Apache Rewrite with Htaccess 理解與技巧

awonwon
Oct 25, 2018 · 21 min read

在搜尋 Apache 針對網址改寫、避免訪問敏感檔案時，都會看到RewriteRule、RewriteCond 等的 Directive，但因為有時候不理解其中特性或Rule 跟 Cond 的差異等，加上若不熟悉正規表達式，完全只想逃避這些 Rule 們呢哭哭，因此本文將介紹：

※ 本文內容會盡量減少複雜的正規表達式，但若讀者對正規表達式一概不通的話，還是要先惡補一些些囉！
（或是你願意等待筆者撰寫下一篇關於正規表達式的文章 :P）

. . .

一、開啟 Rewrite 功能

輸入以下指令開啟 `Rewrite` 模組（筆者使用 Ubuntu 系統）

```
$ sudo a2enmod rewrite
=> LoadModule rewrite_module modules/mod_rewrite.so
```

確認是否已經開啟模組了

接著就可以開始在 Site Config 裡面撰寫所需的 RewriteRule 了。

在理解 Rewrite 的過程中，常常會出現與之搭配的 `.htaccess` 檔案，`.htaccess` 稱作「Hypertext Access」，以一個資料夾為單位改變 Apache 設定的檔案（Override Config），簡單來說就是可以根據每個資料夾 Override 原本 Site Config ，可以針對一個資料夾改寫網址，所以 RewriteRule 並非就只能寫在 `.htaccess` 當中哦！

但本篇文章還是會以使用 `.htaccess` 作為範例，若要打開 Override 的功能，只要修改 Site Config ，加入 `AllowOverride All` 就可以了（例如 `/etc/apache2/site-enabled/default000-conf` ）

```
<Directory "/var/www/html">
 AllowOverride All
</Directory>
```

All、FlieInfo 都可以讓 `.htaccess` 使用 Rewrite 功能，詳情可以參考官方設定值的意義。

二、Rewrite Rule

基本用法

就是改寫網址條件的規則，它的寫法結構如下：

```
RewriteRule [match_uri] [rewrite_uri] [flags]
```

- match_uri：符合 Pattern 的 URI
- rewrite_uri：將被改寫的 URI

這兩個也都可以使用正規表達式撰寫，一個 Rule 範例長這樣：

```
Rewrite ^match\.html$ rewrite.html [NC,L]
```

```
假設輸入網址：http://domain.com/a/b/c.html
uri = a/b/c.html

if (uri.match("match.html") {
 url = "rewrite.html"
}
```

RewriteRule 的 Flag

最後面的 `flags` 代表設定 Rule 的行為，可用逗號代表多個 Flag，中間不能有空格，介紹以下常用的：

```
[L]：Last，代表成功執行這個 Rule 後就會停止，不繼續往下執行。
[NC]：Non Case-sensitive，代表 match_uri 不比對大小寫差異。
[QSA]：Query String Append，代表保留網址尾端帶的 GET 參數，沒使用 flag 的預設是會把參數去掉的。
[QSD]：Query String Discard，與 QSA 相反的作用，apache v2.4 才有。
[R]: Redirect，代表用轉址的方式轉到新的網址，預設是 302 Status Code，如：[R=301]，也可以回傳 400、200、404 等的 Status Code，通常會跟 [L] 一起代表結束，也是除錯常用的 Flag
[DPI]: 不要再接續的 Rule 中結尾中加上 PathInfo，會在「五、一些小特性」的段落說明。
[F]: Forbdien 就是不給看啦！

※ more flags: http://httpd.apache.org/docs/2.4/rewrite/flags.html
```

範例

接著來舉例多個 Rules 加上 Flag 的功用，假設網站資料夾結構如下：

```
root/
  ├ match.html
  ├ rewrite.html
  ├ .htaccess
  └ secret/
     └ database_password.json
```

