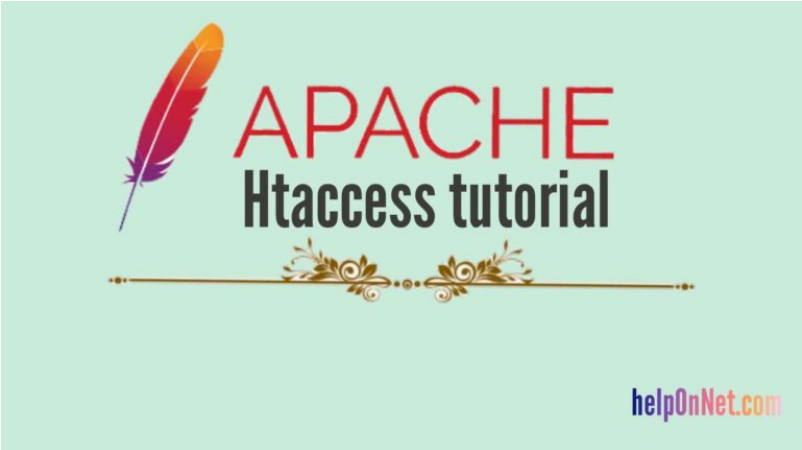
Apache .htaccess tutorial for beginners | Learn URL rewriting and basics of mod_rewrite – helponnet.com

APRIL 15, 2021 apache, Apache mod-rewrite and htaccess related Articles, htaccess

4.8 (107)

Last Updated on August 7, 2021 by Amit



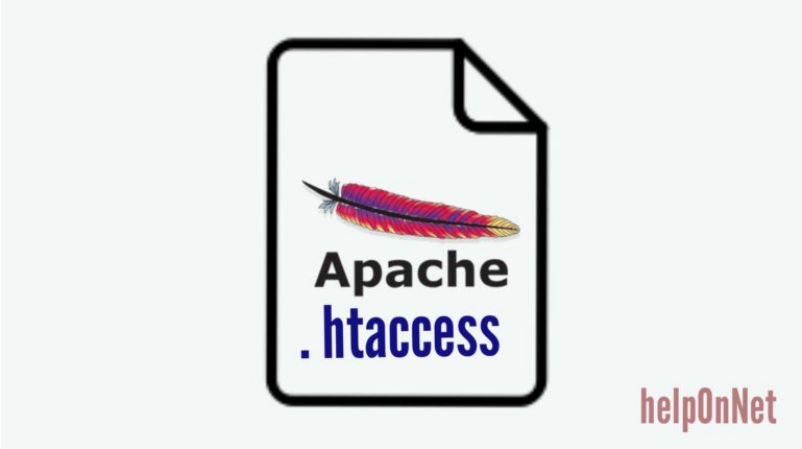
htaccess has always been a confusing topic for all Apache users especially for newbies who are learning to write RewriteRules to modify URLs via an htaccess file. The reason why some people find it confusing is because the Apache mod-rewrite documentation is hard to follow and there are no easy to follow **htaccess step by step tutorial** available on the internet.
I often see users struggling with their RewriteRule code on StackOverflow and I help them fix issues when I can.

In this 10 minute read article I am going to post an htaccess short step by step tutorial for people who wish to learn some basics of URL rewriting on Apache server.

If you follow this tutorial from top to the bottom then I am sure you will learn alot about URL rewriting and you have some basic idea to create your own RewriteRules.

Table of contents

What is an .htaccess file?



It's a directory level configuration file used by Apache web server to modify URLs , directory appearance and other server settings.

.htaccess is a dot prefix file.
The name starts with a single dot **.** and ends with **htaccess** .

You can change the name to make it look something like **htaccess.txt** from your server.config file.

The benefit of using an htaccess file is that it's easy to edit and available on all types of hosting services.

The downside of using an htaccess file is that it slows down your server performance but that's not going to be a big issue for you.
You can see the world's largest CMS WordPress also uses a htaccess file to shorten the blog URLs.

