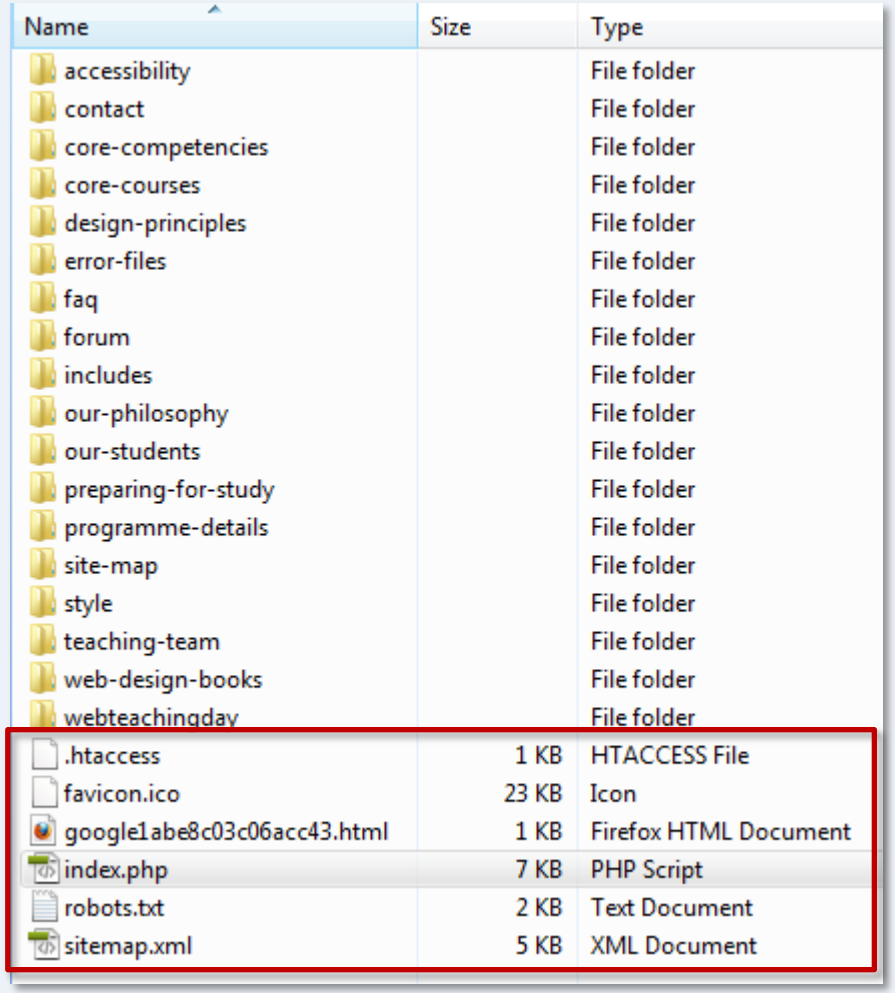
.htaccess and other oddities

Content Management

What are those files?

On the right is the file listing from the root directory of a website as seen in an FTP client. You may recognise *index.php* as being the website homepage, but what are all the other files?

This presentation aims to explain what they are and how they're used.



Name	Size	Type
accessibility		File folder
contact		File folder
core-competencies		File folder
core-courses		File folder
design-principles		File folder
error-files		File folder
faq		File folder
forum		File folder
includes		File folder
our-philosophy		File folder
our-students		File folder
preparing-for-study		File folder
programme-details		File folder
site-map		File folder
style		File folder
teaching-team		File folder
web-design-books		File folder
webteachingday		File folder
.htaccess	1 KB	HTACCESS File
favicon.ico	23 KB	Icon
google1abe8c03c06acc43.html	1 KB	Firefox HTML Document
index.php	7 KB	PHP Script
robots.txt	2 KB	Text Document
sitemap.xml	5 KB	XML Document

Content Management

THE .htaccess FILE

What is a .htaccess file?

- .htaccess is a *localised server configuration file* that can be used to override default server configuration settings.
- Originally, the file's primary purpose was to facilitate password protection to web folders; hence the name (hypertext access).
- On modern servers, .htaccess can be used to perform a range of tasks, including...

What can .htaccess do?

- **Custom Error Pages** – configure the use of custom error pages (e.g. 404 “page not found”).
- **Password Protection** – in combination with a .htpasswd file (containing encrypted username and password).
- **Redirection** – can redirect requests for one page or one folder to another (useful if your site changes).

What can .htaccess do?

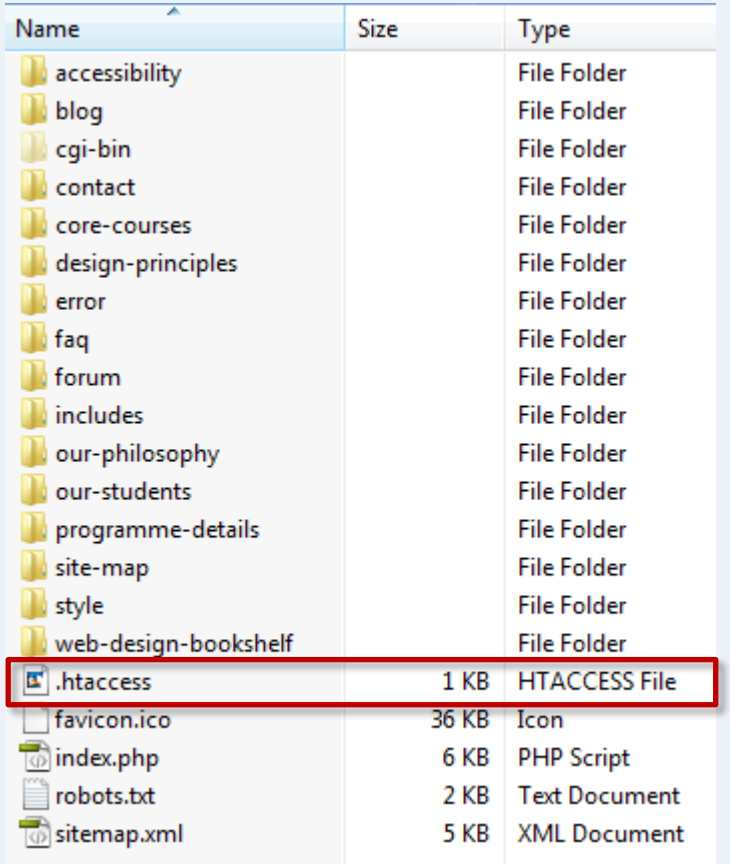
- **Rewrite URLs** – for consistency and for the benefit of search engines you can decide whether your site uses “www” or not. This is known as *URL Canonicalization*.
- **Prevent Hotlinking** – can prevent your web content (usually images) from being embedded in sites outside of your server.
- **Deny access** – block access to your website from specific IP addresses…

…and a great deal more.

URL Canonicalization

Where does .htaccess live?

- Websites do not need a .htaccess file but if they exist, they are placed in the root folder (using FTP).
- There may be additional .htaccess files if password protection is used. Each secure folder will have its own .htaccess file.
- The leading dot tells the web server that this is a hidden file, so you may need to tell your FTP client to display hidden files before you can see it.



Name	Size	Type
accessibility		File Folder
blog		File Folder
cgi-bin		File Folder
contact		File Folder
core-courses		File Folder
design-principles		File Folder
error		File Folder
faq		File Folder
forum		File Folder
includes		File Folder
our-philosophy		File Folder
our-students		File Folder
programme-details		File Folder
site-map		File Folder
style		File Folder
web-design-bookshelf		File Folder
.htaccess	1 KB	HTACCESS File
favicon.ico	36 KB	Icon
index.php	6 KB	PHP Script
robots.txt	2 KB	Text Document
sitemap.xml	5 KB	XML Document

What does .htaccess look like?

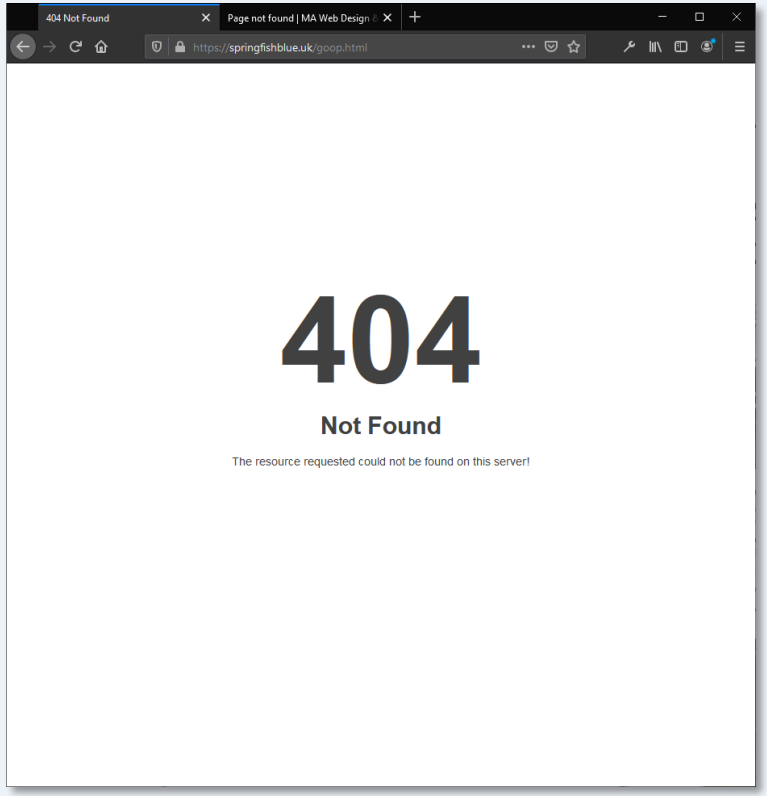
- .htaccess files are simple ASCII text files and can be viewed and edited in any text editor, even Notepad.
- The file contains one or more lines, known as “configuration directives”.

Content Management

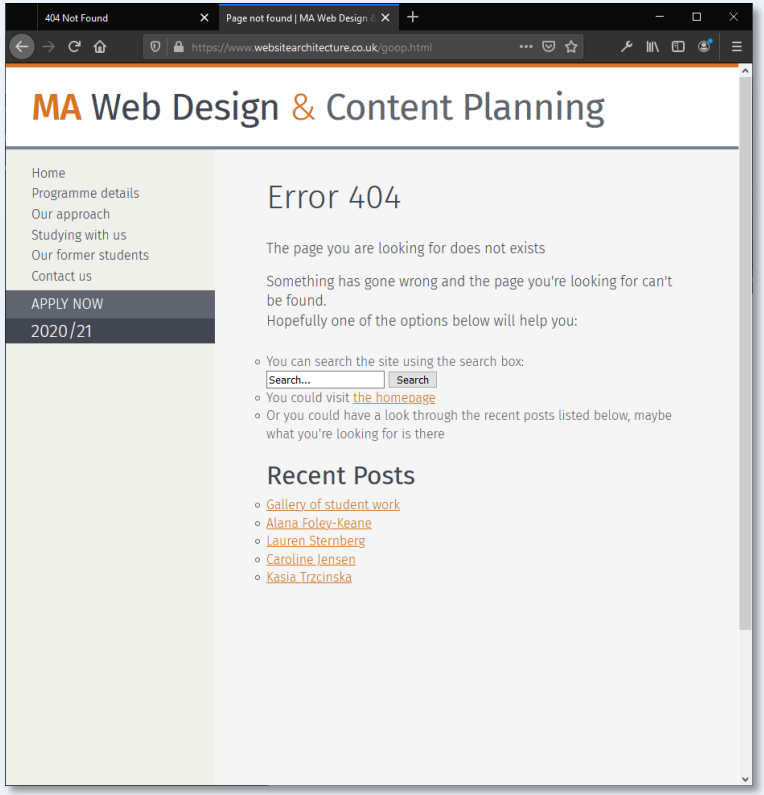
.htaccess: CUSTOM ERROR PAGES

Custom Error Pages

- All good websites make use of custom error pages; they are an excellent user experience tool.
- The most common error is the 404, “page not found”.



