COESIONE ITALIA 21-27 PIEMONTE

Cofinanziato dall'Unione europea

REGIONE PIEMONTE

Web Developer

HTML, CSS e Strumenti di Digital Marketing (SEO, SEM, SEA)

Docente: Shadi Lahham

Intervento realizzato da

ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

.htaccess

Technical seo

Shadi Lahham - Web development

.htaccess

.htaccess

a configuration tool used by Apache web servers to manage website behavior and enhance SEO, allowing webmasters to implement URL redirects, set custom error pages, and create search-engine-friendly URLs through rewriting, thereby improving site structure and visibility in search results, making .htaccess a valuable asset for server management and SEO strategies

Basics uses of .htaccess

- rewrite URLs
- redirect URLs
- define custom error pages

Rewrite

```
# this code conditionally applies the rewrite rules based on whether the mod_rewrite module is loaded
# to work, the mod_rewrite module must be enabled on the Apache server
# this can usually be done from the Apache Configuration File (httpd.conf or equivalent)
# note that in all subsequent examples, the `IfModule` check will be omitted for clarity and brevity

<IfModule mod_rewrite.c>
    RewriteEngine On
    # Your rewrite rules here
</IfModule>
```

Rewrite

Purpose - Create SEO-friendly URLs by rewriting dynamic URLs into clean, descriptive URLs

```
# redirects URLs without a .php extension to their .php equivalent.
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^([^\.]+)$ $1.php [NC,L]
```

- **RewriteEngine On** directive activates URL rewriting
- **RewriteCond %{REQUEST_FILENAME} !-f** checks if the requested file doesn't exist
- If so, the **RewriteRule ^([^\.]+)$ $1.php [NC,L]** rewrites the URL by appending .php and redirects the request
- The **^([^\.]+)$** pattern matches URLs without a trailing dot, and the **[NC,L]** flags ensure case-insensitive matching and prevent further rewriting

This code enables clean URLs by hiding the underlying file structure

Rewrite

```
# Redirects URLs matching /product/ID to /product.php?id=ID
RewriteEngine On
RewriteRule ^product/([0-9]+)/?$ /product.php?id=$1 [L,QSA]
```

- **RewriteEngine On** directive activates URL rewriting
- **RewriteRule ^product/([0-9]+)/?$ /product.php?id=$1 [L,QSA]** rule matches URLs starting with **/product/** followed by a sequence of numbers and optionally a trailing slash
- It then rewrites the URL to **/product.php?id=ID**, where **ID** is the captured sequence of numbers
- The **[L,QSA]** flags ensure that this rule is the last to be applied and that any existing query string parameters are preserved

This code modifies URLs that represent product pages to makes it easier to pass product IDs to the server-side script

Redirect

Purpose - permanently redirect any request for resource to another URL

```
# Redirect old-page.html to new-page.html (within the same domain) using a 301 permanent redirect
Redirect 301 /old-page.html /new-page.html
```

using the Redirect directive with the 301 flag permanently redirects requests for /old-page.html to /new-page.html on the same domain, seamlessly moving users while informing search engines of the permanent change

```
# Redirect old-page.html to https://www.example.com/new-page.html (different domain)
Redirect 301 /old-page.html https://www.example.com/new-page.html
```

redirecting requests to a different domain is useful when a website moves and you want to ensure users land on the correct page, but search engines may take longer to recognize the change due to the external domain

Redirect

```
# another way to redirect old-page.html to new-page.html using rewrite with a 301 permanent redirect
RewriteEngine On
RewriteRule ^old-page\.html$ /new-page.html [L,R=301]
```

using the RewriteEngine and RewriteRule to permanently redirect requests
for /old-page.html to /new-page.html. The [L,R=301] flags make this rule
the last applied and indicate a permanent redirect

```
# but now we can add conditions such as redirecting old page to new page for specific IP addresses
RewriteEngine On
RewriteCond %{REMOTE_ADDR} ^192\.168\.1\.1$
RewriteRule ^old-page\.html$ /new-page.html [L,R=301]
```

a targeted way to redirect, allowing redirects for specific users or groups
based on IP address, where RewriteCond checks if the IP matches 192.168.1.1 and
RewriteRule permanently redirects requests only if the condition is met

Custom error pages

Purpose - provide informative and relevant error pages to improve UX and reduce bounce rates

```
# Redirect 404 and 500 errors to custom error pages
ErrorDocument 404 /404.html
ErrorDocument 500 /500.html
```

defines custom error pages for 404 (Not Found) and 500 (Internal Server Error) errors so that when visitors encounter these errors, Apache redirects them to the specified pages, offering more informative and user-friendly messages instead of the default ones

```
# Define custom error pages (alternative names)
ErrorDocument 404 /not-found.html
ErrorDocument 500 /server-error.html
```

these alternative error page names are more descriptive and can provide more meaningful information to visitors, depending on your preference and the specific messages to display

.htaccess for SEO

1.Managing index/noindex

managing index/noindex with .htaccess allows control over which pages search engines can crawl and index on a website

crucial for SEO as it guides search engines to the most important content while preventing duplicate or low-value pages, such as admin pages, login pages, temporary pages, or thank-you pages after a form submission, from appearing in search results

