



Lesson #8: .htaccess

What is .htaccess?



- .htaccess (hypertext access) is the default name of Apache's directory-level configuration file. It provides the ability to customize configuration for requests to the particular directory.
- .htaccess is the file extension. It is not file.htaccess or somepage.htaccess, it is simply named .htaccess.
- .htaccess files are commonly used for...

Uses of .htaccess



- Authorization, authentication: .htaccess files are often used to specify the security restrictions for the particular directory, hence the filename "access". The .htaccess file is often accompanied by an .htpasswd file which stores valid usernames and their passwords.
- Customized error responses: Changing the page that is shown when a server-side error occurs, for example HTTP 404 Not Found.
- Rewriting URLs: Various server-side PHP scripts use .htaccess to rewrite "ugly" URLs to shorter and prettier ones.

Writing .htaccess



- htaccess files must be uploaded as ASCII, not BINARY. You may need to CHMOD the htaccess file to 644. This makes the file usable by the server, but prevents it from being read by a browser, which can seriously compromise your security.
- For example: If you have password protected directories, and a browser can read the htaccess file, then they can get the location of the authentication file and then extract the list to get full access to any portion that you previously had protected. There are different ways to prevent this, one being to place all your authentication files above the root directory so that they are not www accessible, and the other is through an htaccess series of commands.

Writing .htaccess



- Most commands in htaccess are meant to be placed on one line only, so if you use a text editor that uses word-wrap, make sure it is disabled or it might throw in a few characters that annoy Apache to no end, although Apache is typically very forgiving of malformed content in an htaccess file.
- Before you go off and plant htaccess everywhere, make sure you don't do anything redundant, since it is possible to cause an infinite loop of redirects or errors if you place something weird in the htaccess.

Scope of .htaccess



- htaccess files affect the directory they are placed in and all sub-directories, that is an htaccess file located in your root directory (yoursite.com) would affect yoursite.com/content, yoursite.com/content/contents, etc.
- This can be prevented (if, for example, you did not want certain htaccess commands to affect a specific directory) by placing a new htaccess file within the directory you don't want affected with certain changes, and removing the specific command(s) from the new htaccess file that you do not want affecting this directory. In short, the nearest htaccess file to the current directory is treated as the htaccess file.

Scope of .htaccess



- htaccess files affect the directory they are placed in and all sub-directories, that is an htaccess file located in your root directory (yoursite.com) would affect yoursite.com/content, yoursite.com/content/contents, etc.
- This can be prevented (if, for example, you did not want certain htaccess commands to affect a specific directory) by placing a new htaccess file within the directory you don't want affected with certain changes, and removing the specific command(s) from the new htaccess file that you do not want affecting this directory. In short, the nearest htaccess file to the current directory is treated as the htaccess file.

Warning



- Some hosts do not allow use of htaccess files, since depending on what they are doing, they can slow down a server overloaded with domains if they are all using htaccess files.
- You need to make sure you are allowed to use htaccess before you actually use it.
- Some things that htaccess can do can compromise a server configuration that has been specifically setup by the admin, so don't get in trouble.



Error Handling



- In order to specify your own ErrorDocuments, you need to be slightly familiar with the server returned error codes (see next slide).
- You do not need to specify error pages for all of these, in fact you shouldn't. An ErrorDocument for code 200 would cause an infinite loop, whenever a page was found...this would not be good.



HTTP Error Codes



- Here a few of the most useful HTTP error codes:
- 200 OK
- 301 Moved Permanently
- 302 Moved Temporarily
- 400 Bad Request
- 401 Authorization Required
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error

Error Handling



- You will probably want to create an error document for codes 404 and 500, at the least 404 since this would give you a chance to handle requests for pages not found. 500 would help you out with internal server errors in any scripts you have running. You may also want to consider ErrorDocuments for 401 - Authorization Required (as in when somebody tries to enter a protected area of your site without the proper credentials), 403 - Forbidden (as in when a file with permissions not allowing it to be accessed by the user is requested) and 400 - Bad Request, which is one of those generic kind of errors that people get to by doing some weird stuff with your URL or scripts.

