect for when you are doing a long, maybe unplanned maintenance job and need you and your team to look around the site before letting general browsers in.

```
## Redirect Everyone but Selected IP Address ##  
ErrorDocument 403 http://www.myoldsite.com  
Order deny,allow  
Deny from all  
Allow from 11.111.11.111
```

You'll need to change the IP address for your own (and copy the line for each different IP in your team) and remember to turn it off when you're done! The redirect selection below contains a more professional way to do a maintenance redirect but this one can be useful when you have a bad host or are in need of a quick fix.

## Block Selected IPs

Got an annoying commenter leaving 1000 spam comments a day? Being cyber stalked or pestered for some reason?

One sure fire way to stop them is to block them accessing your site at all on an IP level, if you know their IP (from your logs) then simply add it to this code snippet to stop them accessing your site full stop.

```
## Block IP Address ##  
order allow,deny  
deny from 127.0.0.1  
deny from 127.0.0.2  
deny from 127.0.0.3  
allow from all
```

## Section 2: Redirects & Rewrites

Most SEOs know Htaccess for redirects. Redirects are a handy tool for telling Google you have moved a page and ensuring your visitors get there too.

They also allow you to rewrite and reform URLs moving from ugly automated ones to keyword friendly static ones, they are essential for day to day maintenance of larger sites and understanding how to correctly manipulate redirects is one of the most important additional techniques you can devote time to learning as an SEO.

### **Redirect http:// Version to www.**

One of the oldest SEO indexing problems is getting your site indexed in both the `http://mydomain.com` and `www.mydomain.com` variations. This can in some cases end with entire sites indexed twice, not ideal for any SEO.

Often referred to as a canonical redirect this piece of code will take any request that is made for the http versions of your site and turn it into a www. request.

It is worth mentioning that most of the major search engines have other ways for accounting for this common problem these days but in my opinions it's always better to solve a problem yourself than rely on Google or Microsoft to solve it for you.

```
## Redirect non-www to the www version ##  
RewriteCond %{HTTP_HOST} ^example.com [NC]  
RewriteRule ^(.*)$ http://www.example.com/$1 [L,R=301]
```

All you need to do is modify the `example.com` to fit your own domain.

### **Redirect Index File to Root**

Another frequent duplicate index problem occurs when search engines index the default file of a directly, most frequently the home in its file and root form causing `www.example.com/` and `www.example.com/index.php` to both be indexed.

The last page in the world you want duplicating is your home page so this simple snippet of code will ensure requests for the file are redirect to root. It'll also look a bit tidier for the user.

```
## Redirect index to root ##  
RewriteCond %{THE_REQUEST} ^[A-Z]{3,9}\ /.*/index\.\.(html|php)\ HTTP/  
RewriteRule ^(.*)index\.\.(html|php)$ /$1 [R=301,L]
```

With this piece of code you will need to modify both the name of the index file (99% of the time this is just `index`) and the extension (usually `html`, `htm` or `php`).

## Redirect All Pages to SSL

While not the most useful code for every affiliate, this code can still be very useful for secure subsections of a site, it will move everything in the folder and below into https however so it's not a good idea to use this one in your root htaccess.

This code won't magically set up https for you, you'll still need to get a security certificate and configure the server.

```
## Redirect all Pages to Secure ##  
RewriteCond %{HTTPS} !on  
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}
```

## Maintenance Redirect

I mentioned a slightly slapdash way to temporarily take your site down in an emergency earlier but this code is much more polished maintenance redirect, you will however need to do some preparation.

First create a downtime page, a simple branded page explaining to visitors that your site will be back up in x hours or days, I suggest you do it as static html or php but there's not really any restrictions.

Upload this file as you normally would to the root of your website folder and then use the following code to redirect all your visitors to it while you work in the background.

It will provide 302 status redirect for request pages meaning Google will not index the changes and your visitors will still see a valid page and not a 403 or 404.

```
## Redirect all to Maintenance Page ##  
RewriteBase /  
RewriteCond %{REMOTE_HOST} !^111\.11\.11\.11$  
RewriteCond %{REQUEST_URI} !^/downtime\.html$  
RewriteRule ^(.*)$ /downtime.html [R=302,L]
```

With this code you'll need to insert your IP into the number range, and the filename of your maintenance page into the last two lines, don't forget to change the file type if you page isn't html.