1.Managing index/noindex

```
# Noindex a specific file - can use only noindex, but usually don't want to follow page links either
<Files "private-page.html">
    Header set X-Robots-Tag "noindex, nofollow"
</Files>

# Noindex an entire directory
<Directory "/path/to/private-directory">
    Header set X-Robots-Tag "noindex, nofollow"
</Directory>

# Noindex files with specific extensions
<FilesMatch "\.(txt|pdf)$">
    Header set X-Robots-Tag "noindex, nofollow"
</FilesMatch>

# Noindex files matching a pattern
<FilesMatch "private-*">
    Header set X-Robots-Tag "noindex, nofollow"
</FilesMatch>
```

1.Managing index/noindex

Implementation
to manage index/noindex via .htaccess, X-Robots-Tag headers can provide directives to search engines about indexing behavior, where noindex prevents indexing of the page and **nofollow** prevents following **any** links on the page

Index/Noindex vs robots.txt
robots.txt controls crawling but not indexing, while noindex prevents indexing; a page blocked by robots.txt might still be indexed if linked from elsewhere, while noindex in .htaccess offers more granular control and can be applied to specific files or directories

if a search engine is prevented from crawling a site, it cannot access the pages or detect the noindex tag or header; thus, once a page is indexed and subsequently blocked by robots.txt, the search engine will not encounter the tag that instructs it to remove the page from the index

2.Managing HTTP/HTTPS

managing HTTP/HTTPS redirects in .htaccess is crucial for serving all traffic over a secure connection, essential for both security and SEO, as HTTPS encrypts data and is a ranking factor for Google, potentially improving search rankings

it prevents duplicate content issues between HTTP and HTTPS versions, builds trust, and aligns with search engine preferences, potentially enhancing visibility in search results

2.Managing HTTP/HTTPS

```
# Basic HTTP to HTTPS redirect
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]
```

%{HTTPS}
used to check if the current request is using HTTPS, with common values being "on" for HTTPS and "off" for HTTP, useful for redirecting HTTP traffic to HTTPS

%{HTTP_HOST}
represents the domain name used in the request, including subdomains if present, useful for managing WWW vs. non-WWW redirects or handling multiple domains on the same server

%{REQUEST_URI}
captures the path and query string of the requested URL, including everything after the domain name, useful for redirecting old URLs to new ones or handling specific paths differently

note: more examples in the additional .htaccess lesson

3.Managing www and non-www

managing www and non-www versions of a website is essential to consolidate domain authority and prevent duplicate content issues, as search engines treat these as separate URLs, which can dilute SEO efforts; ensuring all inbound links point to one version and maintaining consistent URLs helps with branding and user experience

```
**Which to Choose?**
from an SEO standpoint, there's no inherent advantage to either version so the choice is based on
branding and technical considerations

E-commerce site -  "www.shop.com" might be preferred for a global brand
Local business - "localbakery.com" might feel more personal without www
Tech startup - "app.io" might use non-www for a modern feel
```

3.Managing www and non-www

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

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

note: this is limited to http, which shouldn't be used anymore; so not very useful!

3.Managing www and non-www - HTTPS

```
# Redirect HTTP to HTTPS and non-WWW to WWW in a single rule
RewriteEngine On
RewriteCond %{HTTPS} off [OR]
RewriteCond %{HTTP_HOST} !^www\. [NC]
RewriteCond %{HTTP_HOST} ^(?:www\.)?(.+)$ [NC]
RewriteRule ^ https://www.%1%{REQUEST_URI} [L,NE,R=301]

# Redirect HTTP to HTTPS and WWW to non-WWW in a single rule
RewriteEngine On
RewriteCond %{HTTPS} off [OR]
RewriteCond %{HTTP_HOST} ^www\. [NC]
RewriteCond %{HTTP_HOST} ^(?:www\.)?(.+)$ [NC]
RewriteRule ^ https://%1%{REQUEST_URI} [L,NE,R=301]

# [NE] stands for or "No Encoding"
# [OR] flag means if either of the conditions is true
# (?:www\.)? is a non-capturing group that optionally matches www.
# The ? makes it optional, and ?: prevents it from being captured or stored
```

Tools

Online .htaccess testers

.htaccess files are crucial for configuring Apache web servers, but testing and debugging them can be challenging

online .htaccess testers are valuable tools that allow quick verification of whether .htaccess rules and directives work as intended

htaccess Tester | TechnicalSEO.com
htaccess tester

VScode extensions

useful extensions for working with Apache configuration files helping to write, edit, and validate Apache configuration files more efficiently

[Apache Conf]
Syntax highlighting for Apache configuration files that provides color coding for keywords, directives, values, and more

[Apache Conf Snippets]
snippets for Apache Conf .htaccess file to quickly insert commonly used Apache directives and their corresponding values, saving time and reducing errors

Try

write your own .htaccess with different rules and test them with both of the following testers

[htaccess Tester | TechnicalSEO.com]
[htaccess tester]

make sure you read the debug information and make sure you understand what happened in each step

References

Use .htaccess file for SEO

Best .htaccess Guide For SEO

The Ultimate Guide to htaccess Files for SEO

How to redirect HTTP to HTTPS and www to non-www