Apache mod-rewrite

Rewrite module aka mod-rewrite is a powerful Apache module that provides URLs rewriting functionalitis to Apache users.
The official documentation of mod_rewrite is available on http://httpd.apache.org/docs/2.4/mod/mod_rewrite.html.
Mod-rewrite provides URL rewriting directives such as

RewriteEngine
RewriteBase
RewriteRule
RewriteCond
All these directives are used in a htaccess file to manipulate incoming URL requests.

You will learn more about these directives later in this article but for now just focus on what the Rewrite module is and how to enable it.
Almost all live web hosting providers will provide you Apache server with pre-installed mod-rewrite. But in some rearest cases if it's not already installed then you can ask your service provider to enable it for you or you can also enable it yourself changing the line AllowOverride none to AllowOverride all this will then make it possible to use mod-rewrite in an htaccess file.

To verify whether your server supports mod-rewrite in htaccess , you can do the following :

- Create an .htaccess in your public_html folder.
- Put some random texts to your htaccess file like "fooooobarrr" .

- Now, visit any URL on your site.

If it generates a **500 internal server** error then the mod-rewrite is **enabled** on your server otherwise if you don't get any error then you will need to enable this .

Let's create our first Hello-world RewriteRule

If you are learning to write htaccess rules while following this article , then I suggest you to start learning with an empty htaccess file.
Create an htaccess file on your document root and put the following contents in that file :

```
RewriteEngine on
RewriteRule ^helloWorld$ /index.php
```

Save the htaccess file and type the following URL into your browser address bar :
`https://yourdomain.com/helloWorld`
You will see the real rewrite mazic happening now.
The rule will internally map your URL path **/helloWorld** to the destination file **/index.php**.
The redirection happens behind the screen. You will see contents of the index.php file on the /helloWorld path.

Congratulations , you have written your first **helloWorld** rule successfully.
Now let's understand how the code above works. Let's understand it line by line.

- **RewriteEngine on** : This directive turns on the engine for rewriting URLs.
  This directive should be placed once at the top of your htaccess.
- **RewriteRule ^helloWorld$ /index.php** : This is a RewriteRule directive provided by mod-rewrite. This line rewrites the request from **/helloWorld** to **/index.php** .
  You might have noticed one thing that our Rule's pattern

doesn't start with a leading slash / .
This is because RewriteRule uses a relative path in its pattern so **^/helloWorld$** will not work in htaccess but in a server.config file.

Okay, now I hope you have had some basic idea about how to create a simpe RewriteRule.
Now we will learn about other parts of a RewriteRule.

Mod-rewrite Flags

Flags are optional in RewriteRule but you will need to use them in most cases.
Flags provide additional instructions to a RewriteRule.
For example a RewriteRule with **[R]** flag tells the rule to make an external redirection of URLs.

See a list of most commonly used flags in mod-rewrite .

Flags are used inside sequere brackets [] .
You can use multiple flags with a RewriteRule seperating them by a comma char **,**.

Flags are optional in RewriteRule but you will most often need to use them in Rule to change the URL rewriting behaviour.

You will needv to use [NC] flag in the following Rule:

```
RewriteRule ^foobar$ /index.php
```

NC stands for NoCase, this flag is used to match both uppercase and lowercase chars in URI.
With NC, the rule above will match the following both URIs :

`/foobar`
and
`/fOObAr`
otherwise without this flag, by default RewriteRule only

matches the case sensitive URI string.

You can learn more about flags on this post .

Conditions in RewriteRule

You can use conditions with your RewriteRule to rewrite or redirect your site URLs based on some specific conditions.

RewriteCond directive is used for this purpose.
With RewriteCond you can rewrite your URL based on host , https , request URI headers.

Check this post to learn more about using RewriteCond directive.
You can use if/else logic for RewriteRule .