Default server error page



Custom error page

Server Errors

- When a hypertext request fails, the server determines the reason and allocates an error code.
- If a requested page cannot be found, the error code is 404.
- However, such codes are meaningless to users and should usually be avoided.
- Far better to use a useful *custom* error page to help the user recover from the error.

Creating a custom error page

- Custom error pages are no different to any other web page – they are built using HTML and CSS (and optionally PHP).
- The custom error page should look and feel like part of your site and should include plenty of navigation options – but not too many.
- You tell the server to serve your custom error page, rather than the default, by adding a directive to the .htaccess file.

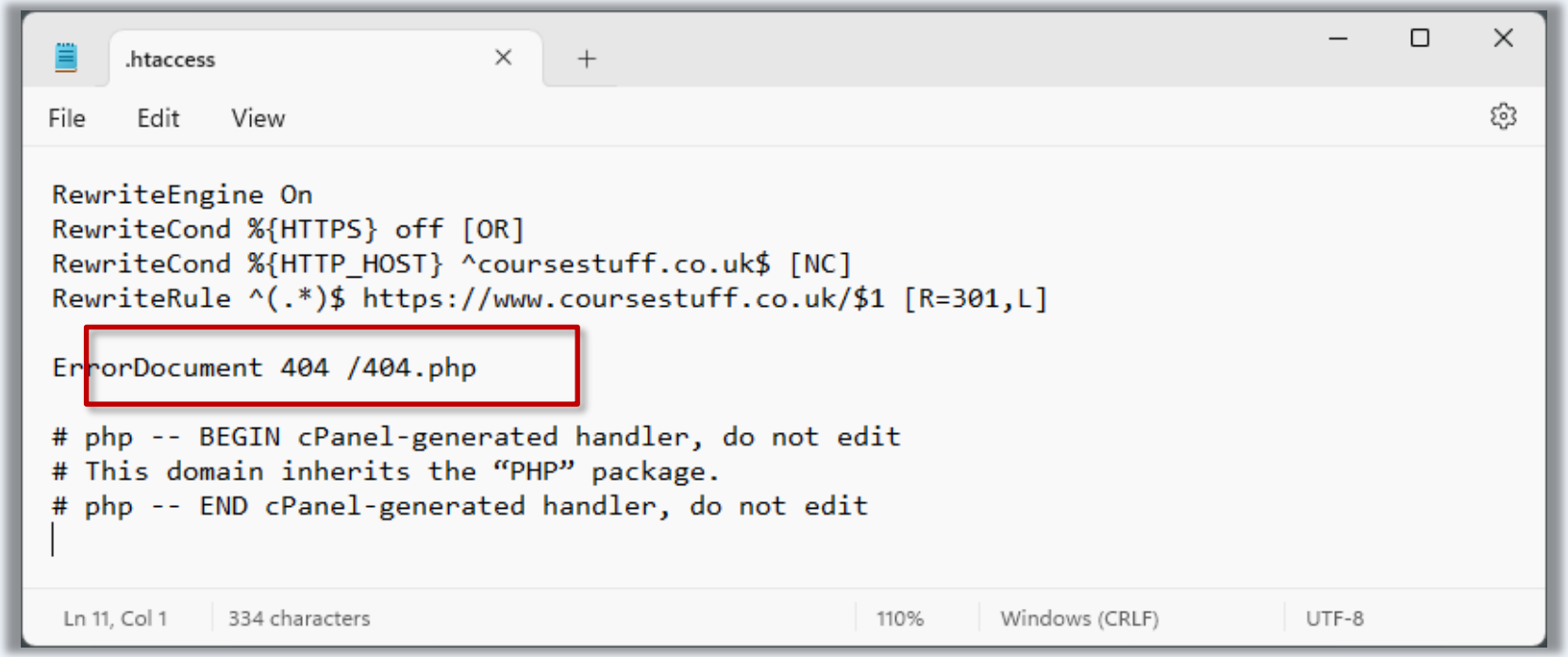
The ErrorDocument directive

```
ErrorDocument 404 /error/404.html
```

- *ErrorDocument* = the directive
- *404* = the error type code
- */error/404.html* = the path from the web root to the page that should be served in the event of this particular error. In this case, a file called *404.html* in a folder called *error* in web root.
- Each of the above elements is separated by a space.

The ErrorDocument directive

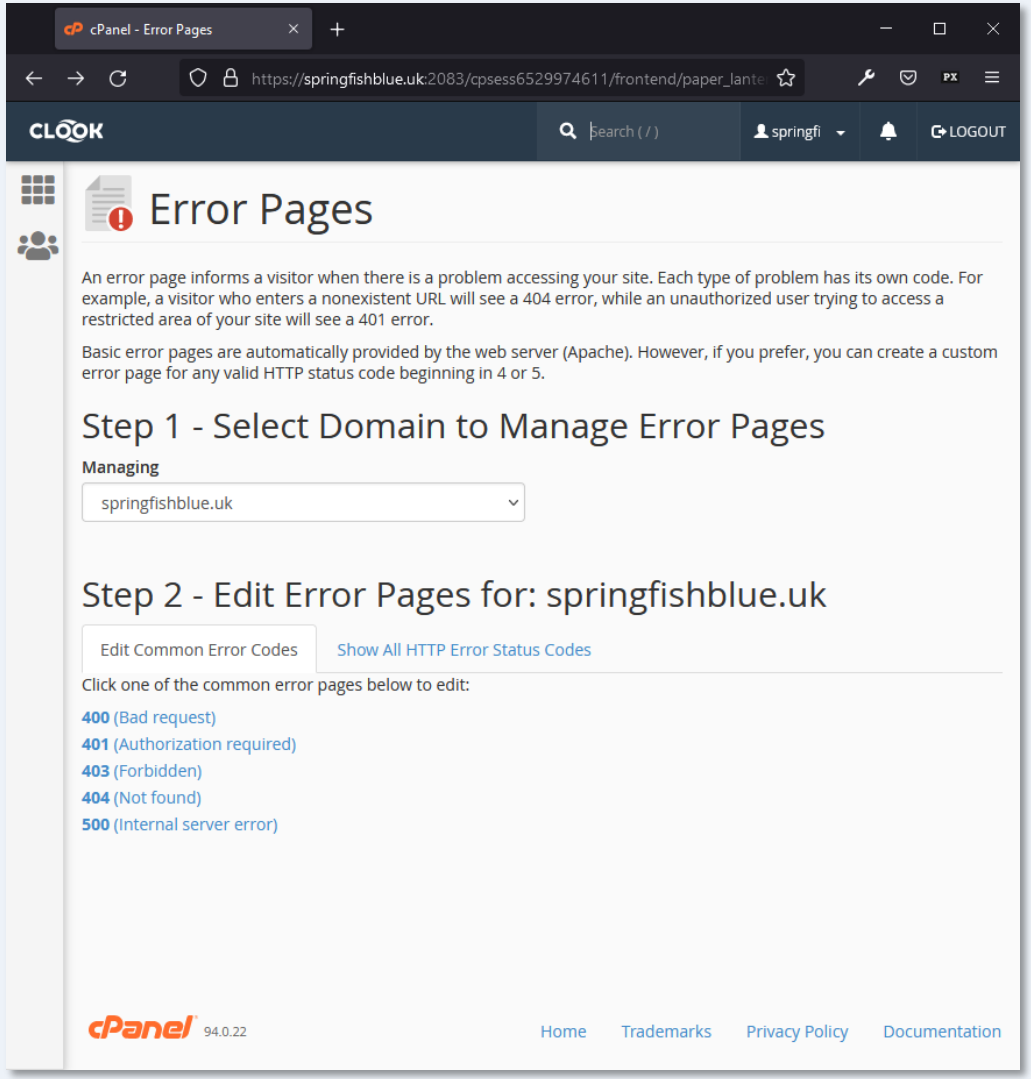
- Below is the .htaccess file at coursestuff.co.uk and you can see that in this case, the error file is in the root folder and is a PHP file (*404.php*).



Note: The leading slash before the filename tells the server to look in the root folder.

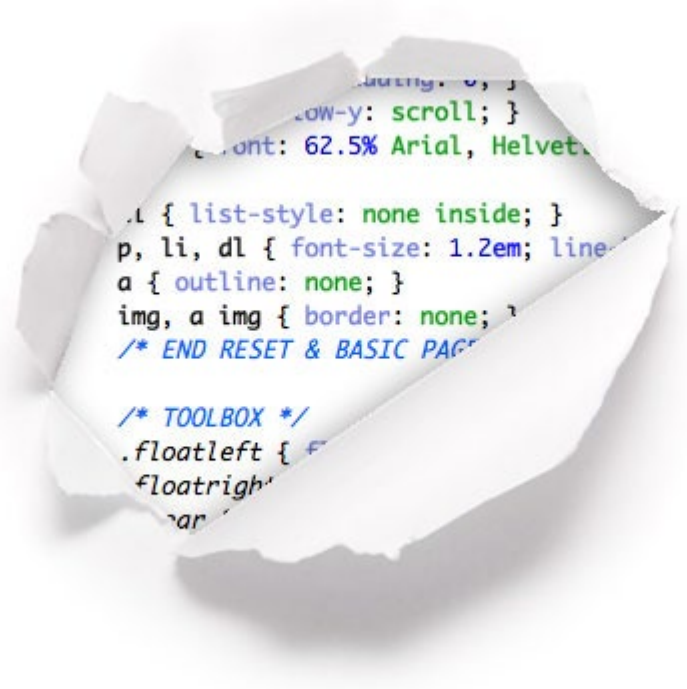
Hosting control panel

Some web hosting control panels allow you to set up error pages via a simple form. cPanel uses such a form which automatically creates the .htaccess file for you. However, the design options are limited, and it will create an error page for the entire account. If you want an error page for each project site or if you want full creative control, you'll need to do it manually with a .htaccess file in each project folder.



