## About this Demo

This is a PDF demo of the book, *.htaccess made easy*. It contains the entire Table of Contents and the first part of chapter 3. The complete book contains over 200 pages, all focused on improving your site with Apache/.htaccess.

Note! This demo does not contain all pages from the book. So any links pointing to missing pages will not work. Of course, all links work perfectly in the complete version of the book.

# .htaccess
## made easy

Learn how to redirect stuff.

Fine-tune site configuration.

Improve site performance.

Strengthen site-security.

Enhance usability.

# .htaccess
## made easy

A PRACTICAL GUIDE
FOR ADMINISTRATORS,
DESIGNERS & DEVELOPERS

*by* JEFF STARR

# contents

## 1.0 welcome


**1.1** Is this book for you? ..... 3
**1.2** Why .htaccess? ..... 3
**1.3** Goals of the book ..... 4
**1.4** Now you're an .htaccess ninja ..... 4
**1.5** Bonus material ..... 5
**1.6** Questions, comments, and errata ..... 5


## 2.0 the basics


**2.1** Required skills ..... 7
**2.2** Required software ..... 7
**2.3** Conventions used in this book ..... 8
**2.4** About the .htaccess file ..... 9
**2.5** How .htaccess files work ..... 10
**2.6** Basic structure and syntax ..... 11
**2.7** Character Definitions ..... 14
Server status-codes ..... 17
**2.8** Other requirements ..... 17
IfModule directives ..... 19
**2.9** Testing locally vs. testing live ..... 20
**2.10** Chapter Summary ..... 22

# 3.0 essential techniques


**3.1** Enable mod_rewrite ..... 25
**3.2** Enable symbolic links ..... 25
**3.3** Disable index views ..... 27
**3.4** Specify the default language ..... 28
**3.5** Specify the default character set ..... 29
**3.6** Disable the server signature ..... 30
**3.7** Configure ETags ..... 31
**3.8** Enable basic spell-checking ..... 32
**3.9** Combining Options ..... 33
**3.10** .htaccess starter-template ..... 34


# 4.0 optimizing performance


**4.1** Essential techniques ..... 37
**4.2** Enabling file compression ..... 37
Basic configuration ..... 38
Configure compression with mod_filter ..... 39
Compression tips and tricks ..... 40
Simple way to compress only the basics ..... 40
Compress everything except images ..... 41
Help proxies deliver correct content ..... 41
Force compression of mangled headers ..... 42
Compress additional file types ..... 43

Putting it all together ..... 44
**4.3** Optimizing cache-control ..... 46
Optimize cache-control with mod_expires ..... 47
Tuning ExpiresByType directives ..... 48
Additional file-types for mod_expires ..... 50
Cache-control for favicons ..... 51
Alternate method for cache-control ..... 52
Disable caching during site development ..... 54
Disable caching for scripts and other dynamic files ..... 55
**4.4** Using cookie-free domains ..... 56
**4.5** Configuring environmental variables ..... 57
Set the timezone ..... 59
Set the email address for the server administrator ..... 59

# 5.0 improving SEO

**5.1** Universal www-canonicalization ..... 61
Remove the www ..... 61
Require the www ..... 62
**5.2** Redirecting broken links ..... 63
Redirect all (broken) links from an external site ..... 65
Redirect a few external links ..... 67
**5.3** Cleaning up malicious links ..... 68

**5.4** Cleaning up common 404 errors ..... 70
Deny all requests for non-existent mobile content ..... 71
Universal redirect for nonexistent files ..... 72

# 6.0 .redirecting stuff

**6.1** Redirecting with mod_alias ..... 75
Redirecting subdirectories to the root directory ..... 76
Removing a subdirectory from the URL ..... 77
Redirect common 404-requests to canonical resources ..... 78
More rewriting tricks with mod_alias ..... 79
Redirect an entire website to any location ..... 79
Redirect a single file or directory ..... 81
Redirecting multiple files ..... 81
Advanced redirecting with RedirectMatch ..... 82
Combine multiple redirects into one ..... 83
Using multiple variables with RedirectMatch ..... 84
**6.2** Redirecting with mod_rewrite ..... 85
Basic example of mod_rewrite ..... 86
Targeting different server variables ..... 87
Redirecting based on the request-method ..... 88
Redirecting based on the complete URL-request ..... 88
Redirecting based on IP-address ..... 89
Redirect based on the query-string ..... 90
Redirect based on the user-agent ..... 91