而範例的 `.htaccess` 的內容為：

```
### 設定 Rewrite 前面會加上的 path，預設會是 DocumentRoot
(如：/var/www/html)
RewriteBase /

# Rules 將會由上往下依序執行
# 直到最後一行或遇到有符合且有 L flag 的 Rule 就會停止

### Rule 1. 輸入 domain.com/match.html 將會顯示 rewrite.html 的內容
RewriteRule ^match\.html$ rewrite.html [NC,L]

### Rule 2. 輸入 domain.com/redirect.html 將會被導至
domain.com/rewrite.html
RewriteRule ^redirect\.html$ rewrite.html [NC,R=302,L]

### Rule 3. 如果輸入 domain.com/secret/… 這樣格式的網址，則去掉 secret/
後，轉回 root 並加上 .html
### $1 是正規表達式的 group capture，就是 $1=(.*) 取得括號內的值
RewriteRule ^secret/(.*)$ $1.html [NC,L]
```

在瀏覽器上輸入 `match.html` 或 `redirect.html` 均會顯示 `rewrite.html` 的內容，明顯是 Rule 2 將網址改變了，而 Rule 1 沒有，因為就是 302 Status Code 的關係。

在此建議可以使用 `POSTMAN` 或是 `cURL` 命令列或 Request 擷取等工具觀察整個 Response的差異，將會非常容易除錯，詳細原因將在「四、如何除錯」段落中解釋，以下筆者使用 `cURL` 示範：

1. Request： `http://localhost/match.html`

```
$ curl 'http://localhost/match.html'

I'm rewrite.html
```

2. Request： `http://localhost/redirect.html`

```
$ curl 'http://localhost/redirect.html'

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
```

```
</body></html>
```

與瀏覽器不同的是，這裡的 Response 會指出網址被轉移至另外一個新網址 `http://localhost/rewrite.html`，瀏覽器自動幫忙處理轉址的工作，根據上面回傳的網址，再發出另一個 Request 取得網頁內容，讓使用者僅感覺到網址與頁面改變而已。

這時候讀者可試著把 `RewriteBase` 這行註解，將會發現回傳結果出錯了，Rewrite 結果與 Domain 之間被加上 `DocumentRoot` 預設的路徑。

```
$ curl 'http://localhost/redirect.html'

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a
href="http://localhost/var/www/html/rewrite.html">here</a>.</p>
<hr>
<address>Apache/2.4.33 (Win32) OpenSSL/1.1.0h PHP/7.2.7 Server at
localhost Port 80</address>
</body></html>
```

前面兩個 Rule 僅是一個基本的示範，平常應用當然會難上許多，來繼續看第三個 Rule 吧

```
### Rule 3. 如果輸入 domain.com/secret/… 這樣格式的網址，則去掉 secret/
後，轉回 root 並加上 .html
### $1 是正規表達式的 group capture，就是取得那個滿足括號內的值
RewriteRule ^secret/(.*)$ $1.html [NC,L]
```

第三個 Rule 是禁止使用者訪問敏感的 `secret` 資料夾，來試著訪問 `database_password.json` 看能不能得到結果：

```
Vary: accept-language,accept-charset
Accept-Ranges: bytes
Content-Type: text/html; charset=utf-8
Content-Language: en
```

不管網址改成任何 `http://localhost/secret/...`，都會回傳 `404找不到頁面`，是理想中的狀況，非常棒！但是訪問 `http://localhost/secret/` 反而變成 `403禁止訪問` 了耶，咦？

```
$ curl 'http://localhost/secret/'

HTTP/1.1 403 Forbidden
Date: Sun, 07 Oct 2018 13:43:08 GMT
Server: Apache/2.4.33 (Win32) OpenSSL/1.1.0h PHP/7.2.7
Vary: accept-language,accept-charset
Accept-Ranges: bytes
Content-Type: text/html; charset=utf-8
Content-Language: en
```