It's a good idea to set yourself a reminder to remove/comment out this code when you're done, it's very easy to forget you're the exception to the rule and other people can't see your site!

## Standard Permanent (301) Redirect – Single Page

The bread and butter of an SEO htaccess the 301 redirect instructs Google and browsers that the page they requested has been permanently moved and offers another specific location.

It is useful for when you decide to remove pages from your site or want to rename their urls and causes minimal disruption to your SEO efforts.

One thing you should know is that a 301 redirect is not watertight, there is always [Google] PR leakage from a redirect, exactly how much is constantly in debate in the SEO community and knowing Google is probably constantly in flux anyway. The bottom line is it's much better than visitors hitting a 404 page and those external links will at least still be coming into valid pages.

The 301 is still a versatile creature however allowing you to redesign sites with minimal SEO disruption and fix pesky broken links coded in other peoples sites and the best thing about it is that it's extremely easy to implement.

```
## Standard 301 Redirect for Permanent Changes ##  
redirect 301 /old-page.html http://www.example.com/newpage.htm
```

All you need to change is the two urls, the first one is the old page, and relative to the htaccess (usually the root) and the second is a full link to the new location. You can have as many lines as you want redirecting but you shouldn't redirect the same url more than once.

On larger sites I often group my redirects by date in order to make them easier to manage in the long run, remember you can add # to turn a line into a comment.

## Standard Permanent (301) Redirect – Subdirectory

You may also wish to redirect entire subdirectories of your site, maybe you have moved your blog into the root or got rid of an old article section, and there are two ways to do this.

```
## Redirect for a Directory - Preserves Request ##  
RedirectMatch 301 ^/old-folder/(.*) http://www.example.com/$1
```

This first example will *preserve the requested file*; that means if someone asks for `mysite.com/blog/example.html` it will redirect the user to `mystie.com/example.html`. It's incredibly useful for when you decide a particular set of files shouldn't be in a subfolder anymore or for when you want to move a blog or article section to root.

Please be aware this won't move the files for you, you will need to make sure the new files are there or you'll get a 404.

The other way to redirect a subdirectory should be used when you have *removed* a subdirectory and you want to ensure you catch any user or link that goes to the old subdirectory and bring them to an update page or just your index.

```
## Redirect a Directory - Does not Preserve Request ##  
RedirectMatch 301 ^/old-folder/(.*)$ http://www.example.com/new-page.php
```

Again customising these code sections is easy, you just need to fill in your old folder and the new locations, if you want to redirect to a new subfolder just change the ending url to include that subfolder but ensure you keep the \$1 in the first example.

## Temporary Permenant (302) Redirect

The three code examples above can also be easily modified to become temporary redirects. The big difference between a 301 and 302 redirect is how machines treat them. Users will see no difference but for Google and the other search engines a 302 sends a very different message.

For SEO the key factor is indexing, it's unlikely Google will update its index for a 302 redirect where it is extremely likely it will update it for a 301. To that end you should use 302 redirects only when the changes you made are temporarily and you want Google to largely ignore them.

To turn a redirect into 302 simply change the number at the start, it will work on almost any redirect code snippet where we use 301, for example;

```
## Standard 302 Redirect for Temporary Changes ##  
redirect 302 /old-page.html http://www.example.com/newpage.html
```

## Change File Type Requests

Every now and again a situation occurs where you end up changing file types for some reason, maybe a new image compression is better, maybe you started using a special include that needs php, either way it can be an *extremely* big tasks manually redirecting all your old html pages to php but htaccess can help.

This piece of code comes with a warning, **it will redirect all requests** that means you need to be sure you actually want to do this before implementing it, ensure you have no stray files of the old type and that all the new pages are uploaded.

```
## Code to make all HTML requests become PHP requests - BE CAREFUL WITH  
THIS ONE ##  
RedirectMatch 301 (.*)\.html$ http://www.example.com$1.php
```

To customise this code simply swap out the two file types, in this example the html is the old file type and php is the new one, you will of course also need to put your own domain in.