Redirecting based on other server-variables ..... 91
REQUEST_URI ..... 92
HTTP_COOKIE ..... 92
HTTP_REFERER ..... 92
Send visitors to a subdomain ..... 93
Redirect only if the file or directory is not found ..... 93
Browser-sniffing based on the user-agent ..... 94
Redirect search queries to Google's search engine ..... 95
Redirect a specific IP-address to a custom page ..... 95
**6.3** Site-maintenance mode ..... 96
Features ..... 96
Customizing ..... 98
Send a custom message in plain-text ..... 98
Use a custom maintenance.html page ..... 98

# 7.0 tighten security

**7.1** Basic security techniques ..... 101
Prevent unauthorized directory browsing ..... 101
Disable directory-views ..... 102
Enable directory-views ..... 102
Enable directory-views, disable file-views ..... 102
Enable directory-views, disable specific files ..... 102
Disable listing of sensitive files ..... 102
Prevent access to specific files ..... 103

Prevent access to specific types of files ..... 103
Disguise script extensions ..... 104
Disguise all file extensions ..... 104
Require SSL/HTTPS ..... 105
Limit size of file-uploads ..... 106
**7.2** Disable trace and track ..... 106
**7.3** Prevent hotlinking ..... 108
Usage and customization ..... 109
Allow hotlinking from a specific directory ..... 111
Disable hotlinking in a specific directory ..... 111
**7.4** Password-protect directories ..... 112
Basic password protection ..... 114
Allow open-access for specific IPs ..... 115
Password protect specific files ..... 116
Allow access to specific files ..... 117
**7.5** Block proxy servers ..... 118
.htaccess proxy firewall ..... 118
Allow only specific proxies ..... 119
Block tough proxies ..... 120
**7.6** Controlling IP access ..... 121
Blocking and allowing specific IPs ..... 121
Denying and allowing ranges of IPs ..... 123
Denying and allowing based on CIDR number ..... 123
Denying and allowing based on wildcard IP-values ..... 125

Sending blocked IPs to a custom page ..... 126
Miscellaneous rules for blocking IP-addresses ..... 128
Block a partial-domain via network/netmask values
Limit access to Local Area Network (LAN)
Deny access based on domain-name ..... 128
Block domain.com but allow subdomain.domain.com ..... 129
**7.7** Whitelisting access ..... 129
**7.8** Blacklisting access ..... 132
Blacklist via the request-method ..... 132
Blacklist via the referrer ..... 133
Blacklist via cookies ..... 135
Blacklist via the user-agent ..... 136
Blacklist via the query-string ..... 138
Blacklist via the request ..... 139
Blacklist via request-URI ..... 140
Dealing with blacklisted visitors ..... 142
Redirect to homepage ..... 142
Redirect to an external site ..... 142
Redirect them back to their own site ..... 143
Custom processing ..... 143
Blacklisting with mod_alias ..... 143
Basic example of blacklisting with RedirectMatch ..... 144
The 5G Blacklist/Firewall ..... 144

# 8.0 enhance usability


**8.1** Serve custom error pages ..... 147
- Change the default error message ..... 148
- Redirect errors to a custom script ..... 148
- Redirect to an external URL ..... 149
- Provide a universal error-page ..... 149

**8.2** Serve browser-specific content ..... 150
- Detecting the user-agent with .htaccess ..... 150
- Serving customized content with PHP ..... 151

**8.3** Improving directory-views ..... 152
- Before diving in... ..... 152
- Basic customization ..... 153
- Customizing markup ..... 155
- Customizing with CSS ..... 157

**8.4** More usability enhancements ..... 158
- Basic spell-checking for requested URLs ..... 158
- Display source-code for dynamic files ..... 158
- Force download of specific file-types ..... 159
- Block access during at specific times ..... 160
- Quick IE tips ..... 161
- Remove the IE imagetoolbar ..... 161
- Minimize CSS image-flicker in IE6 ..... 161