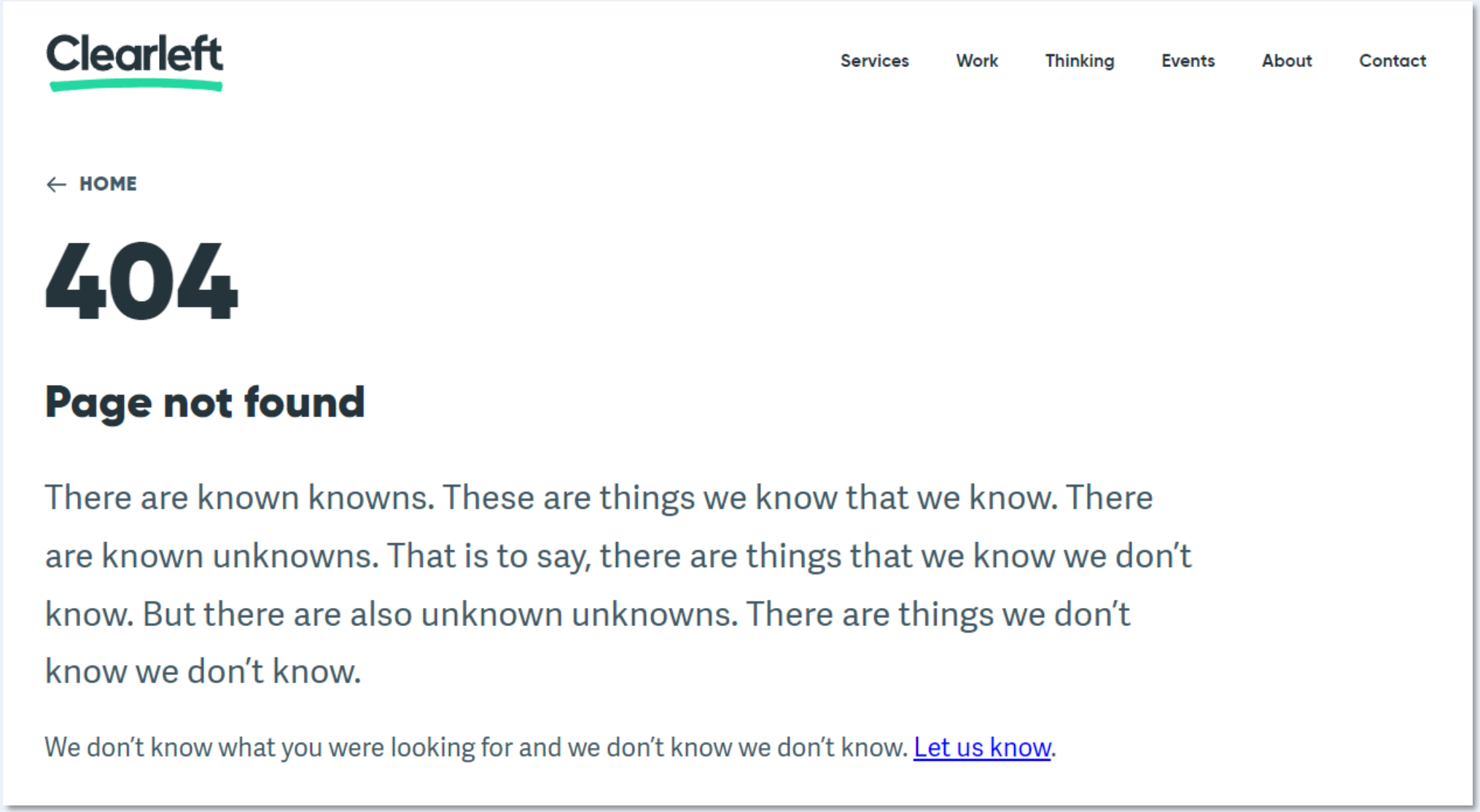
Humour?

- It has become somewhat of a tradition to inject some humour into your custom 404 error page – there are plenty of good examples...

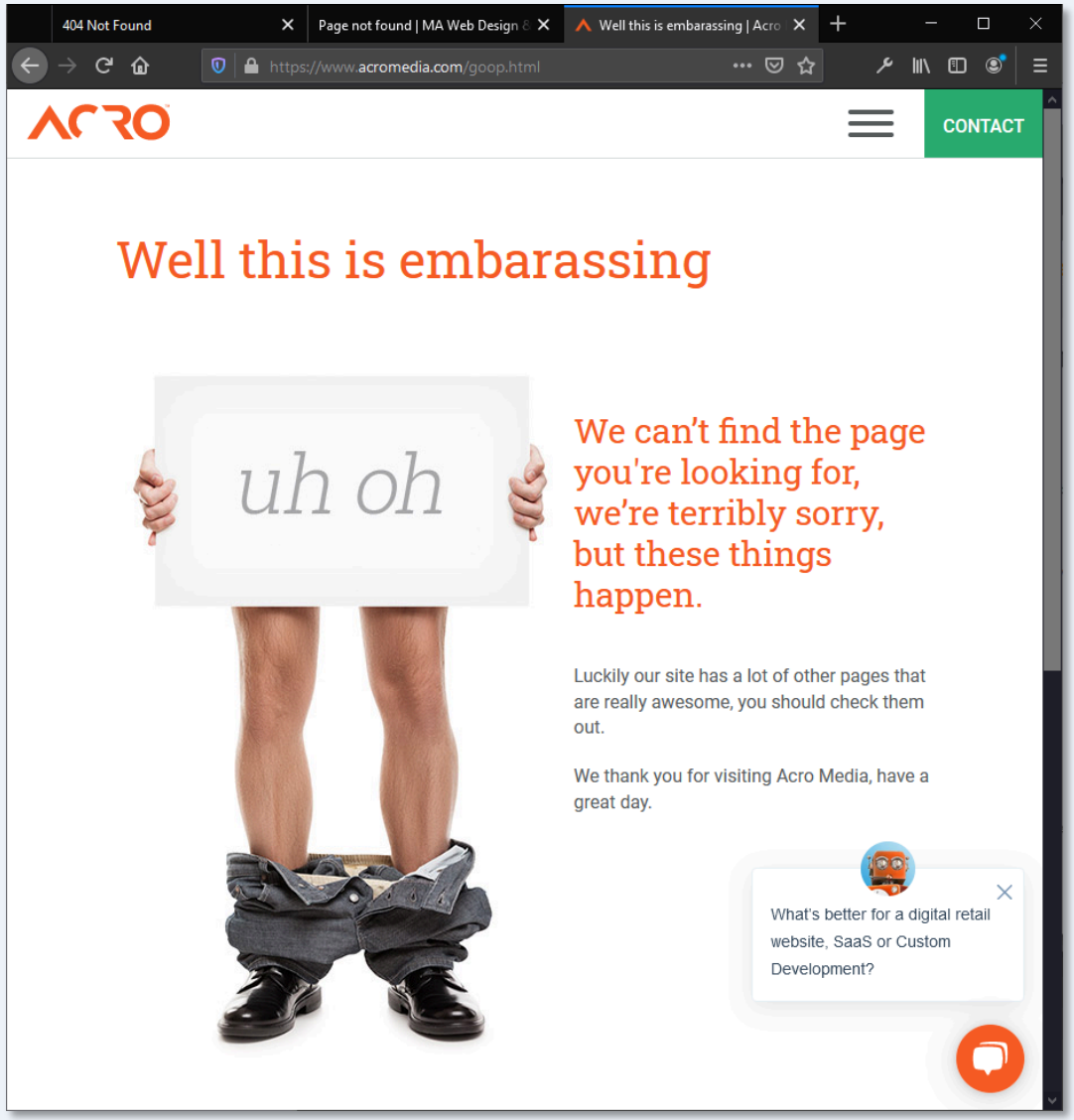


Take a look at [50 Creative and Inspiring 404 Pages] for inspiration

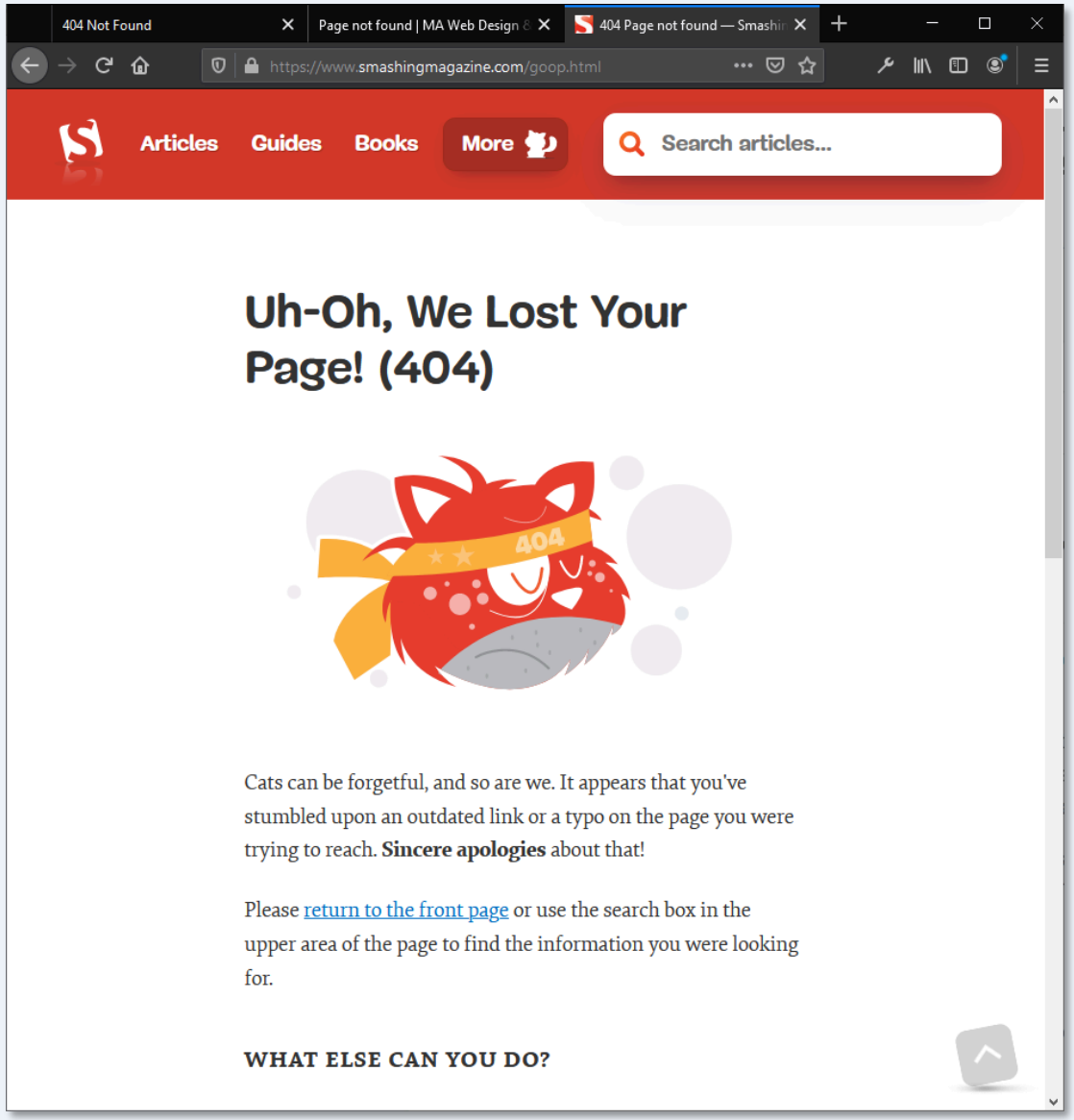
clearleft.com



acromediainc.com



smashingmagazine.com



Content Management

.htaccess: PASSWORD PROTECTION

Go to: springishblue.uk/secret

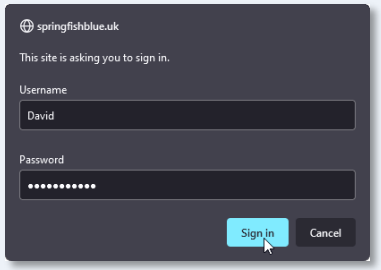
Password protection

- Password protection requires a .htaccess file in the folder to be protected and a .htpasswd file located anywhere on the domain (ideally in a secure location).
- In many cases, the .htpasswd file is located in the same folder as .htaccess but if you have access to folders above the web root, it should be placed there as it is more secure.