這時候不知出了甚麼錯的話，可以將 Rule 的 Flag 加上 `R=302`，檢查最後改寫的結果出了甚麼狀況：

```
$ curl 'http://localhost/secret/'

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://localhost/.html">here</a>.
</p>
<hr>
<address>Apache/2.4.33 (Win32) OpenSSL/1.1.0h PHP/7.2.7 Server at
localhost Port 80</address>
</body></html>
```

發現 `$1` 沒有抓到任何字串，所以沒有任何檔名加上 `.html`，就變成 403 的結果，所以只要在 Rule 3 之前加上以下新的 Rule 就可以囉！

以上就是 RewriteRule 與簡單的 Debu 要再加上看得懂正規表達式就能懂一般常見的 Rules 囉！繼續看更難一點的 RewriteCond 吧～

三、RewriteCond

RewriteRule 僅僅只能判斷 Request URI 是否匹配而改寫 URI，但有很多需求是希望根據一些 Request Header（Host、User-agent) 與 Apache 的環境變數做改寫，先滿足某些條件後，再次 Rewrite URI，因此有了 RewriteCond 的出現，它的寫法結構如下：

```
RewriteCond [test_string] [match_string] [flags]
RewriteRule …
```

- test_string：要比對的條件
- match_string：符合的條件

以上這兩個都可以使用正規表達式撰寫，而且 RewriteCond 結束一定會接著一個 RewriteRule ，真正的範例會長這樣：

```
RewriteCond %{HTTP_USER_AGENT} (facebookexternalhit)
RewriteRule ^blog/(.*)$ fb-bot.html?path=$1&type=%1 [L]
```

以上意義等同於以下 Pseudo Code：

```
if ($HTTP_USER_AGENT == 'facebookexternalhit') {
 if (url.match('^blog/(.*)$')) {
 url = 'fb-bot.html?path=$1&type=facebookexternalhit';
 }
}
```

※ 前面說到 `$1` 是 Group Capture 的用法，而 RewriteCond 則是用 `%1` 表示

RewriteCond 可使用的變數

`${HTTP_USER_AGENT}` 是 RewriteCond 可使用的變數，有以下常見的變數：

```
%{HTTP_USER_AGENT}：User Agent
%{REQUEST_FILENAME}：訪問的檔案名稱

※ more variables：
https://httpd.apache.org/docs/current/mod/mod_rewrite.html#rewritecond
※ 其實變數也可以放在 RewriteRule 的 rewrite url 當中
```

以及判斷檔案時，很常見搭配這兩個 match 用法：

```
-d：directory. 代表如果有這個資料夾
-f：regular file. 代表如果有這個檔案
```

搭配起來寫法就像以下範例，代表著如果沒有這個檔案就轉到首頁：

```
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$ index.html
```

其實還有不同的 match 寫法，可以參考 Apache 官網的 RewriteCond 。

RewriteCond 的 Flag 用法

```
NC：Non case-sensitive
OR：就是 OR 條件，下面會說明
```

RewriteCond 也有 OR 跟 AND 的條件，先前提到 RewriteCond 後面一定會接一個 RewriteRule，有個特性是它只吃接續的第一個 rule ，來看下面的範例：

```
### 範例 A
RewriteCond 1
RewriteRule 1
RewriteRule 2
// 等同於
```

```
### 範例 B
RewriteCond 1
RewriteCond 2
RewriteRule 1
// 等同於
if (Cond1 && Cond2) {
 Rule1
}

### 範例 C
RewriteCond 1 [OR]
RewriteCond 2
RewriteRule 1
// 等同於
if (Cond1 || Cond2) {
 Rule1
}
```