The following Rule uses RewriteCond to redirect the URL based on host header. The condition used in this rule can also be read as
“If **host** ==**example.com** then **Redirect** .

```
RewriteEngine on
RewriteCond %{HTTP_HOST} ^example.com$
RewriteRule ^foobar$ /index.php [R]
```

This rule will redirect /foobar to /index.php if the condition is met.

You can use multiple conditions with a single RewriteRule .

```
RewriteEngine on
RewriteCond %{HTTP_HOST} ^example.com$ [OR]
RewriteCond %{HTTP_HOST} ^www.example.com$
RewriteRule ^foobar$ /index.php [R]
```

The conditions above uses [OR] flag. This makes one of the conditions optional. So you can read it as :
“If host ==example.com OR host==www.example.com” then

Redirect.

To redirect http URLs to https , you need a condition logic. That checks if the incoming URL is using **http** scheme , then redirect the URL to use **https** :

```
RewriteEngine on

RewriteCond %{REQUEST_SCHEME} ^http$
RewriteRule (.*) https://example.com/$1 [R]
```

I hope now you have had some basic idea of how conditions work and how to use them with RewriteRule.

Regex in mod-rewrite

Mod-rewrite directives RewriteRule and RewriteCond use a regular expression based pattern that makes the match easier.
With a Regex based pattern a single RewriteRule can match multiple URIs.

If you want to learn Regex , you can follow this tutorial .

The following rule with regex pattern can either match **/helloWorld** or **/hello ladies** :

```
RewriteEngine on
RewriteRule ^(helloWorld|hello ladies)$ /index
```

The match is saved in **$n** variable . Since we have one capture group in the rule above , the match is saved in **$1** .

You can use the matched value in the destination string of RewriteRule.

```
RewriteEngine on
RewriteRule ^(helloWorld|hello ladies)$ /index
```

```

```

The rule above rewrites **/helloWorld** to **/index.php?q=helloWorld**

You can use Regular expressions with RewriteCond :

```
RewriteCond %{HTTP_HOST} ^(www\.)example.com$
```

The condition above checks both **www** and **non-www** domain .

Isn't it easier with Regex?

Comments in htaccess code

In any programming languages , comments are used to explain the block of code. Like in PHP we use # to write a single line comment and /*...*/ for multiple lines.
In htaccess , you can only comment out a single line using the hash character # .

Anything that goes after # is treated as a comment in htaccess.

```
#This is a comment.
#This is another comment.
```

To explain an htaccess code you can start your comment with a # character but keep in mind that the # is a single line comment. To add multiple line comments you will need to start each line with a hash character like the example shown below

```
#redirect http to https
#This rule also reditects non-www to www at th
```

```
RewriteCond %{HTTPS} off [OR]
RewriteCond %{HTTP_HOST} !^www
RewriteRule (.*) https://www.example.com/$1 [L
```

Error handling with htaccess

With an htaccess file you can handle server errors with just a few lines of code.
On Apache there are different types of server errors . Some most common errors are :

- 403 (Forbidden error)
- 404 (Not found error)
- 410 (Gone)
- 500 (Internal server error)
- 503 (Server unavailable) .

With an htaccess file you can easily handle these server errors using **ErrorDocument** directive which is part of Apache core module.
403 is the error status code which represents a “Forbidden error” (Access to the resource is denied) .
To redirect your site URLs to a different location when this error occurs, you can use the following code in htaccess :

```
ErrorDocument 403 /403.php
```

Here **403.php** is the file that is called when you request something that is protected on your server.

The code below handles 404 “not found” URLs.

```
ErrorDocument 404 /404.php
```

The same code format is used to handle other server errors except 500 (Internal server error) . You need to handle it from

your main server configuration file.