How it works...

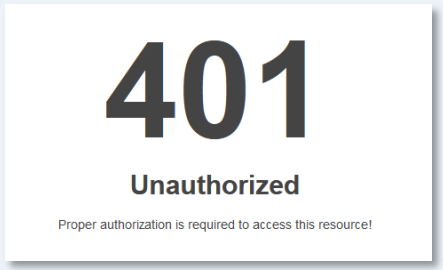
1. User requests access to folder by entering address in browser.

2. Server checks if folder contains .htaccess. If authentication is required...

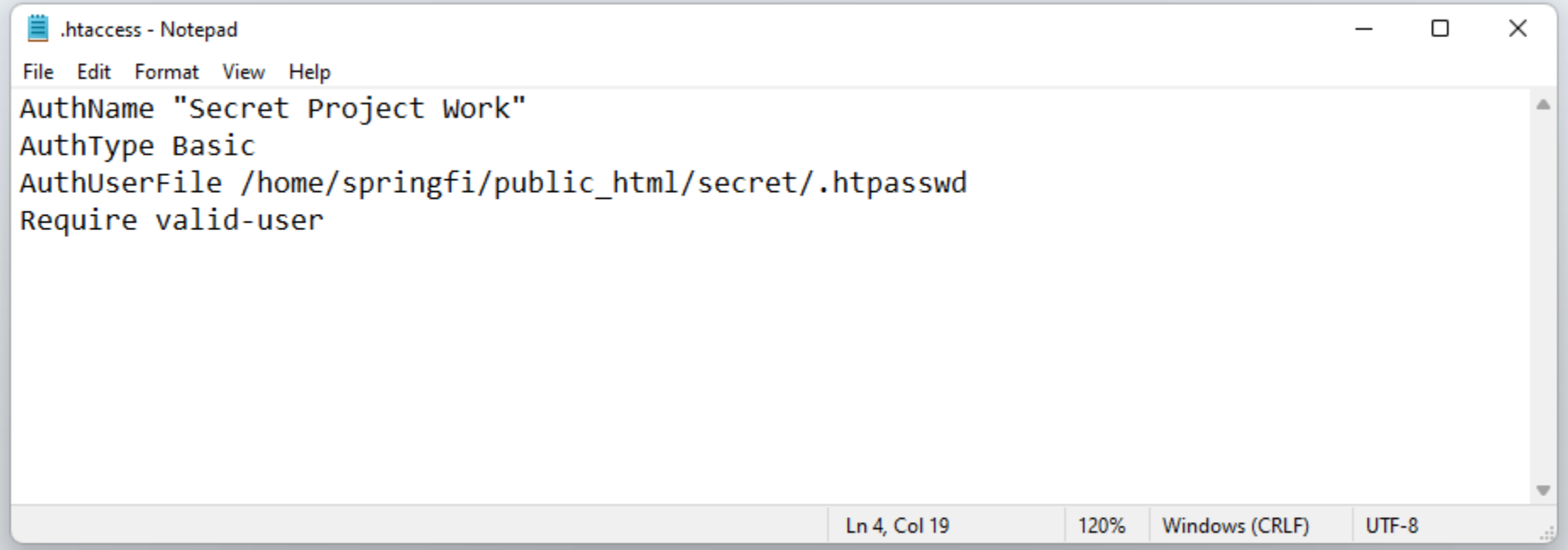


...user is asked to enter User Name and Password.

3. Server checks details against .htpasswd file. If correct, access is granted, if incorrect a 401 error is issued and error page displayed.



Password protection .htaccess

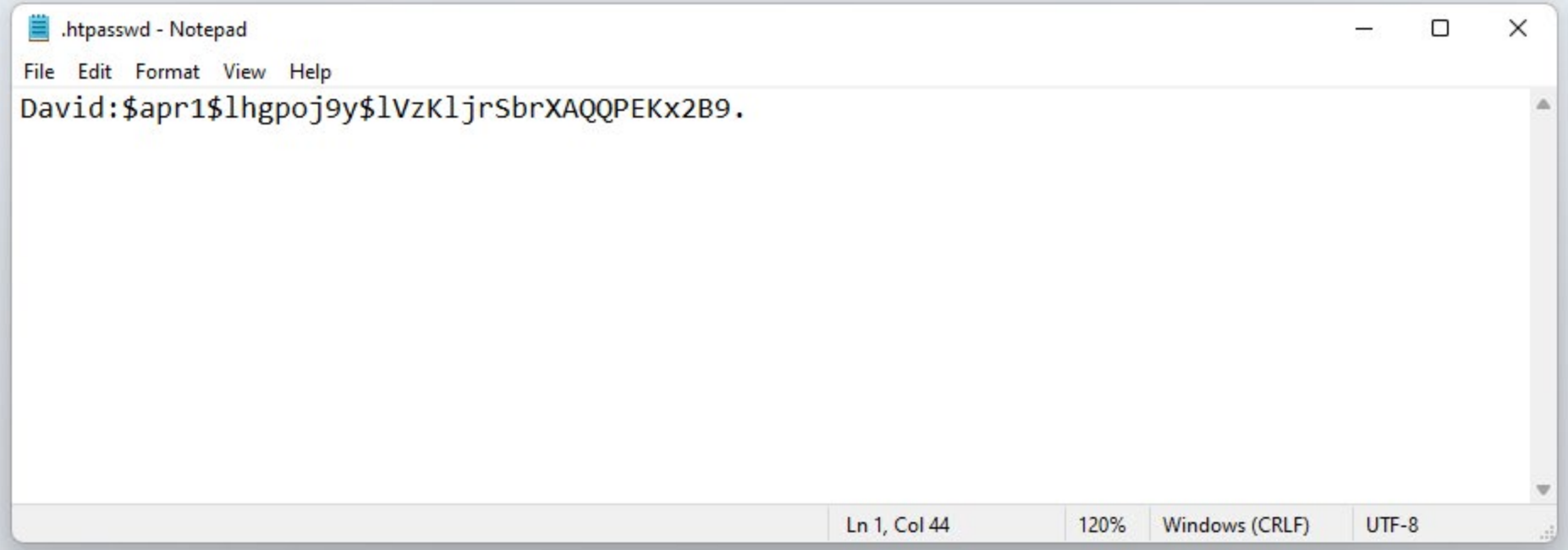


- *AuthName* = will display on some authentication dialogue boxes.
- *AuthType* = method used, *Basic* is the default.
- *AuthUserFile* = **server** path to the password file.
- *Require* = type of access (e.g. group access can be specified)

Take a look at Authentication, Authorization and Access Control for more information

Password protection .htpasswd

- The .htpasswd file contains a list of all the valid User Name/Password combinations, one on each line.
- The User Name is plain text but the Password is encrypted using the MD5 algorithm.



Wikipedia: [MD5]

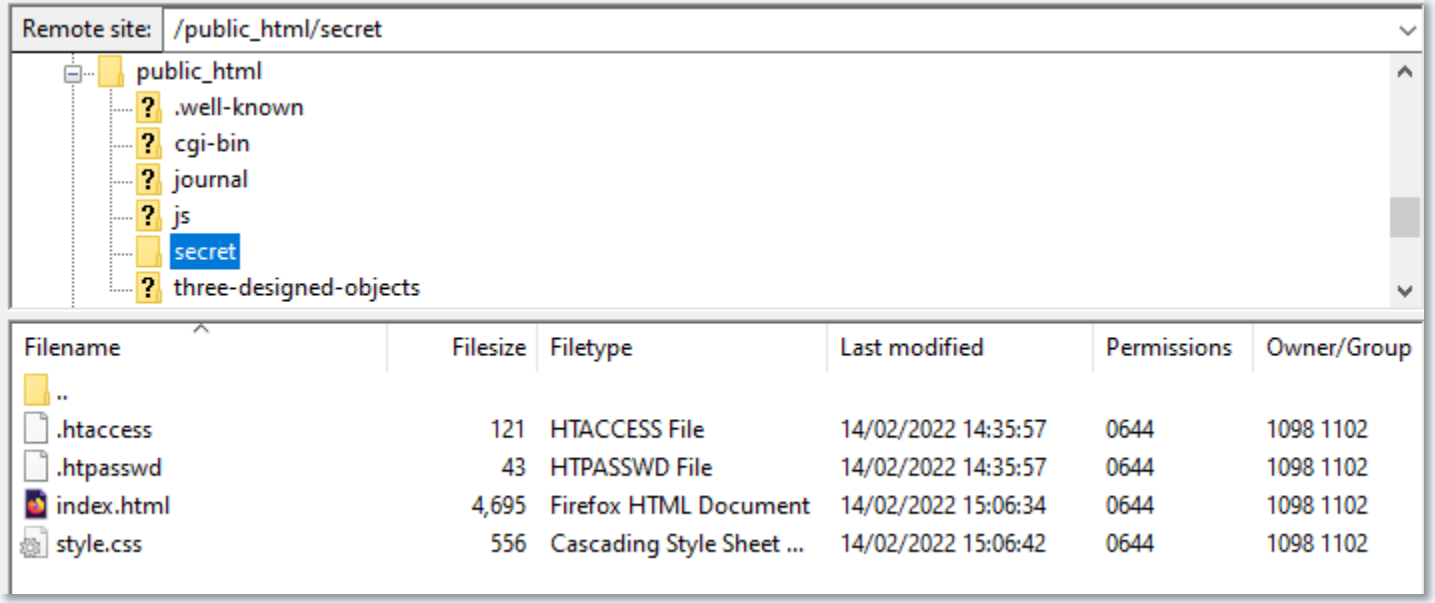
How to make .htpasswd

- There are plenty of free online tools that will automatically create .htpasswd files for you.
- Use Notepad to save your .htpasswd file and then upload to your site using FTP.
- Once both .htaccess and .htpasswd are in place, the folder is protected and accessible only by entering the correct authentication details.

Example .htpasswd generator

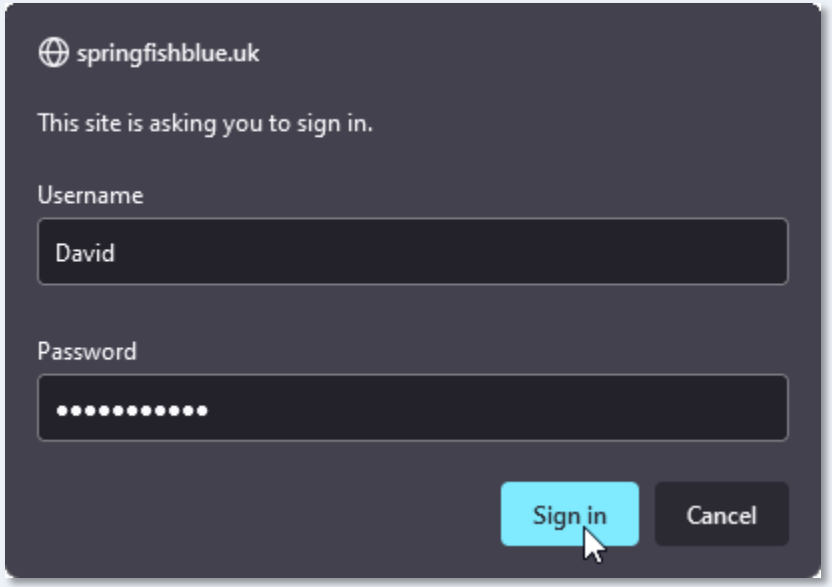
Uploading the files

- In the example below, we have a password protected folder called “secret”. That folder contains the .htaccess and .htpasswd files in addition to any content that needs to be protected.