瞭解 RewriteCond 跟 RewriteRule 的用法與每一行執行下來的邏輯，就可以更輕易的改寫網路上別人寫好的規則囉～

四、如何偵錯

先前在 RewriteRule 的段落有稍微簡單展示 Debug 的流程，這裡僅使用文字講解一些小秘訣。

1. 該使用甚麼 Debug 工具

使用 POSTMAN、cURL 等的工具，瀏覽器除了有 Cache 外，也會幫忙轉址，此時就沒辦法觀察第一次轉址的網址內容，譬如：最常見就是瀏覽器直接顯示轉址太多次的錯誤，但使用工具的話，就可以看到 Response 回傳的轉址結果。

2. 如何知道撰寫的 Regular Expression 是否正確？

把 RewriteRule 的 Flag 加上 `[R=302]`，302 Status Code 代表 `Moved Temporarily`，瀏覽器並不會 Cache 302 的轉址結果，但 301 會，可以確定 Rule 無誤後再拿掉或改為原本 301 就好，像這樣觀察轉址的結果：

```
RewriteRule ^(.*)$ =$1 [L,R=302]
```

3. 非得要修改正在運行中的網站怎麼辦？

前面提到瀏覽器 cache 的問題，若認為沒的沒問題，但訪問網站仍是舊有結果的話，就開啟私密瀏覽訪問看看，最後仍沒辦法只好重開 Apache 看看囉。

五、關於一些小特性

1. RewriteRule 在巢狀 `.htaccess` 當中不會取得完整 URI Path

直接看範例，假設資料夾結構如下：

```
root/
 ├ a/
   ├ b.html
   └ .htaccess
 ├ c.html
 └ .htaccess

### 兩個 .htaccess 都只有這一行內容
RewriteRule ^(.*)$ $1 [L]
```

1. Request： `c.html`

```
使用 root/.htaccess
Rule 結果： c.html
```

2. Request： `a/b.html`

```
使用 root/a/.htaccess
Rule 結果： b.html
```

沒錯，發生了不會拿到 `a/` 的路徑的情況。

※ Apache 會自動選擇最接近的 .htaccess 檔案（詳情會在下一段落說明）

3. 如果拿掉 `root/a/.htaccess` 檔案，重新 Request： `a/b.html`

`{REQUEST_URL}` 變數來取得 URI 囉！

```
RewriteCond %{REQUEST_URI} ^(.*)$
RewriteRule ^ %1 [L]
```

2. RewriteRule 自動附加在結尾的 PATH_INFO

有時候會希望遇到 Rule1 改寫之後，再傳遞至下一個 Rule2 判斷與改寫，可是會遇到後面莫名多了先前的 URI，來看一個簡單的示範例子：

```
RewriteRule ^(.*)$ web/$1 [NC]
RewriteRule ^(.*)$ sec======$1====== [NC,R=301]
```

Request： `a/b/c.html`

```
經過第一個 Rule 變成： web/a/b/c.html
最後到第二個 Rule 變成： sec======web/a/b/c.html/b/c.html======
```

發現它在 Rule1 的結果末端多了一個不需要的 `/b/c.html` ，這是因為 `PATH_INFO` 的緣故，若不需要後面 Path 的話，可以在 Rule 1 加上 `DPI` Flag 移除它，由於一些 php 的 CMS 或是 Framework 會使用到 `PATH_INFO` 的功能，所以是否關掉 `PATH_INFO` 的作用還是要注意一下囉！

3. 重新尋找 .htaccess 檔案

有時候要的 Rule 很純，就只是將所有 Request 都轉到 `web/` 資料夾下：

```
RewriteBase /
RewriteRule ^(.*)$ web/$1 [QSA,L]
```

此時有兩種解法，第一種是在 `web/` 資料夾下放一個空的 `.htaccess` ，第二種是可以參考下一點停止 Redirect Loop 的寫法，放在 RewriteRule 前面。

4. 停止 Redirect Loop 的情況

有時候會遇到無窮 Loop 的問題：

```
Request exceeded the limit of 10 internal redirects due to probable
configuration error. Use 'LimitInternalRecursion' to increase the
limit if necessary. Use 'LogLevel debug' to get a backtrace.
```