## Redirect All Subfolders but Keep Root Intact

This was a special request I had from a Twitter follower, he was moving large portions of his domain but wanted to keep his index and root intact for the time being, it's a slightly more unusual one but you never know when it may come in handy, it could be useful for when you decide to move a blog to its own domain.

You'll just need to customise the URL for the new site, please be aware this code is designed to work in the root and that it also preserves requests.

```
## Redirect all Subfolders to New Domain but Keep Root Intact ##  
RewriteRule ^([^/]+)\/(.*)$ http://www.example.com/$1/$2/$3 [R=301,L]
```

## Redirect an Entire Domain

Sometimes you decide it's time for a new domain but as an SEO you want to keep all the effort you've put into your current domain, well htaccess can help here. As I have already mentioned 301 redirects are not watertight for PR but they are the best option available.

If you have Google Webmaster Tools installed on your old domain (which you should) then you can also inform Google that you are moving domains, you still need to do a 301 but this should strongly help your case.

There are two main ways to do this, preserving and not preserving requests. If your new domain is going to have the exact same site, pages and resources it would be wise to preserve your requests this means a user asking for /best-page.html on your old site will still see best-page.html on your new site.

```
## Redirect an Entire Domain - Preserve Request ##  
RewriteRule (.*) http://www.newdomain.com/$1 [R=301,L]
```

If you want to completely redirect your domain with no preservation of request then use.

```
## Redirect an Entire Domain - No Preservation ##  
Redirect 301 / http://www.newdomain.com
```

Really, you should individually redirect sections of your website to relevant sections of your new site when moving a valuable SEO domain but time is not always on hand for such a job so those two redirects are you other options.

## Rewrite a Dynamic URL

OK time for the really juicy htaccess tricks, rewriting a dynamic URL to a static one. I want to bring back a point I made in the introduction here, this technique is not perfect and while it is better for SEO and usability than those ugly dynamic ones it's much better to simply build/choose a system that generates proper URLs.

No system is perfect however so let's look at how we rewrite a URL and most importantly, redirect the original request to the pretty new URL.

To rewrite a URL you need to understand the structure of the variable one you're trying to rewrite, most dynamic URLs look something like this;

<http://www.mysite.com/posts.php?category=widgets>

It's made up of the domain, the file and the query string which is itself made of two parts, field and value. Here is that broken down.



What you want to do to make this an "SEO friendly URL" is take away the query string and use it to create a structured URL.

What we need to tell the server is to replace the query string section with something else.

In our first example we will look at making a URL that ends as a static file i.e. turning that URL into <http://www.mysite.com/posts-about-widgets.php>

```
RewriteRule ^posts-about-([^\./]+)\.php/?$ /posts.php?category=$1&foo=bar  
[L]  
RewriteCond %{QUERY_STRING} category=([^\./]+)  
RewriteCond %{QUERY_STRING} !foo=bar
```

OK here we have three lines; the first specifies the mapping of the old URL to the new

The second specifies where the 1<sup>st</sup> query string is

The third is a stop point so we can avoid redirect loops

So let's take a closer look at the first line, the first section lets us set the format for the new page, in our example it would make our new page <http://www.mysite.com/posts-about-widgets.php>. You can change the first half of this to whatever you need to make your URL make sense.

Notice the mess in brackets at the end? This is the *value* field of the query string being pulled in, that way your URL is still dynamic enough to relate to what it's getting from the database. In my example it's pulling in posts from the widgets category so I would obviously want the widgets keyword in my URL.

The second highlighted part in this line is simply the file type you want, I have chosen PHP but can have html or htm, whatever you need.

The third highlighted section contains the *original* URL and query string. Notice that it is just the *field* included; the *value* is drawn in as a variable.