# 9.0 .htaccess tricks for WordPress


**9.1** Optimizing WordPress Permalinks ..... 163
Canonical permalinks with www or non-www ..... 164
Cleaning-up dead-end permalinks ..... 164
Optimize date-based permalinks ..... 165
Redirect year/month/day permalinks to post-name only ..... 167
Redirect year/month permalinks to year/post-name only ..... 167
Redirect year/month permalinks to post-name only ..... 167
Redirecting WordPress Date Archives ..... 167
Eliminate all date-based archives ..... 169
Step 1. Add code to the root .htaccess file ..... 169
Step 2. Clean-up all instances of date-archive URLs ..... 169
Redirect any removed or missing pages ..... 170
Make dead pages go away ..... 171
Redirect entire category to another site ..... 172
**9.2** WordPress Multisite ..... 173
WordPress Multisite Subdomains on MAMP ..... 174
Step 1. Edit the Mac hosts file ..... 174
Step 2. Edit the Apache config file ..... 175
Step 3. Install & configure WordPress ..... 176
**9.3** Redirecting WordPress feeds ..... 177
Redirecting feeds to FeedBurner ..... 177

Redirecting category-feeds to FeedBurner ..... 178
Redirecting default query-string feed-formats ..... 180
**9.4** WordPress security techniques ..... 181
Block spam by denying access to no-referrer requests ..... 181
Secure posting for visitors ..... 183
Blocking spam on contact and other forms ..... 185

# 10.0 even more techniques

**10.1** Miscellaneous tricks ..... 187
Change the default index page ..... 187
Activate SSI for HTML/SHTML file types ..... 187
Retain rules defined in httpd.conf ..... 188
**10.2** Logging stuff ..... 189
Logging errors ..... 190
Logging access ..... 190
How to log mod_rewrite activity ..... 193
Customizing logs via .htaccess ..... 193
How to enable PHP error-logging ..... 195
Hide PHP errors from visitors ..... 195
Enable private PHP error logging ..... 196
**10.3** Troubleshooting guide ..... 197
Make sure Apache is running ..... 198
Check AllowOverride in httpd.conf ..... 198
Verify that a specific module is running ..... 199

Check the server logs ..... 200
Check HTTP status-codes ..... 200
Check your code for errors ..... 200
Is the directive allowed in .htaccess ..... 201
Isolating problems in .htaccess files ..... 201
**10.4** Where to get help with Apache ..... 202

# Epilogue

*Thank you* ..... 203
*About the author* ..... 203

- *Sections 2.7 and 10.3 are highlighted for quick reference.*

# httpd.conf sidebar menu

AllowOverride & FollowSymLinks ..... 26
Rename the .htaccess file ..... 35
Optimizing via AllowOverride ..... 58
Disable .htaccess files ..... 102

# chapter 3


**3.1** Enable mod_rewrite.....25
**3.2** Enable symbolic links.....25
**3.3** Disable index views.....27
**3.4** Specify the default language.....28
**3.5** Specify the default character set.....29
**3.6** Disable the server signature.....30
**3.7** Disable ETags.....31
**3.8** Enable basic spell-checking.....32
**3.9** Combining Options.....33
**3.10** .htaccess starter-template.....34


# essential techniques

.htaccess techniques may vary greatly from site to site, but there are a handful that are useful for virtually any website. From enabling functionality to logging activity, these essential techniques culminate in a "universal" .htaccess starter template.

When beginning a new website, you can streamline production by utilizing a predefined set of "template" files — or "boilerplate" files — that are common to most any site on the Web. Such files include stuff like the robots.txt file, favicons, JavaScript libraries, CSS templates, and so on. The same principle may be applied when configuring a site with .htaccess: some directives are super-useful for most any setup. In this chapter, we'll cover these essential techniques and then combine them into a starter-template designed to kick-start development and speed-up production.