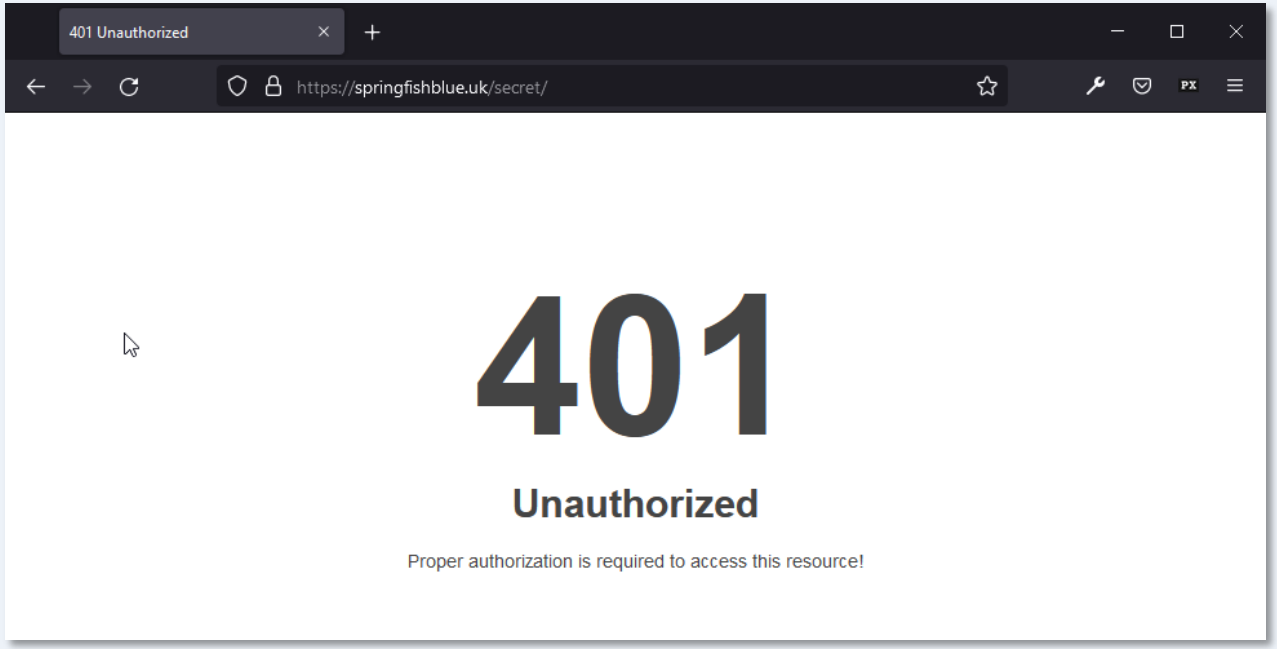


Authentication

- The authentication dialogue box is displayed when a user navigates to the folder.
- The dialogue box varies depending on browser. Firefox is shown below:

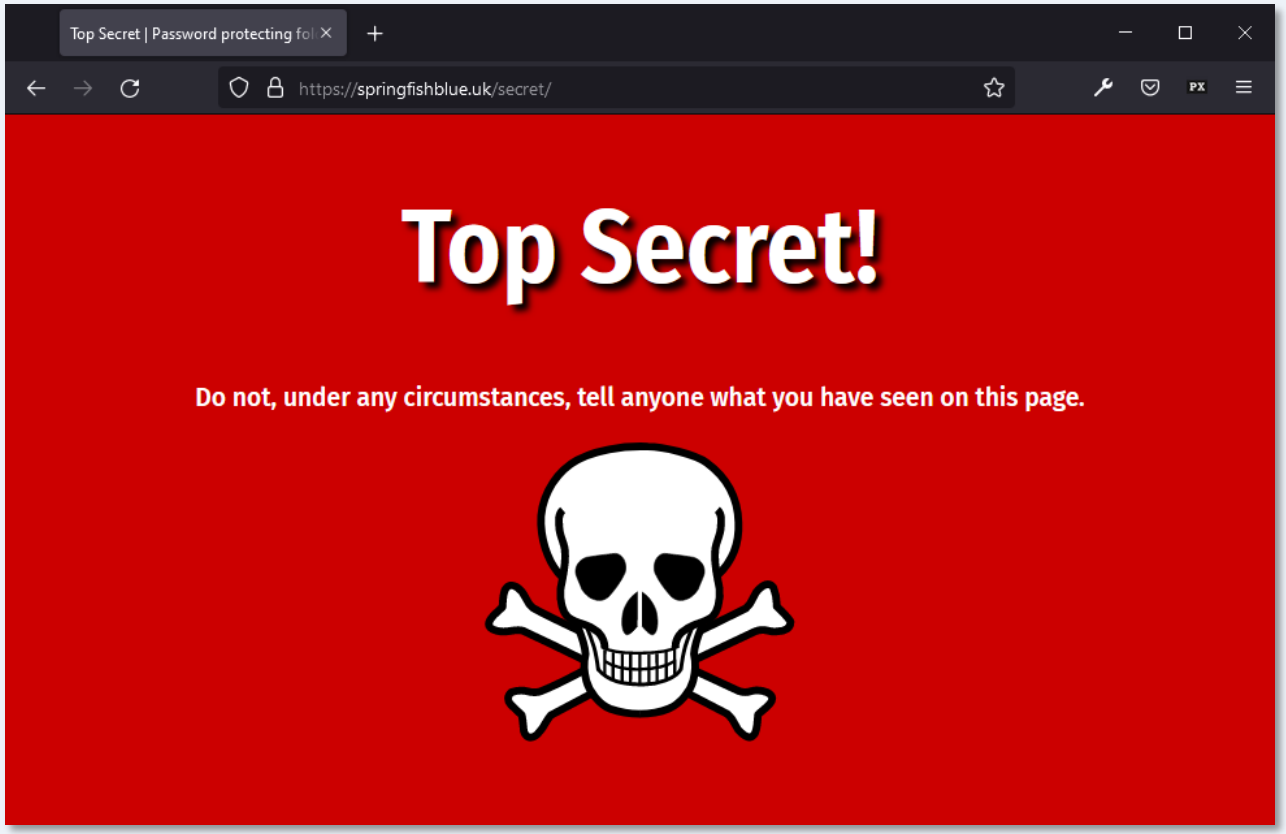


If authentication is unsuccessful



- If the authentication is unsuccessful (Username or Password are incorrect), a 401 error is issued.
- If you wanted, you could make a custom error page for 401 errors.

If authentication is successful

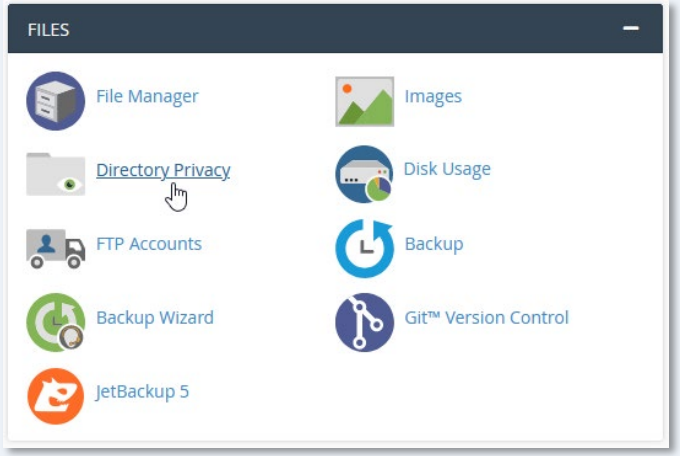


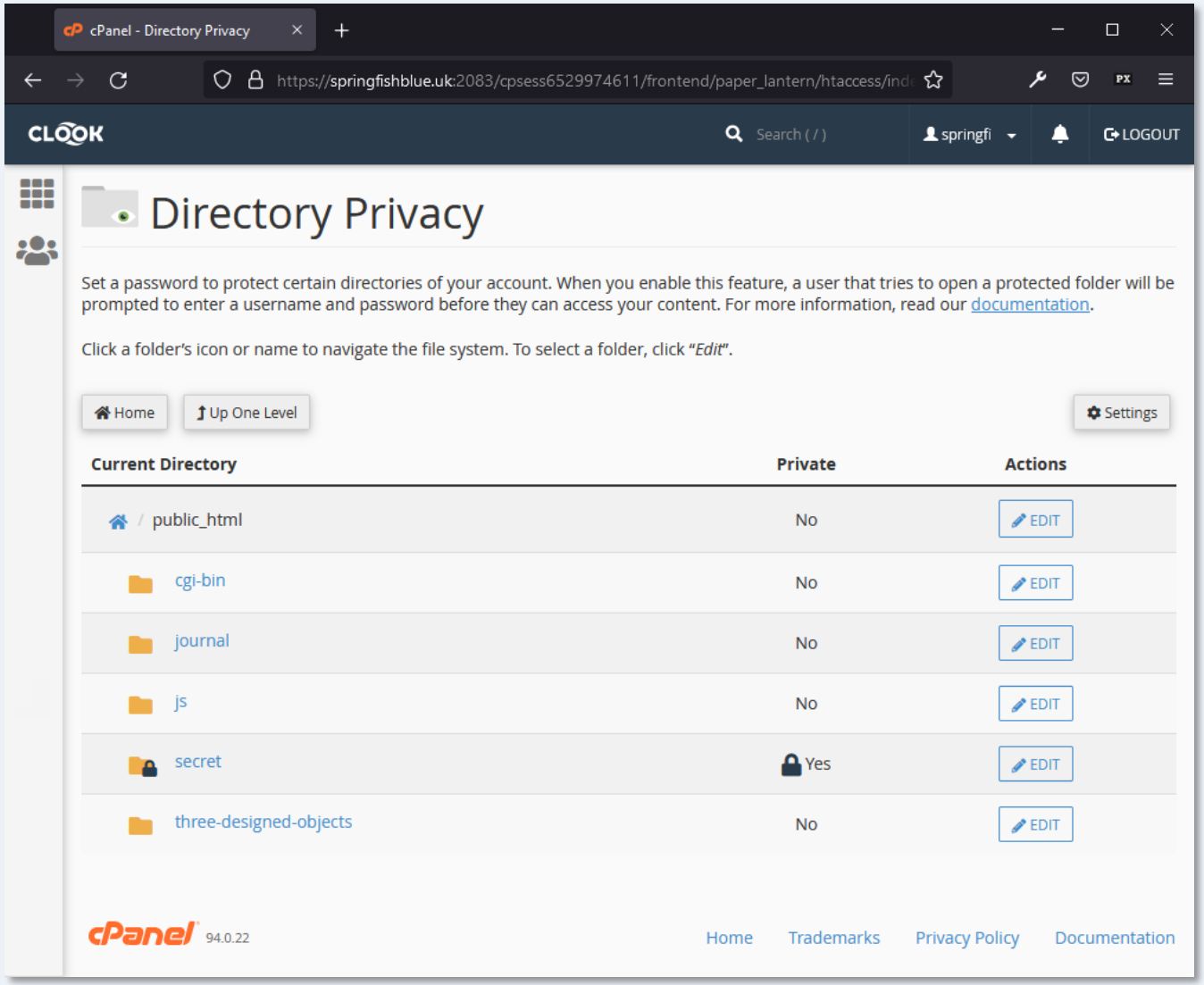
If the authentication is successful, the user is given access to the folder for the duration of the current session.