RewriteRule code examples

WWW to non-www

With the following rule you can redirect your **www** subdomain to root domain :

```
RewriteEngine on
RewriteCond %{HTTP_HOST} ^www\.example\.com$ [
RewriteRule (.*) https://example.com/$1 [R=301
```

Redirect non-www to www (naked domain to www subdomain)

If you want to redirect your root domain to www subdomain , here is the rule you can use to enforce **www** on your website :

```
RewriteEngine on
RewriteCond %{HTTP_HOST} ^example\.com$ [NC]
RewriteRule (.*) https://www.example.com/$1 [R
```

Change directory index file

By default , your server shows **index.html** , **index.php** or whatever index file exists to client when a directory is direct accessed , but you can change this behaviour by simply using the following line in htaccess , read more about DirectoryIndex

```
DirectoryIndex somefile.php
```

Access all files without adding an extension at the end

If you need visit your file URLs without adding extension at the end , for example : to visit /**file.php** as /**file** you can use the short code below in your htaccess :

```
Options +Multiviews
```

Enforce SSL

You can redirect your insecure traffic (http) to the secure version (https) using this rule :

```
RewriteEngine on
RewriteCond %{HTTPS} off [NC]
RewriteRule (.*) https://example.com/$1 [R=301
```

Add a trailing slash

To add a trailing slash to your URLs , you can use the following :

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule !/$ %{REQUEST_URI}/ [R,L]
```

Allow non-php extension

If you want to access your .PHP files without using extension at the end , you can use :

```
RewriteEngine on
RewriteCond %{REQUEST_FILENAME}.php -f
RwriteRule ^(.+?)/?$ /$1.php [L]
```

More rules : 5 Awasome htaccess rules you can just copy & paste .

I hope this tutorial was helpful. If there is anything that you couldn't understand or you need my help with htaccess , you can post a comment bellow.

Thank you so much reading this.

How useful was this post?

Click on a star to rate it!

Average rating 4.8 / 5. Vote count: 107

Tags: htaccess, Mod-rewrite

Top 5 reasons to avoid using Google blogger for blogging

How to point a custom domain to 000webhost using nameservers and cname

8 thoughts on "Apache .htaccess tutorial for beginners | Learn URL rewriting and basics of mod_rewrite – helponnet.com"

Peter SM says:

April 16, 2021 at 1:29 pm

Nice Article Amit. It helped me in writing a rule for my website.

Thank you so much.

Reply

amitoverflow1 says:

April 16, 2021 at 4:27 pm
Your welcome. Glad it helped you.

Reply


adhitiyan says:
July 8, 2021 at 10:53 pm
Hello Amit

That is a good article for me. Thank You Ver much…

I have some question regarding htaccess tutorial, how to redirect https://ipaddress to the https://domainname?

Reply


Amit says:
July 10, 2021 at 4:04 pm
Hi Adhitiyan! Thanks so much for posting your comment.
To redirect IP address to domain name , you could use :

```
RewriteEngine on
RewriteCond %{HTTP_HOST}
^IP_address_here$
RewriteRule ^ https://example.com%
{REQUEST_URI} [NE,L,R=301]
```

Just put your domain ip address in the condition pattern and Change example.com to your domain . Hope it helps.

Reply


Nick55 says:
July 19, 2021 at 10:46 am
Very informative and very helpful tutorial Amit. I always read your URL rewriting articles and it helps me . You also helped me on StackOverflow Thank you so much.

Reply

Amit says:

July 22, 2021 at 5:13 am

I am glad my tutorials are helpful to you. Thank you so much Nick!

Reply

Raghav says:

August 7, 2021 at 4:45 pm

This tutorial is really really helpful for me. Thanks Amit for such an easy htaccess tutorial.

Reply

Amit says:

August 7, 2021 at 4:50 pm

Thanks Raghav. I am glad to hear this from you. Keep learning.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

Captcha
*

Type the text displayed above:

POST COMMENT

Amazon ads

Archives

July 2021

June 2021

May 2021

April 2021

March 2021

January 2021

December 2020

April 2020

March 2020

February 2020

January 2020

December 2019

November 2019