Such as the most-awesome HTML5 Boilerplate: https://html5boilerplate.com/

There's even a jQuery Boilerplate: https://htaccessbook.com/9f

Boilerplate WordPress Theme: https://htaccessbook.com/o

And of course the .htaccess "boilerplate", aka the "starter" file (requires login): https://htaccessbook.com/members/

# 3.1 Enable mod_rewrite

As discussed in Chapter 2, certain servers may not have **mod_rewrite** enabled by default. The rewrite module is required for rewriting (redirecting) URLs from one page to another. To ensure that **mod_rewrite** is enabled on your server, install the temporary maintenance page and visit your site in a web-browser. The maintenance page requires **mod_rewrite** to work, so if you see the "We'll be right back..." message, that means you're good to go for URL-rewriting. If it's not working, then add the following line to the root .htaccess file:

```
<IfModule mod_rewrite.c>
      RewriteEngine On
</IfModule>
```

You only need to include this once, but it's perfectly safe to include it multiple times.

# 3.2 Enable symbolic links

You'll notice in the .htaccess starter-template that there are several listed values for the "`Options`" directives, located near the top of the file. These options are used to configure certain features, such as CGI, SSI, and symbolic links, or "symlinks" if you're nasty.

Symbolic links are used to integrate external directories into the filesystem. By default, files and directories not strictly beneath the "`DocumentRoot`" (i.e., the web-accessible root-

Apache docs for `mod_rewrite`:
https://htaccessbook.com/p

If things aren't working, see section 10.2 and 10.3 for troubleshooting and how to log `mod_rewrite` activity.

Learn more about the site-maintenance technique in section 6.3.

Apache docs on symbolic links:
https://htaccessbook.com/r

## httpd.conf
### AllowOverride & FollowSymLinks

As discussed, for symbolic links to work, Apache must be given explicit permission. The .htaccess method makes use of the Options directive, which itself must be enabled from within the httpd.conf file. Example:

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

For performance considerations, it is important to only enable `AllowOverride` in the specific directory in which it is required.

While working with the httpd.conf file, we may go ahead and enable symbolic links from that location, rather than via .htaccess. Just add this to your httpd.conf file:

```
<Directory "/var/www/html">
        Options FollowSymLinks
</Directory>
```

directory) is not a part of the Apache filesystem, and thus not configurable via .htaccess.

Apache provides several ways of bringing other parts of the filesystem under the `DocumentRoot`, including "`Alias`", "`ScriptAlias`", and "`ScriptAliasMatch`" directives, as well as via shell-induced symbolic links.

Regardless of which method is used, Apache will follow symbolic links only when given explicit permission, either in httpd.conf or .htaccess. See the blue "httpd.conf" sidebar for more information on using either of these techniques.

To enable symlinks via .htaccess, add the following directive to the target directory:

```
# enable symbolic links
Options +FollowSymLinks
```

If you know that your site won't be using any symbolic links, or if you've enabled them via the main configuration file, feel free to comment-out or remove this directive. In general it's

The `SymLinksIfOwnerMatch` directive may be used in place of FollowSymLinks — either works to enable symbolic links. Read more in the Apache Docs: https://htaccessbook.com/r

Quick tutorial explaining two ways to create symbolic links: https://htaccessbook.com/s

Straight-up post on "How to Create a Symlink": https://htaccessbook.com/t

good practice to disable any functionality that's not needed, but technically it's fine to repeat the `FollowSymLinks` directive. See the blue sidebar for more about httpd.conf.

# 3.3 Disable index views

By default, Apache will display the contents of any directory that doesn't include some sort of index file. For some directories, this may be a useful for public content, but most of the time it's undesirable. For example, directories that contain sensitive core files, such as WordPress' `/wp-admin/` and `/wp-includes/` — there's no reason to list the contents of these directories for the public.

To disable directory listings for all directories, add this line to the site's root .htaccess file:

```
# disable directory listing
Options -Indexes
```

To test, visit any directory that doesn't contain an index file. If you need to enable directory listing for some specific directory, create an .htaccess file for it, and add the following lines:

**Index of /Pink-Floyd/More**

| Name | Last modified | Size | Description |
|---|---|---|---|
| Parent Directory | | - | |
| 01 - cirrus minor.mp3 | 15-Jun-2012 20:40 | 12M | |
| 02 - the nile song.mp3 | 15-Jun-2012 20:40 | 7.8M | |
| 03 - crying song.mp3 | 15-Jun-2012 20:41 | 8.2M | |
| 04 - up the khyber.mp3 | 15-Jun-2012 20:41 | 5.1M | |
| 05 - green is the colour.mp3 | 15-Jun-2012 20:41 | 6.8M | |
| 06 - cymbaline.mp3 | 15-Jun-2012 20:41 | 11M | |
| 07 - party sequence.mp3 | 15-Jun-2012 20:41 | 2.7M | |
| 08 - main theme.mp3 | 15-Jun-2012 20:41 | 13M | |
| 09 - ibiza bar.mp3 | 15-Jun-2012 20:41 | 7.5M | |
| 10 - more blues.mp3 | 15-Jun-2012 20:41 | 5.1M | |
| 11 - quicksilver.mp3 | 15-Jun-2012 20:42 | 16M | |
| 12 - a spanish piece.mp3 | 15-Jun-2012 20:42 | 2.5M | |
| 13 - dramatic theme.mp3 | 15-Jun-2012 20:42 | 5.2M | |
| Seabirds.mp3 | 15-Jun-2012 20:42 | 731K | |

*By default, Apache displays the contents of directories that don't include an index file.*

**Error 403 — Forbidden**

You don't have permission to access `Requested URL: http://bluefeed.net/Pink-Floyd/More/` on this server..

*Apache returns a "403 — Forbidden" response when directory listing is disabled. This prevents scripts, bots, and humans from any meddling.*

There are several other "httpd.conf" sidebars throughout the book. Refer to the Table of Contents for more info.

WordPress protects these directories with a blank `index.php` file, which is an alternate way of doing it.

Unless disabled by the web-host. Some hosts disable index-views as an added security measure. Ask if unsure.

In general, it's preferred to configure Apache directives in the `httpd.conf` file to ensure optimal performance.

```
# ENABLE DIRECTORY LISTING
Options All +Indexes
```

As with other `Options` directives, `AllowOverride` must be enabled in the httpd.conf file.

# 3.4 Specify the default language

Apache makes it easy to control the default language used by different directories. For example, if you provide translated versions of your web pages, each in their own directory, you can set the default language for each with Apache's `DefaultLanguage` directive.

This can be a huge time-saver when working with multilingual sites — no more messing with meta tags to set the language. To specify the default language for the entire site, place the following directive near the top of the root .htaccess file:

```
DefaultLanguage en
```

Here, we're specifying English as the default language using its two-digit abbreviation. This will cascade down the filesystem and apply to all directories and files therein.

As mentioned, the default language may be overridden in specific subdirectories. If we have a subdirectory for a French translation, for example, we would create an .htaccess file

See the blue httpd.conf sidebar on page 26 for basic information about enabling `AllowOverride`.

Note that multiple language variations may be specified like so: `DefaultLanguage el, em, en`

`DefaultLanguage` applies to all files in the directive's scope, which excludes files that have an explicit language extension, such as "`.ch`" or "`.de`", see: https://htaccessbook.com/u

Quick example showing how to override language-defaults for specific file-types: `AddLanguage en .html .css .js`

and add the following line:

```
DefaultLanguage fr
```

See the footer for more information and resources about setting the default language.

# 3.5 Specify the default character set

It's also possible to specify the default character-set (charset) for all of your HTML and plain-text content. Apache's `AddDefaultCharset` directive may be used to add a default charset parameter to the server-response header. Basically, that just means the server lets the browser know how the content is encoded.

The screenshot at right shows an example of this. First, the request to Google.com is made by the web browser. The server then responds with `UTF-8` as the charset parameter for the `Content-Type` header.

By default, Apache disables `AddDefaultCharset`. If you enable it using the "`On`" value, the default charset is ISO-8859-1. Otherwise, to specify your own default charset, such as UTF-8, add the following directive to the root .htaccess file, preferably somewhere near the top of the file:

```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:12.0) Ge
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: NID=60=CZGVI8Zz7mwHFTSKpQakTKy8MveJ2IIlvXM_ekt-E-
Cache-Control: max-age=0

HTTP/1.1 200 OK
Date: Mon, 18 Jun 2012 22:49:05 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Server: gws
Content-Length: 25341
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
```

The screenshot shows Google.com returning the UTF-8 charset along with the `Content-Type` header.