Password = secret-project-work

Keeping track in cPanel

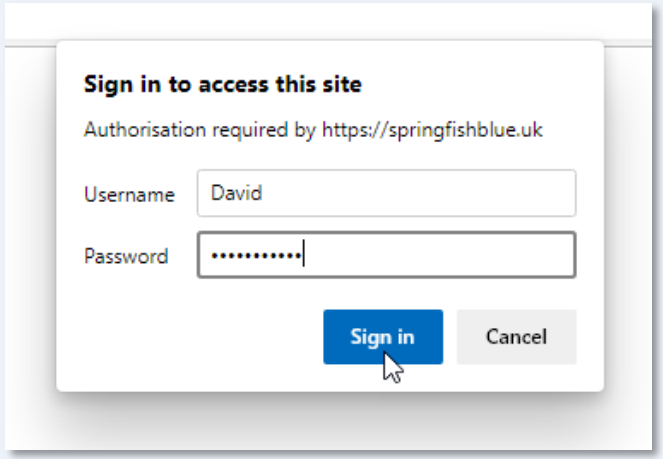
You can use the Directory Privacy page in cPanel to check which folders are protected. You can even use cPanel to password protect a folder if you'd prefer to do it that way…



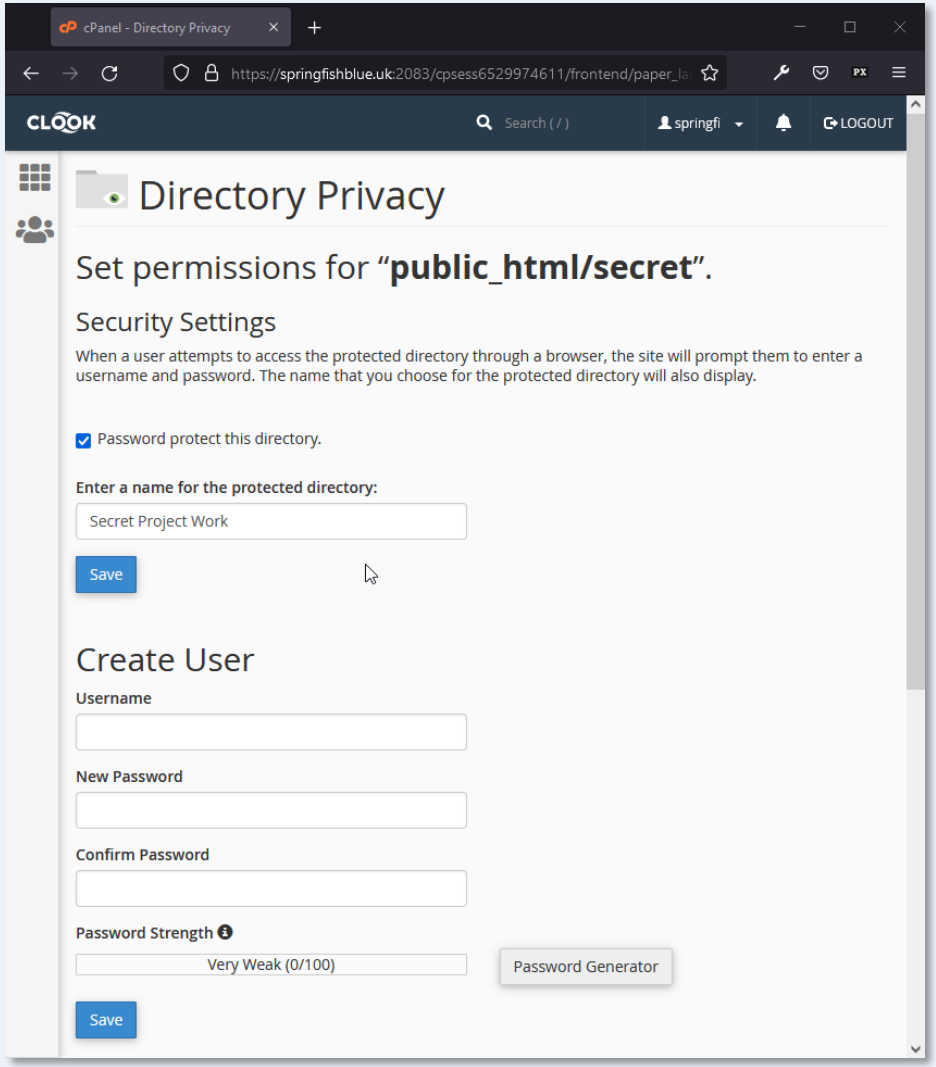


Hosting control panel

Setting up password protection manually can be a bit of a faff, so most hosting control panels have a tool you can use to do it more conveniently. cPanel refers to this as “Directory Privacy”.



The Microsoft Edge authorisation dialogue box



Content Management

.htaccess: REDIRECTION

Websites change

- Websites change: FACT
- In some cases you may want to rename a file or even rename your folders for SEO or for consistency as a site expands.
- So what happens when that popular page has to move or is renamed?
- All the inbound links will be broken, including those from search engines – disaster!

Inbound links

- So, you need to make some major changes to your site...
- ...how can this be done without breaking all the inbound links?
- You can use a 301 redirect to tell search engines where the content has moved to.
- Furthermore, a 301 redirect tells search engines that this is a *permanent* move, so they can update their index accordingly.

The 301 Redirect

- You can use a 301 “permanent” redirect in .htaccess.
- This does 2 things:
  - it serves a new page when an old page is requested.
  - it tells search engines to change their index and replace the old page with the new one.

Directive syntax:
Redirect[space]301[space]old path from root[space]new absolute path
The example below redirects any request for the folder */acad* to the new folder */tutorials/autocad*, for example:
a request for */acad/index.html* is redirected to */tutorials/autocad/index.html*

```
Redirect 301 /acad/ https://www.cadtutor.net/tutorials/autocad/
```

Continue redirecting

- Although search engines will learn the new location of content very quickly via your 301 redirect, inbound links are not usually updated in any systematic way, so it’s a good idea to keep the redirect in place for as many years as you think appropriate.
- Most webmasters want their content to be correct and a quick email asking them to update their link usually works.

Temporary moves

- It’s less common that you may need to move content temporarily...
- ...but if you do, there’s a way to do that too.
- Simply use a 302 redirect directive.
- This redirects user requests in the same way as a 301 but it tells search engines not to update their index.

```
Redirect 302 /existing/ https://www.temporary.co.uk/mystuff/
```

Content Management

.htaccess: REWRITING URLS

Rewriting URLs

- .htaccess allows you to rewrite any URL and change its form using a Rewrite Engine module in the Apache server, called *mod_rewrite*.
- Common uses:
  - to change https://www.mydomain.com to https://mydomain.com or vice versa.
  - to change mydomain.co.uk to mydomain.com
  - to change difficult URLs (generated by blogs etc.) to search engine friendly ones.

Wikipedia: Rewrite engine

Canonicalization

- Canonicalization is an SEO issue.
- Search engines may consider *https://www.mysite.com* and *https://mysite.com* to be different websites when, in fact, they are the same.
- The following directive forces all URLs to be rewritten with the “www” even if the request was made without it.

```
RewriteEngine On
RewriteCond %{HTTP_HOST} ^mysite.com$ [NC]
RewriteRule ^(.*)$ https://www.mysite.com/$1 [R=301,L]
```

Wikipedia: Canonicalization

Matt Cutts: SEO Advice: url canonicalization

Regular Expressions

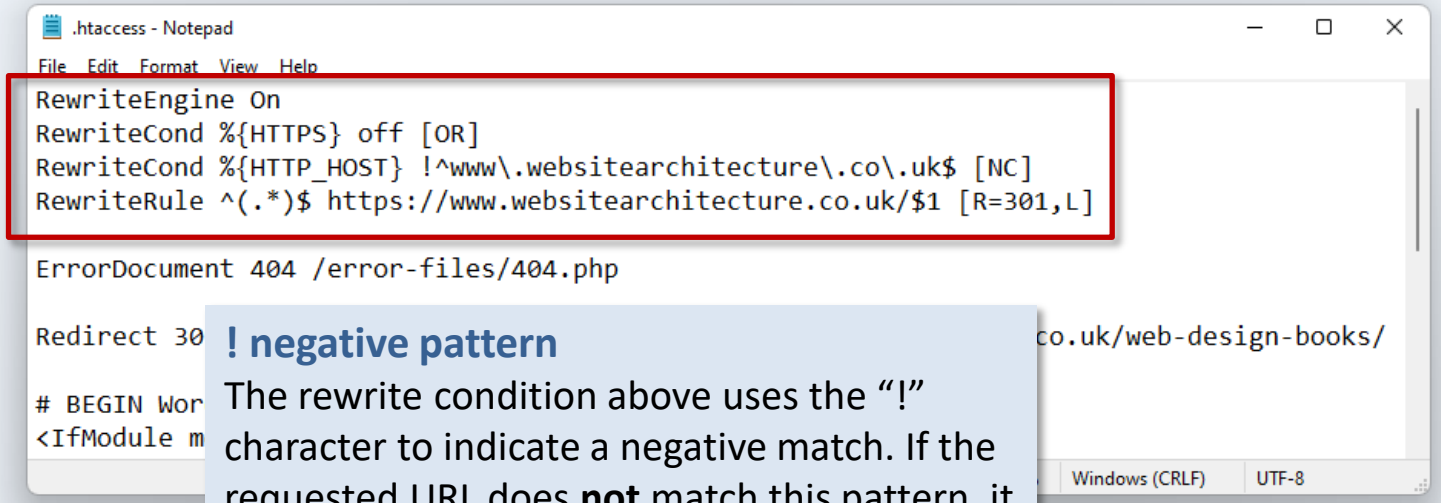
```
RewriteEngine On
RewriteCond %{HTTP_HOST} ^mysite.com$ [NC]
RewriteRule ^(.*)$ https://www.mysite.com/$1 [R=301,L]
```

- The directive strings for RewriteCond and RewriteRule look a bit odd.
- They use *regular expressions* (regex) to mach URL patterns.
- There's no need to craft your own regex, just use those that others have designed and substitute your own domain details.

Wikipedia: Regular expression

Normalising TLDs

- If you have a number of Top Level Domains (e.g. .com, .net, .co.uk) for the same name, mod_rewrite can be used to change them all to one preferred TLD.



! negative pattern

The rewrite condition above uses the "!" character to indicate a negative match. If the requested URL does **not** match this pattern, it will be rewritten so that it matches the pattern defined in the rewrite rule.

On the left is the .htaccess file used at the websitearchitecture website. The directive changes all TLD and hostname variations, with or without the "www" to the preferred URL. For example,
http://websitearctitecture.net
will be rewritten as:
https://www.websitearchitecture.co.uk
and that's what will appear in the address bar.