The second line also contains a reference to the variable, you will need to change *field* to represent your own URL but leave the rest, and essentially this code is saying "any number of different characters and numbers"

OK now we have our new page but more than likely your system will still be generating old ugly URLs so now we need to redirect people to the new page. We do this using a Rewrite Rule that replaces requests for the query string with your shiny new URL.

```
RewriteRule ^posts\.php$ /posts-about-%1\.php? [L,R=301]
```

Relatively simple right? You just need to match up your old URL into the first section and your new URL into the second and hey presto, you've got a dynamic redirect from the old variable to the new page.

Let's put all that code together in an easy copy paste segment so we can understand it in the htaccess file itself. I have replaced the example with placeholder text to help us remember the parts each statement refers to.

```
## Rewrite Dynamic URL to A Static Page ##  
RewriteRule ^new-page-([^\./]+)\.php/?$ /old.php?field=$1&foo=bar [L]  
RewriteCond %{QUERY_STRING} field=([^\./]+)  
RewriteCond %{QUERY_STRING} !foo=bar  
RewriteRule ^old\.php$ /new-page-%1.php? [L,R=301]
```

Maybe however you want to go to a different type of URL maybe (going back to our example) you want `http://www.mysite.com/posts/widgets/` instead of that `.php` page. Well that's possible too, all we need is a slight modification to the code.

```
## Rewrite Dynamic URL to A Static SubFodler ##  
RewriteRule ^new-folder/([^\./]+)/?$ /old.php?field=$1&foo=bar [L]  
RewriteCond %{QUERY_STRING} field=([^\./]+)  
RewriteCond %{QUERY_STRING} !foo=bar  
RewriteRule ^old\.php$ /new-folder/%1/? [L,R=301]
```

It's mostly the same but this time we name the first subfolder manually and allow the *field* to be the second subfolder.

### Section 3: Speed & Tweaking

The final section of this guide looks at a few tweaking codes to improve the speed and performance of your website. Many of the code snippets in this section rely on certain modules or permissions being active and that means that different servers and hosts can behave differently.

Unfortunately it's impossible for me to account for every hosting setup, check with your host if you find certain code snippets don't work, maybe they will have an alternate suggestion or maybe they are blocking the module on purpose.

Site speed and SEO are becoming increasingly related, while it's always been important to have a reliable site now, more than ever load time counts too. Do what you can code side with includes and CDN hosting on large sites to reduce load time but compression and caching can help too and these can be activated with htaccess.

#### Activate GZIP Compression

GZIP compression is a server side function that compresses web pages and scripts before they are sent to the browser in order to minimize what is downloaded.

The technology was originally a bit janky and often caused problems for sites, especially secure sections and payment gateways. It is however much better these days, most browsers, even older ones, accept GZIP and while more complex function sites can still have issues most of the sites you will build as an affiliate should be fine.

There are different ways to activate GZIP depending on your server configuration, I am going to give 4 variations, all of which work in different circumstances, I personally prefer the first bit of code (credit to @jhuskisson on that) but if you find it doesn't work just try one of the others.

You don't need to customise the code for these so just copy and paste away.

This website will let you know if GZIP is working on your site and how much time/space you're saving!

<http://www.gidnetwork.com/tools/gzip-test.php>

```
##Enable GZIP Version 1##
php_value output_handler ob_gzhandler
css_value output_handler ob_gzhandler
js_value output_handler ob_gzhandler

##ENABLE GZIP Version 2##
<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 rsphheader ^Content-Encoding:.*gzip.*
</IfModule>

##Enable GZIP Version 3##
<Files *.php>
SetOutputFilter DEFLATE
</Files>
<Files *.js>
SetOutputFilter DEFLATE
</Files>
<Files *.css>
SetOutputFilter DEFLATE
</Files>
<Files *.html>
SetOutputFilter DEFLATE
</Files>

##Enable GZIP Version 4##
AddOutputFilterByType DEFLATE text/html text/plain text/xml
application/xml application/xhtml+xml text/javascript text/css
application/x-javascript
BrowserMatch ^Mozilla/4 gzip-only-text/html
BrowserMatch ^Mozilla/4.0[678] no-gzip
BrowserMatch bMSIE !no-gzip !gzip-only-text/html
```

Please remember you only need one, not all four!