The scoop about UTF-8: http://www.utf-8.com/

ISO 8859-1 character set overview: https://htaccessbook.com/7d

The charset should be an IANA registered charset value for use in MIME media-types. See https://htaccessbook.com/v

```
AddDefaultCharset utf-8
```

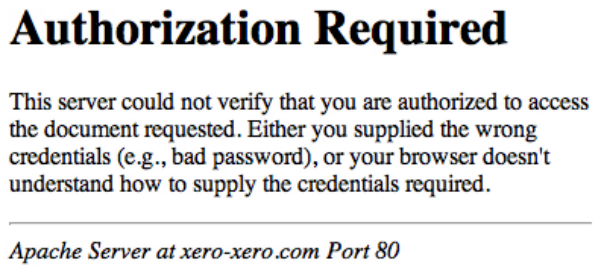
Specifying the default charset via server-response header should override any charset specified in the body of the response, such as those included via `<meta>` tag. If your web pages specify a character set via `<meta>` tags, and you're not hosting any non-HTML content, it's fine to disable `AddDefaultCharset` and just roll with the `<meta>` tags, although Google PageSpeed suggests going the `AddDefaultCharset` route for better performance.

```
AddDefaultCharset Off
```

# 3.6 Disable the server signature

Unless you have reason to do otherwise, disabling your server signature is a good way to keep sensitive information out of the wrong hands. Why broadcast sensitive server details such as which port you're using, your server name, and possibly other information?

Fortunately this behavior is disabled by default, but some hosts enable it for certain configurations. If you're sure you don't need to display that information, it should be disabled as a basic security measure. As seen in the "Authorization Required" screenshot, server-generated documents include the default server-signature displayed in the footer area.




For more info on setting the default charset with .htaccess, check out: https://htaccessbook.com/w

Quick Reference Guide for HTML Meta Tags https://htaccessbook.com/7e

How to disable (apache's) server signature https://htaccessbook.com/x

How (and why) to disable apache server signature https://htaccessbook.com/y

It's possible to customize the footer-line using the `ServerTokens` directive or by specifying "EMail" as the value for the `ServerSignature` directive.

Unless you have reason to do otherwise, it's best to disable this feature, preferably via the main configuration file, but it's also possible using the following line in the root .htaccess:

```
ServerSignature Off
```

Along with the other essential techniques in this chapter, this directive is included in the .htaccess starter-template included with this book. We'll get to that after seeing two more widely used .htaccess techniques.

# 3.7 Configure ETags

According to the Yahoo! Developer team, disabling ETags can improve site performance by decreasing response-sizes by around 12 Kilobytes each. There's a long, sordid story that goes with the "what", "why" and "how" of ETags, but let's not get into that here. Rather, let's stay focused on the task at hand: best practices and essential .htaccess techniques. And when it comes to ETags, it all depends on how your website is hosted. If you're hosting your site on a single server, Apache's default configuration should work fine. If, on the other hand, your site is hosted on a network of servers, ETags are probably *decreasing* performance and should be disabled.

The `ServerSignature` directive allows the configuration of a trailing footer-line under server-generated documents such as error-messages, directory-listings, and module output. More details in the Apache Docs: https://htaccessbook.com/z

Yahoo's "Best Practices for Speeding Up Your Web Site": https://htaccessbook.com/10

Steve Souders' "High Performance Web Sites": https://htaccessbook.com/86

Fortunately, Apache makes it easy to override default settings using the **FileETag** directive. For example, to **disable ETags**, add the following line to the root .htaccess:

```
FileETag none
```

Whenever possible, I like to keep core directives — such as `DefaultLanguage`, `ServerSignature`, and `FileETag` — located at the beginning of the .htaccess file. It makes sense mostly from a functional point of view, but really they may be placed anywhere.

# 3.8 Enable basic spell-checking

Apache has a built-in spelling-check module — ironically named "**mod_speling**" — that can "fix" basic spelling and capitalization errors in the URL request. The Apache documentation really explains it best:

> *[mod_speling] does its work by comparing each document name in the requested directory against the requested document name without regard to case, and allowing up to one misspelling (character insertion / omission / transposition or wrong character).*

For example, let's say a visitor misspells the URL to your site's "About" page, located at `https://example.com/about/`. With `CheckSpelling` enabled, one of the following things will happen:

Some servers require more "convincing" to disable ETags, so if FileETag isn't cutting it, try adding this:

```
<IfModule mod_headers.c>
        Header unset ETag
</IfModule>
```

More info about ETags: https://htaccessbook.com/9l

Read more about FileETag in the Apache Docs: https://htaccessbook.com/11

For more details about how `mod_speling` works, visit the Apache Documentation: https://htaccessbook.com/12

- Apache can't find a matching document, and so delivers a "document not found" error.
- Apache finds something that "almost" matches the URL request, and redirects to it.
- Apache finds multiple possible matches and presents a list of options to the client.