Tidy URL parameters

- URLs with parameters look untidy and may look suspicious to users who don’t understand how they work. They may also be bad for SEO.
- The RewriteEngine can be used to tidy such URLs.
- This technique is often used by content management systems.

```
RewriteEngine On
RewriteRule ^([0-9]+)\/?$ index.php?id=$1 [NC]
```

http://interaction.gallery/dream/index.php?id=25

becomes

http://interaction.gallery/dream/25

Content Management

.htaccess: PREVENT HOTLINKING

Stop Hotlinking!

- mod_rewrite can also be used to prevent people hotlinking (or inline linking) to your content and stealing your bandwidth.
- The directives below (added to .htaccess) will cause a “failed request” when .GIF, .JPG, .JS or .CSS files are requested from outside the server.

```
RewriteEngine on
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER}
!^http://(www\.)?mydomain.com/.*$ [NC]
RewriteRule \.(gif|jpg|js|css)$ - [F]
```

Wikipedia: Inline linking

Serving Alternate Content

- mod_rewrite can even be used to serve alternate content in response to a hot linking request.
- The directives below serve an image called *angryman.gif* every time a .GIF or .JPG file is requested from outside the server.

```
RewriteEngine on
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER}
!^https://(www\.)?mydomain.com/.*$ [NC]
RewriteRule \.(gif|jpg)$
https://www.mydomain.com/angryman.gif [R,L]
```

Hotlink prevention with cPanel

The Security section of cPanel includes a range of tools that you can use to protect your site and your content. Hotlink protection can be configured using cPanel, you don’t need to manually edit .htaccess.

Content Management

.htaccess: DENY ACCESS

Deny access by IP address

```
order allow,deny
deny from 123.16.14.245
deny from 41.251.66.32
deny from 105.238.0.
allow from all
```

deny from…
You can deny access from any specific IP address by adding a “deny from” directive and adding the explicit IP address, e.g. 123.16.14.245. But you can also deny access from an IP range by omitting one or more sets of digits. So, 105.238.0. means all IP addresses between 105.238.0.0 and 105.238.0.225.

There may be times when you want to prevent access to your website from certain IP addresses. Say you suspect a hacking attempt and you have the user IP address from your server logs or you just want to stop a bandwidth-hogging bot.

Simply, add any IP addresses you want to deny access to in your .htaccess file using the syntax shown above.

This can also be used to deny access to specific folders – just add a .htaccess file to that folder with the appropriate deny/allow directives.

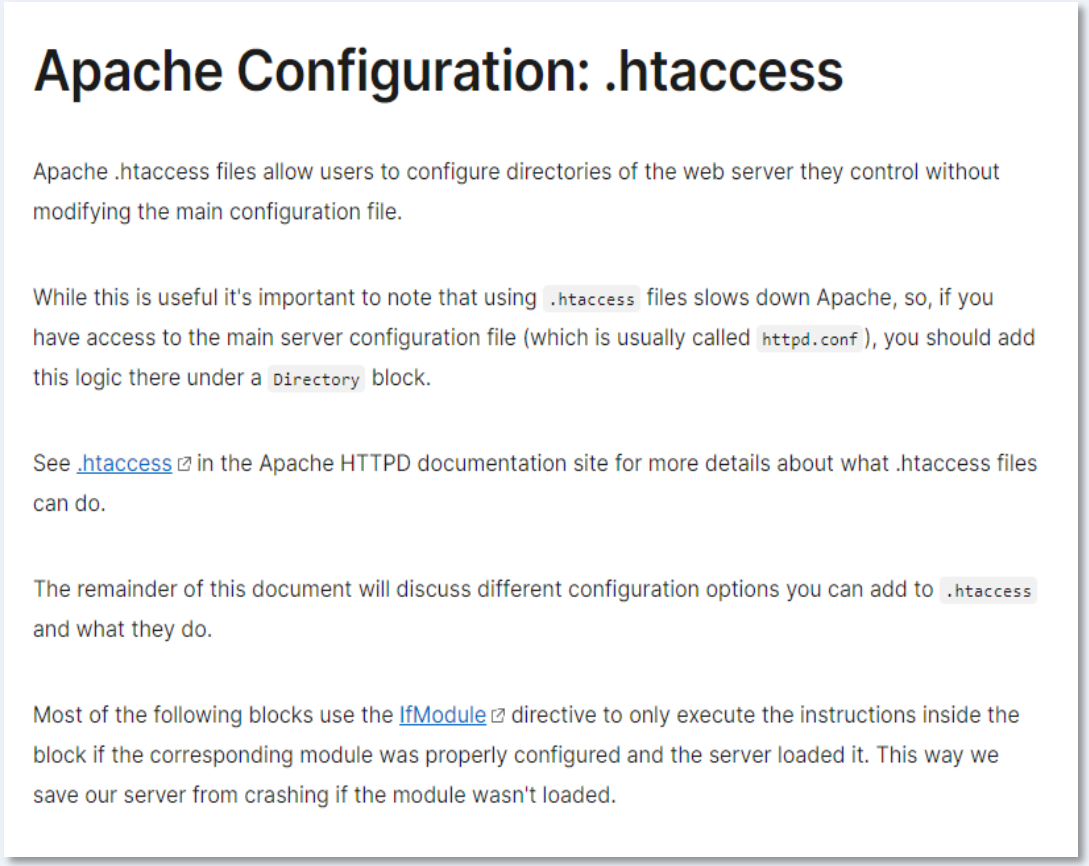
Host restriction from control panel

Just like many of the other .htaccess functions, denying access by IP address (or *host restriction*) can be implemented from your hosting control panel.

.htaccess is your friend

- There’s much more to .htaccess than we’ve covered here, but these are the most common use cases.
- You should at least be aware of the functions covered because you will need to use them from time-to-time and although some of the syntax looks like gobbledygook (particularly regex), .htaccess can be a very powerful friend.

Learning to use .htaccess



Apache Configuration: .htaccess

Apache .htaccess files allow users to configure directories of the web server they control without modifying the main configuration file.

While this is useful it's important to note that using `.htaccess` files slows down Apache, so, if you have access to the main server configuration file (which is usually called `httpd.conf`), you should add this logic there under a `Directory` block.

See .htaccess in the Apache HTTPD documentation site for more details about what .htaccess files can do.

The remainder of this document will discuss different configuration options you can add to `.htaccess` and what they do.

Most of the following blocks use the IfModule directive to only execute the instructions inside the block if the corresponding module was properly configured and the server loaded it. This way we save our server from crashing if the module wasn't loaded.

Apache Configuration: .htaccess at MDN Web Docs

.htaccess made easy

.htaccess made easy the book by Jeff Starr

Content Management

sitemap.xml

sitemap.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.google.com/schemas/sitemap/0.84">
  <url>
    <loc>http://www.websitearchitecture.co.uk/</loc>
    <changefreq>weekly</changefreq>
    <priority>0.5</priority>
  </url>
  <url>
    <loc>http://www.websitearchitecture.co.uk/programme-details</loc>
    <changefreq>weekly</changefreq>
    <priority>0.5</priority>
  </url>
  <url>
    <loc>http://www.websitearchitecture.co.uk/core-courses</loc>
    <changefreq>weekly</changefreq>
    <priority>0.5</priority>
  </url>
</urlset>
```

As its name suggests, sitemap.xml is an XML file that lists all the important content on your website. It tells Google and other search engine spiders which content you would like them to index. It also includes options that allow you to specify how often the content changes and its relative importance.

Element Definitions

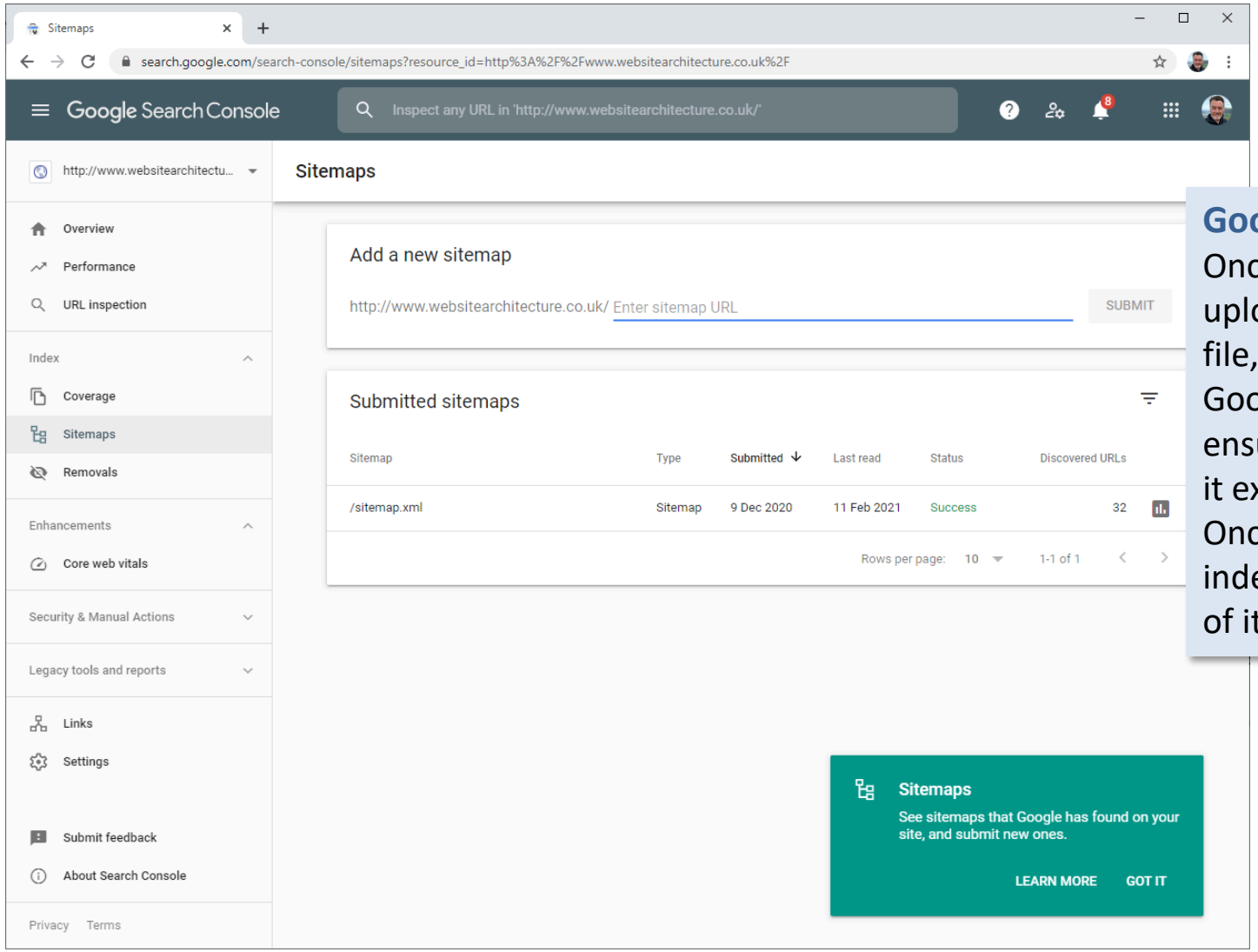
Element	Required?	Description
<urlset>	Yes	The document-level element for the Sitemap. The rest of the document after the '<?xml version>' element must be contained in this.
<url>	Yes	Parent element for each entry. The remaining elements are children of this.
<loc>	Yes	Provides the full URL of the page, including the protocol (e.g. http, https) and a trailing slash, if required by the site's hosting server. This value must be less than 2,048 characters.
<lastmod>	No	The date that the file was last modified, in ISO 8601 format. This can display the full date and time or, if desired, may simply be the date in the format YYYY-MM-DD.
<changefreq>	No	How frequently the page may change: • always • hourly • daily • weekly • monthly • yearly • never 'Always' is used to denote documents that change each time that they are accessed. 'Never' is used to denote archived URLs (i.e. files that will not be changed again). This is used only as a guide for crawlers, and is not used to determine how frequently pages are indexed.
<priority>	No	The priority of that URL relative to other URLs on the site. This allows webmasters to suggest to crawlers which pages are considered more important. The valid range is from 0.0 to 1.0, with 1.0 being the most important. The default value is 0.5. Rating all pages on a site with a high priority does not affect search listings, as it is only used to suggest to the crawlers how important pages in the site are to one another.

The sitemap protocol is recognised by Google, Yahoo! And Microsoft.

Wikipedia: Sitemaps

Building sitemaps

- You can easily build your own sitemaps if you have a simple site with a few pages. All the information you need is available at sitemaps.org.
- If you have a site with many 100s or 1000s of pages, what should you do then?
- Fortunately, there are a few free services that will crawl your site and build sitemap.xml for you. For example: XML-Sitemaps.com.
- However, always check that you get what you want. These services do not discriminate, and you may want to edit the result before using it.
- Google Search Console recommends you use *sitemap.xml* for all your sites – that's a pretty good hint that you should have one!



Google Search Console

Once you have created and uploaded your sitemap.xml file, you should submit it to Google Search Console. This ensures that Google knows it exists and how to find it. Once submitted and indexed, you can keep track of its use by Google.

https://search.google.com/search-console

Content Management

robots.txt

robots.txt