Error Handling



- You will probably want to create an error document for codes 404 and 500, at the least 404 since this would give you a chance to handle requests for pages not found. 500 would help you out with internal server errors in any scripts you have running. You may also want to consider ErrorDocuments for 401 - Authorization Required (as in when somebody tries to enter a protected area of your site without the proper credentials), 403 - Forbidden (as in when a file with permissions not allowing it to be accessed by the user is requested) and 400 - Bad Request, which is one of those generic kind of errors that people get to by doing some weird stuff with your URL or scripts.

Error Handling



- Example:

```
ErrorDocument 400 /errors/badrequest.html
```

```
ErrorDocument 401 /errors/authreqd.html
```

```
ErrorDocument 403 /errors/forbid.html
```

```
ErrorDocument 404 /errors/notfound.html
```

```
ErrorDocument 500 /errors/serverr.html
```

- You can use html instead of a file reference. Make sure you do it on one line per error though.

Password Protection



- Ever wanted a specific directory in your site to be available only to people who you want it to be available to? Ever got frustrated with the seeming holes in client-side options for this that allowed virtually anyone with enough skill to mess around in your source to get in? htaccess is the answer!
- The first thing you will need to do is create a file called .htpasswd. It will contain the list of user names and passwords (encrypted).

wsabstract:y4E7Ep8e7EYV

Password Protection



- Create and .htaccess file with the following commands:

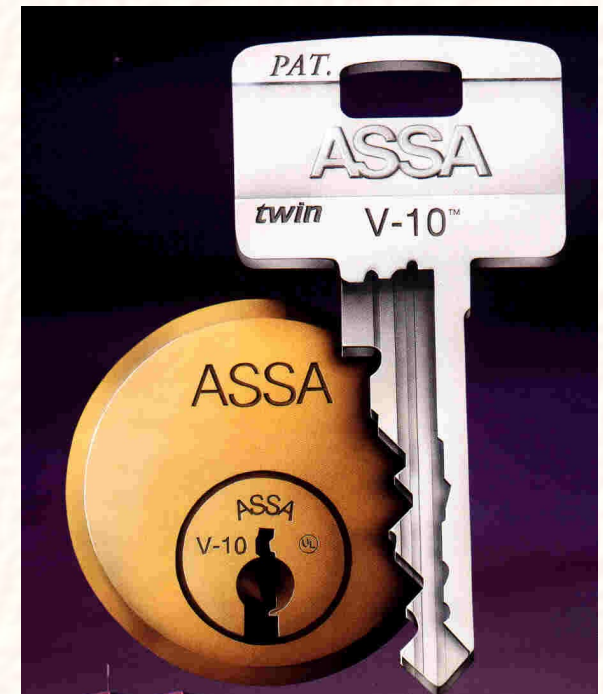
```
AuthName "Restricted Area"
```

```
AuthType Basic
```

```
AuthUserFile /home/mysite/.htpasswd
```

```
AuthGroupFile /dev/null
```

```
require valid-user
```



Enabling SSI on host



- Many people want to use SSI, but don't seem to have the ability to do so with their current web host. You can change that with htaccess. A note of caution first... definitely ask permission from your host before you do this, it can be considered 'hacking' or violation of your host's term of service!

```
AddType text/html .shtml
```

```
AddHandler server-parsed .shtml
```

```
Options Indexes FollowSymLinks Includes
```

```
AddHandler server-parsed .html
```

```
DirectoryIndex index.shtml index.html
```

Optional!

Blocking users by IP



- In your htaccess file, add the following code (changing the IPs to suit your needs) to block undesirable visitors.

```
order allow,deny
```

```
deny from 123.45.6.7
```

```
deny from 012.34.5.
```

```
allow from all
```

- You can also allow or deny by domain name rather than IP address

Blocking users by referrer



- Blocking users or sites that originate from a particular domain is another useful trick of .htaccess. Lets say you check your logs one day, and see tons of referrals from a particular site, yet upon inspection you can't find a single visible link to your site on theirs. The referral isn't a "legitimate" one, with the site most likely hot linking to certain files on your site such as images, .css files, or files you can't even make out. Remember, your logs will generate a referrer entry for any kind of reference to your site that has a traceable origin.
- So, to deny access all traffic that originate from a particular domain (referrers) to your site, use the following code:

Blocking users by referrer



- Block traffic from a single referrer:

```
RewriteEngine on
```

```
# Options +FollowSymlinks
```

Uncomment
if error 500

```
RewriteCond %{HTTP_REFERER} badsite\.com [NC]
```

```
RewriteRule .* - [F]
```

Not case
sensitive

Blocking users by referrer



- Block traffic from a multiple referrers:

```
RewriteEngine on
```

```
# Options +FollowSymlinks
```

```
RewriteCond %{HTTP_REFERER} badsite\.com [NC,OR]
```

```
RewriteCond %{HTTP_REFERER} anotherbadsite\.com
```

```
RewriteRule .* - [F]
```

For
multiple
domains



Blocking bad bots and site rippers

- The definition of a "bad bot" varies depending on who you ask, but most would agree they are the spiders that do a lot more harm than good on your site (ie: an email harvester). A site ripper on the other hand are offline browsing programs that a surfer may unleash on your site to crawl and download every one of its pages for offline viewing.

RewriteEngine On

```
RewriteCond %{HTTP_USER_AGENT} ^BlackWidow [OR]
```

```
RewriteCond %{HTTP_USER_AGENT} ^ChinaClaw [OR]
```

```
RewriteCond %{HTTP_USER_AGENT} ^DISCo [OR]
```

```
RewriteCond %{HTTP_USER_AGENT} ^Zeus
```

```
RewriteRule ^.* - [F,L]
```



Change your default directory page

- **Placing the above command in your htaccess file will cause this to happen: When a user types in yoursite.com, your site will look for filename.html in your root directory (or any directory if you specify this in the global htaccess), and if it finds it, it will load that page as the default page. If it does not find filename.html, it will then look for index.cgi; if it finds that one, it will load it, if not, it will look for index.pl and the whole process repeats until it finds a file it can use. Basically, the list of files is read from left to right.**

**DirectoryIndex filename.html index.cgi index.pl
default.htm**

Redirects



- Ever go through the nightmare of changing significantly portions of your site, then having to deal with the problem of people finding their way from the old pages to the new? It can be nasty. There are different ways of redirecting pages, through http-equiv, javascript or any of the server-side languages. And then you can do it through htaccess, which is probably the most effective, considering the minimal amount of work required to do it.

`Redirect /olddirectory/oldfile.html`

`http://yoursite.com/newdirectory/newfile.html`

Prevent viewing of .htaccess file



- It is possible to prevent an htaccess file from being viewed in this manner.

```
<Files .htaccess>
```

```
order allow,deny
```

```
deny from all
```

```
</Files>
```

Preventing Directory Listing



- Do you have a directory full of images or zips that you do not want people to be able to browse through? Typically a server is setup to prevent directory listing, but sometimes they are not. If not, become self-sufficient and fix it yourself:

IndexIgnore *

- The * is a wildcard that matches all files, so if you stick that line into an htaccess file in your images directory, nothing in that directory will be allowed to be listed.
- On the other hand, what if you did want the directory contents to be listed, but only if they were HTML pages and not images?

IndexIgnore *.gif *.jpg

- This would return a list of all files not ending in .jpg or .gif, but would still list .txt, .html, etc.

Preventing "hot linking"



- In the webmaster community, "hot linking" is a curse phrase. Also known as "bandwidth stealing" by the angry site owner, it refers to linking directly to non-html objects not on one own's server, such as images, .js files etc. The victim's server in this case is robbed of bandwidth (and in turn money) as the violator enjoys showing content without having to pay for its deliverance. The most common practice of hot linking pertains to another site's images. This code disables hot linking.

RewriteEngine on

```
RewriteCond %{HTTP_REFERER} !^$
```

```
RewriteCond %{HTTP_REFERER} !^http://(www\.)?  
mydomain.com/.*$ [NC]
```

```
RewriteRule \.(gif|jpg|js|css)$ - [F]
```



End of lesson