`CheckSpelling` is disabled by default, but the general consensus is that it's useful for SEO[1]. The following directive is also included in the starter-template, and may be added to the root .htaccess file to enable basic spell-checking on your site[2]:

```
<IfModule mod_speling.c>
      CheckSpelling On
</IfModule>
```

"Mission accomplished," as it were. Further details are available in the Apache Docs.

# 3.9 Combining Options

Two of the techniques in this chapter — `Indexes` and `FollowSymlinks` — are enabled via the `Options` directive[3]. It's perfectly fine to write them separately, like so:

```
Options -Indexes
Options +FollowSymlinks
```

[1] SEO = Search Engine Optimization
https://htaccessbook.com/2i

[2] If you don't want the server trying to "guess" at which URL to serve, but would like to ensure that all URLs are returned in lowercase format, you may want to try the `CheckCaseOnly` directive:
https://htaccessbook.com/13

[3] The Options directive is part of the Apache core:
https://htaccessbook.com/14

By default, the `Options` directive is set to "`All`", which is overridden by any `Options` values that are more specific. Similarly, when *multiple options* are specified, only the most specific is applied, again by default. The key to specifying multiple values is the plus-sign "+" or minus-sign "-", which instruct Apache to merge the options or remove them from the `Options` currently in place. This enables us to combine options into a single directive:

```
Options -Indexes +FollowSymlinks
```

In this fashion, as many options as needed may be combined. See the Apache Docs for further information and examples.

# 3.10 .htaccess starter-template

Any or all of the techniques in this chapter may be applied to your site, but as discussed in the chapter-introduction, they are all general and useful enough to be included in just about any .htaccess file, collectively as a foundation.

To help streamline development, I've combined these essential techniques into an .htaccess "boilerplate", or "starter" template. It's a simple and flexible template to customize and build upon. Anything that's not needed may be removed or commented-out with a hash-symbol "#". I use a similar file, tuned to my particular server setup, for new projects or implementing .htaccess on client sites.

The Options directive is part of the Apache core, as described in the official documentation: https://htaccessbook.com/14

The .htaccess starter-template contains only directives and techniques that are covered explicitly in this book. Log in to the Member's Area for current download (requires login): https://htaccessbook.com/members/

**Get the book to read more!**

https://htaccessbook.com/9y

> Jeff has long been the go-to guy for all things .htaccess. I always think of .htaccess as pretty much programming voodoo, where saying a chant as you save the file has about the same chance of working as any actual characters I type. But of course that isn't true, and I'm glad Jeff is here to help us through it with this book.
>
> — **Chris Coyier**
> **css-tricks.com**

> I'll probably never master regex or the mysterious ways of the .htaccess file but thanks to Jeff I am a few steps closer to being a little more in control of both! It's all in here, from redirects, to optimisation through to a dedicated section on using .htaccess to super charge your WordPress site. I can't recommend it enough — grab a copy now.
>
> — **Keir Whitaker**
> **keirwhitaker.com**

> If you ever wanted to finally fully understand .htaccess and all its peculiarities, this comprehensive, well-written book will be just what you are looking for.
>
> — **Vitaly Friedman**
> **smashingmagazine.com**

> In .htaccess made easy, Jeff Starr has carried out the difficult task of creating a guide that provides instruction for beginners while acting as a useful reference that advanced users can dip in and out of. This beautiful guide to .htaccess is delivered in Jeff's good-humoured and engaging style, making it readable, accessible, and a must-have for your web development library.
>
> — **Siobhan McKeown**
> **siobhanmckeown.com**

.htaccess made easy
htaccessbook.com
jeff starr | perishable press

$50.00
ISBN 978-0-9835178-3-2
55000>
9 780983 517832
.htaccess made easy