可以在所有 RewriteRule 之前加上判斷，若 Redirect Status 是 200 的話，就停止 Loop：

```
RewriteCond %{ENV:REDIRECT_STATUS} 200
RewriteRule ^ - [L]
```

不過建議還是檢查 RewriteRule 哪邊有寫錯的，畢竟這解不太算是萬靈藥。

Ref：https://stackoverflow.com/a/20930010

六、使用 Htaccess 的缺點

使用 Override 很方便，只要將 `.htaccess` 放到資料夾下面就有效果，但官方其實不推薦開啟 Override 的功能，它會降低效能，主要是有以下缺點：

1. 每次 Request 的巢狀搜尋

每一次 Request 都會使得 Apache 透過巢狀遞迴的方式搜尋 `.htaccess` 檔案，導致 Apache 緩慢，譬如送出這個 Request：

- Request： `/example/sub/index.html`

Apache 則會根據路徑尋找以下 `.htaccess` 檔案

最後 Apache 選擇離檔案最近的 `/var/www/example/sub/.htaccess`

2. 重複 Compile RewirteRule

由於每次 Request 都會巢狀搜尋 `.htaccess`，所以再遇到 RewriteRule 都會重新 Compile 一次，不像 Site Config 只會 Compile 一次後做 cache，所以 RewriteRule 非常多的話，也會導致 apache 緩慢。

筆者使用 Apache Benchmark 做了一個小型的壓力測試，連續訪問 10000 次，同時 1000 個連線，取其三次執行 `ab` 指令的平均值，分別比較是否有打開 Override 功能，和 Rewrite 數量多寡是否有影響，Rewrite 的狀況都是以跑到倒數第三行結束為主，且每行 RewriteRule 跟 RewriteCond 不重複。

		RewriteRule and RewriteCond's count							
		more (46 lines)				less (11 lines)			
		Connect (ms)	Processing (ms)	Waiting (ms)	Total (ms)	Connect (ms)	Processing (ms)	Waiting (ms)	Total (ms)
Overide	OFF	776	254	56	1030	965	151	31	1115
		862	231	52	1093	767	241	48	1008
		959	222	51	1181	750	222	44	972
				Avg.	**1101.3**			Avg.	**1031.7**
	ON	902	186	37	1089	928	182	37	1110
		884	230	49	1114	927	246	49	1173
		1201	235	54	1436	805	259	51	1064
				Avg.	**1213.0**			Avg.	**1115.7**

圖1. 針對 orverride 功能開關和 Rule 數量的壓力測試比較圖

如圖上的 Total 平均值，有開啟 Override 功能還是比沒開啟花的時間多了一些，而行數的多寡也會影響到花費的時間，但筆者還是有點疑惑，差0.1 秒好像也沒什麼關係吶，這邊還有請專業的讀者可以為筆者解惑呢 QQ

官方認為 `.htaccess` 主要還是給無法編輯 Site Config 的情況下使用，像是一台主機上有多站共用，但真的想用 `.htaccess` 效果又不想開啟 Override 的話怎麼辦呢？也是有一個取得中間值的做法。

在 Site Config 中預先指定 `.htaccess` 檔案路徑

把 Override 功能關掉，並使用 Include 指定引入的 `.htaccess` ，如下寫法：

```
    Include /var/www/.htaccess
    ...
</Directory>
```

但這方法也是有個小缺點：每次更新 `.htaccess` 都必須重新啟動 Apache 重新讀取設定。

如果主機流量不大的話，效能問題並沒這麼嚴重，最後還是得依據主機上的網站與情況，衡量哪種做法比較好哦！

. . .

Reference

- Apache.org — When (not) to use .htaccess files

以上就是 Rewrite 的介紹，有甚麼疑問歡迎一起留言探討

感謝讀者閱讀這篇長文 :)

Apache Rewrite Rewriterule Htaccess Rewritecond