```
User-agent: *
Disallow: /error/
Disallow: /includes/
Disallow: /forum/clientscript/
Disallow: /forum/cpstyles/
Disallow: /forum/customavatars/
Disallow: /forum/customgroupicons/
Disallow: /forum/customprofilepics/
Disallow: /forum/images/
Disallow: /forum/includes/
Disallow: /forum/install/
Disallow: /forum/signaturepics/

Sitemap: http://www.websitearchitecture.co.uk/sitemap.xml
```

The purpose of robots.txt is to tell crawlers/spiders where they should not go. In other words, it lists any content that you **do not** want indexed. By default, spiders will index any content they find.
In the example above, robots.txt is also used to alert spiders to the fact that sitemap.xml is available. Essentially, that file tells spiders what you **do** want them to index.

Building robots.txt

- As its name suggests, robots.txt is just a simple text file and you can easily write your own following the protocol at robotstxt.org.
- All spiders request robots.txt when they first access a website. If the file is not found, a 404 error is issued, and the spider continues with crawling your site.
- Even if you have no content to hide, having a robots.txt file avoids the 404 error and the serving of your custom error page, if you have one.

Wikipedia: Robots exclusion standard

Empty robots.txt file

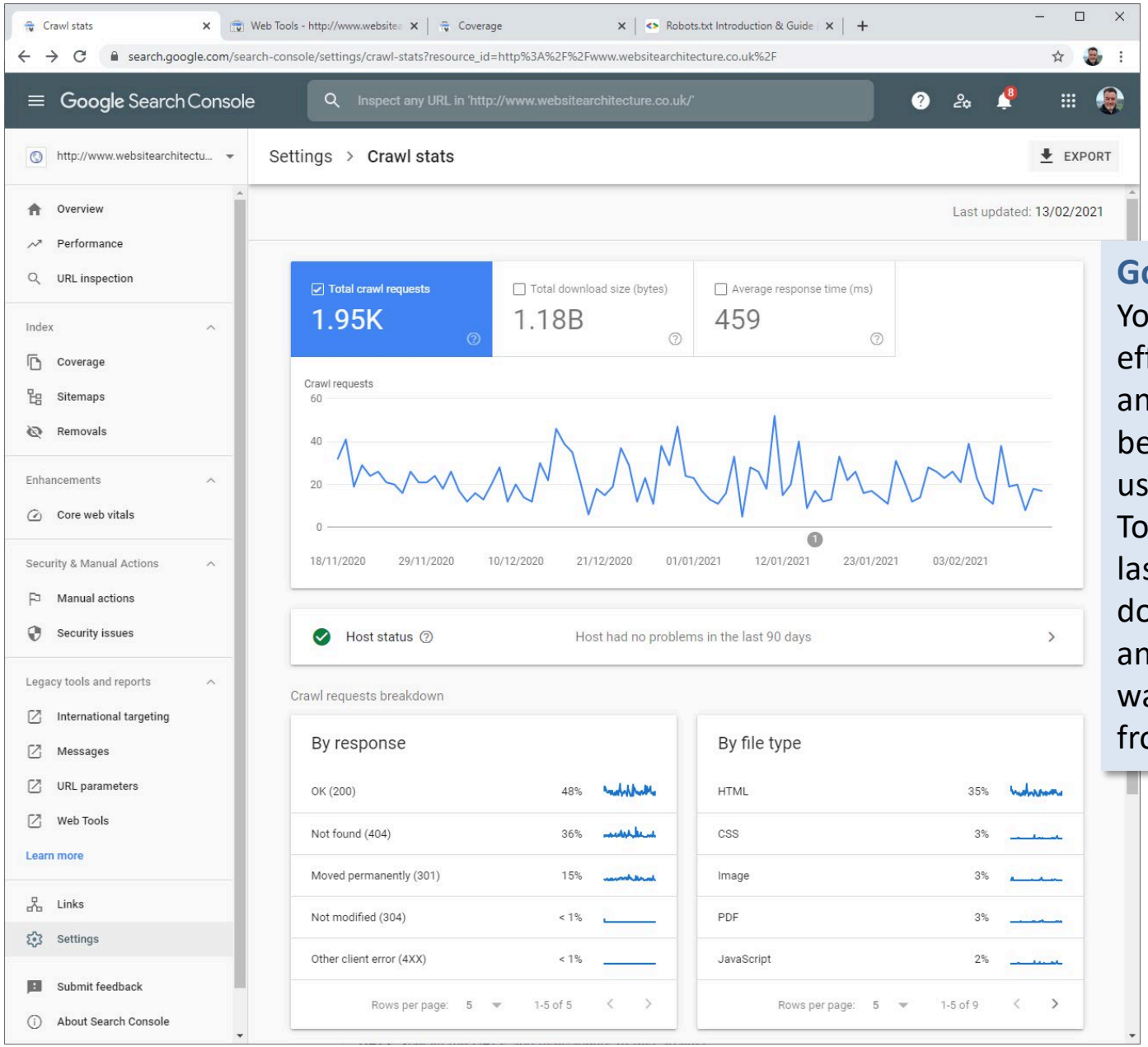
```
==============
User-agent: *
Disallow:


==============
```

It’s probably a good idea to include a robots.txt file in your web root in order to avoid 404 errors. Something like the text above is all you need (note the 2 blank lines after “Disallow:”). Don’t forget it to add to your sitemap when you have one in place.

Note: this is not a substitute for password protection because not all spiders play by the rules!

Webmaster Central: Do I need a robots.txt file?



Google Testing Tool

You can check the effectiveness of robots.txt and to see whether it is being correctly interpreted using the Google Testing Tool. You can also see the last time robots.txt was downloaded (by Google) and whether the request was completed successfully from the Search Console.

https://developers.google.com/search/docs/advanced/robots/intro

Content Management

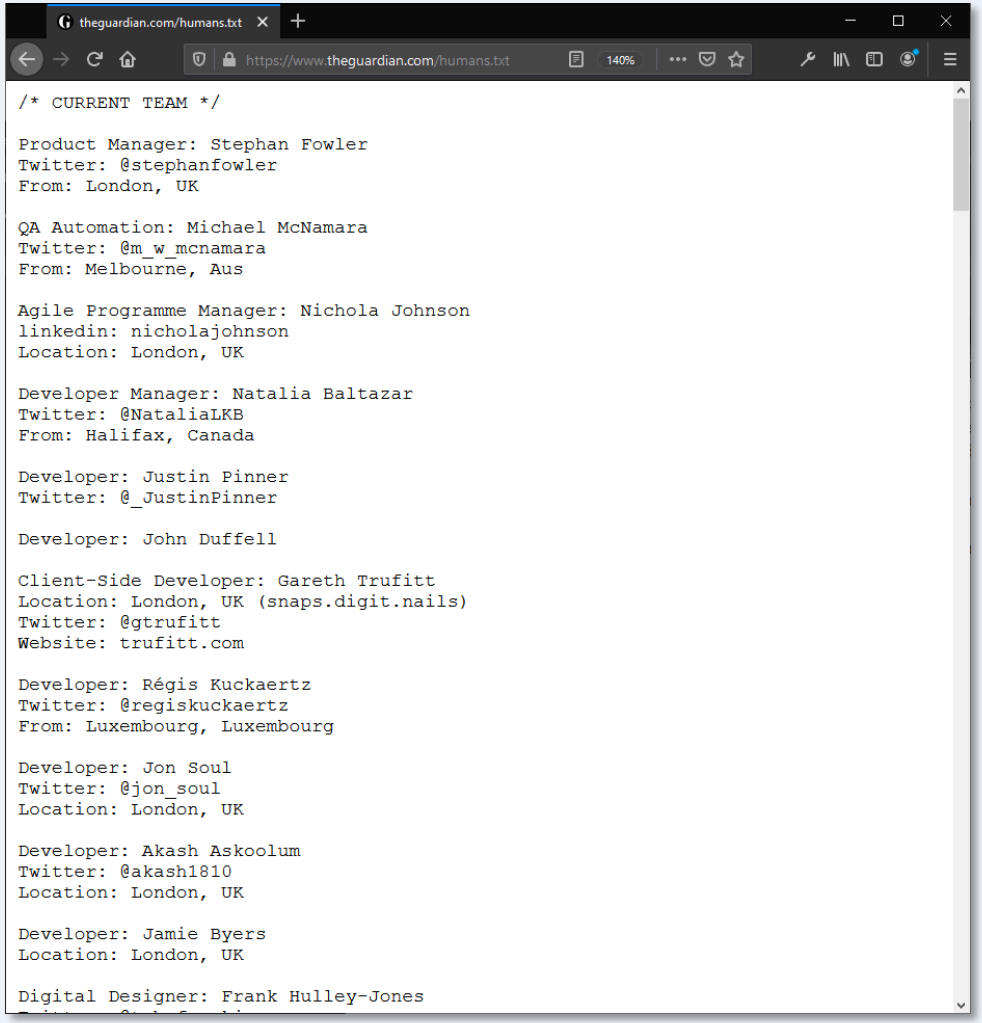
humans.txt

humans.txt

Optionally, you may add a humans.txt file to the root folder of your website. This file is for humans to read (hence the name) and should contain information about the authors of the website and details of the technologies and methods used in its construction as well as any other relevant information.

Unlike robots.txt, this file has no practical function and is not commonly used but it does demonstrate good attention to detail and it’s a nice way to give credit to those involved in a design project.

humanstxt.org



theguardian.com/humans.txt is a good example of a typical humans.txt file it contains brief details of those behind the website.

Redirect 301 start end