## Activate Caching

Caching is a way to stop repeat visitors completely re-downloading every element of your site, if you are using Wordpress then there are some pretty good plugins to handle this so I suggest you go and download one of those instead (such as [WP Hyper Cache](#)).

If you have a static site you can still use caching with htaccess.

A few notes, caching can “mess” with things if you’re not careful. If you have a custom CMS or some kind of dynamic content and you are caching the wrong thing for too long you may notice your site not updating, I recommend you use caching but check with your development team (if you work with one) to ensure you’re not going to be battling against a system.

If you want a more advanced caching command or more control I recommend using the script in this WMM thread but obviously make sure you read up on what you’re doing.

<http://www.webmasterworld.com/apache/3999528.htm>

The code below has different cache settings for different types of files, it stores some longer than others on the grounds you are likely to update pages but not say icons. Remember you can always disable the code.

```
##Enable Caching##

## Files to Cache for One Month
<FilesMatch "\.(flv|gif|jpg|jpeg|png|ico|swf)$">
Header set Cache-Control "max-age=2592000"
</FilesMatch>

## Files to Cache for One Week
<FilesMatch "\.(js|css|pdf|txt)$">
Header set Cache-Control "max-age=604800"
</FilesMatch>

## Files to Cache for One Day
<FilesMatch "\.(html|htm)$">
Header set Cache-Control "max-age=43200"
</FilesMatch>

## Disable cache for script files
<FilesMatch "\.(pl|php|cgi|spl|scgi|fcgi)$">
Header unset Cache-Control
</FilesMatch>
```

With GZIP and Caching turned on you should see a significant increase in site performance. Remember Google WMT has a graph for this in the Labs sections.

## Minimise 404s with CheckSpelling

CheckSpelling is a Linux Server Module that is an unsung hero for SEO's. Basically this function kicks in when a URL is requested that would generate a 404, it checks the server for similar URLs and suggests/redirects user to them. For example a user types in this to their browser:

```
http://www.mysite.com/caje/red.html
```

What they meant was;

```
http://www.mysite.com/cake/red.html
```

CheckSpelling would pick up this slight mistake and fix it (unless of course there really *was* a page called `caje/red.html`). The great thing is that it also works for the CaSe sensitive Linux URLs i.e.

```
http://www.mysite.com/cake/RED.html
```

is actually a different page to

```
http://www.mysite.com/cake/red.html
```

But with CheckSpelling enabled it would redirect.

This works for all requests not just type ins so it can really save you if another webmaster or even you have made typos in links on the site.

It's not flawless and it does increase processor load (for which some hosts deactivate it) but if you can use it I strongly recommend CheckSpelling for your site.

```
##Make Linux correct case sensitive urls/slight typos in urls##  
CheckSpelling On
```

## Conclusion

This document contains a lot of htaccess tips and tricks but htaccess can do much, much more. If you are looking to learn more I recommend [WebmasterWorlds Apache Forum](#) and the [Official Documentation Site](#).

It's really important to try and understand how technology works if you really want to leverage it for your advantage so while everything here can be copied pasted and used without too much knowledge I encourage you to use it as a starting block to learn more and more about SEO and web server technology.

Always remember that in SEO the rules change, maybe in 6, 12 or 120 months time a 301 redirect will be the completely *wrong* thing to do in a site move, always keep up to date on what the current techniques are, in SEO this is one of the most important and most difficult tasks.

Most importantly of all good luck with all your SEO projects!

## Disclaimer

I have not written every bit of code in this document, I have accumulated it over the years as an SEO. Some of it I have tweaked, some I have been given when people helped me, some I have taken from other peoples guides and since forgotten and some I have constructed directly from documentation.

I am providing this guide as a collection of code snippets mainly aimed at small website owner and I'm sorry if it looks like I "stole" your code without credit. Please consider there is likely 1000s of copies of "your" code written by completely different people out there, htaccess is after all part of a configuration setting on an open platform.

It's also true I have slightly simplified some descriptions and technical terms for this guide; this is because I want it to be relatively easy to read so keep this in mind before emailing me a correction about the difference between URI and URL.