How does rewriting work ?

First, let's examine an URL. Here is an URL

```
http://www.example.com/foo/bar?x=toto&y=titi
```

It has 4 parts :

1. http : The protocol
2. www.example.com : the servername
3. foo/bar : the path
4. x=toto&y=titi : the query string

apache only rewrite the path. Apache can add things to the query string with its QSA flag. Apache can only add to the query string. It has no other possibility to change it.

So, how does it work ?

The PATH is compared with the first rule. if it matches, then it is rewritten. apache goes to the next rule. It compares the (maybe changed) PATH with this rule and, if it matches the PATH is modified. Apache goes on until there is no more rewrite rules. When this is done, apache has a new form of the url and deals with new URL.

Rules can do more than just rewriting the PATH. I just show 2 things that happen here :

1. A rule can say : “OK, I've rewritten the path. No need to go further” with its L (last) flag
2. A rule can also say : “I rewrie the path, but I also add something to the query string”, with its QSA flag

It is important to note that this happens inside the apache server only. Your browser is not aware of this, and will never see the result of the rewriting rules.

example1 :

1. You request http://www.example.com
2. The rewrite rules transform it to http://www.example.com/doku.php
3. Apache now has to deal with this new URL. Hopefully, it will know what to do with php files and it will run doku.php ; but this is beyond the scope of rewrite rules.

example 2 :

1. You request http://www.example.com/foo:bar
2. The rewrite rules transform it to http://www.example.com/doku.php?id=foo:bar
3. Apache now has to deal with this new URL. Hopefully, it will know what to do with php files...

Default rules explained

These are the default rules, presented in `.htaccess.dist`

1. if PATH starts with _media/, give it to `lib/exe/fetch.php` and we're done

```
RewriteRule ^_media/(.*)              lib/exe/fetch.php?media=$1
[QSA,L]
```

   Example and explanation: Given the URL `http://example.com/_media/wiki:dokuwiki.png` we have

   - `^_media/(.*)` that matches `_media/wiki:dokuwiki.png`
     - The part corresponding to `(.*)`[1] is associated with `$1`. So `$1` is now `wiki:wiki.png`.
   - it is rewritten as `lib/exe/fetch.php?media=wiki:dokuwiki.png`
     - Note that for the `?media=wiki:dokuwiki.png` part to be effectively added to the URL, we have to use the `[QSA]` flag
   - Our job is done. ie no need to test for the next rewrite rule. Since this rule applied, it is the **L**ast one in this run. That's the role of the `[L]` flag.
   - In the end, apache now has to deal with this URL:

     `http://example.com/lib/exe/fetch.php?media=wiki:dokuwiki.png`

2. if PATH starts with _details/, give it to lib/exe/detail.php and we're done

```
RewriteRule ^_detail/(.*)             lib/exe/detail.php?media=$1
[QSA,L]
```

   Example and explanation: This is the same as above, but with `_details` instead of `_media`.

3. if PATH starts with _export/**FOO**/, give it to doku.php?do=export_**FOO** and we're done

```
RewriteRule ^_export/([^/]+)/(.*)     doku.php?do=export_$1&id=$2
[QSA,L]
```

   Example and explanation: This is almost the same as the preceding cases. The only difference is we have 2 groups of parentheses instead of 1. We'll have `$1` and `$2` as “variables”. So, given the URL `http://example.com/_export/raw/wiki:syntax` we have

   - `_export/([^/]+)/(.*)` that matches `_export/raw/wiki:syntax`
     - The part corresponding to `([^/]*)`[2] is associated with `$1`. So `$1` is now `raw`.
     - The part corresponding to `(.*)`[3] is associated with `$2`. So `$2` is now `wiki:syntax`.
   - it is rewritten as `doku.php?do=export_raw&id=wiki:syntax`
   - Note that for the `?do=export_raw&id=wiki:syntax` part to be effectively added to the URL, we have to use the `[QSA]` flag
   - Our job is done. ie no need to test for the next rewrite rule. Since this rule applied, it is the **L**ast one in this run. That's the role of the `[L]` flag.
   - In the end, apache now has to deal with this URL:

     `http://example.com/doku.php?do=export_raw&id=wiki:syntax`

4. if PATH is empty (ie full URL is just `http://example.com`), give it to doku.php, and we're done.

```
RewriteRule ^$                          doku.php  [L]
```

The resulting URL is

```
http://example.com/doku.php
```

5. if the PATH does not map to a file

```
RewriteCond %{REQUEST_FILENAME}       !-f
```

and if it does not map to a folder

```
RewriteCond %{REQUEST_FILENAME}       !-d
```

then give it to doku.php, and we're done

```
RewriteRule (.*)                        doku.php?id=$1  [QSA,L]
```

It transforms

```
http://example.com/faq:footerbuttons<code>into
http://example.com/doku.php?id=faq:footerbuttons
```

6. Because of the tests performed in 5, we are sure that from here on, we deal with an existing file or folder.
7. if the url/file is index.php, then use doku.php instead of index.php and continue with next rule, as there is no `[L]` flag.

```
RewriteRule ^index.php$ doku.php
```

It transforms

```
http://example.com/index.php<code>into
http://example.com/doku.php
```

8. There is no more rule, so no special rewrite ⇒ just serve the file or folder as usual.

Here is presented the same information but more friendly

initial URL	**rewritten URL** (what apache will effectively deal with)
`http://example.com/_media/wiki:dokuwiki.png`	`http://example.com/lib/exe/fetch.php?media=wiki:dokuwiki.png`
`http://example.com/_details/wiki:dokuwiki.png`	`http://example.com/lib/exe/fetch.php?media=wiki:dokuwiki.png`
`http://example.com/lib/image/page.png`	No rewrite because `lib/image/page.png` is a file that exists.
`http://example.com/_export/raw/wiki:syntax`	`http://example.com/doku.php?do=export_raw&id=wiki:syntax`
`http://example.com/index.php`	`http://example.com/doku.php`

1)

that means all the characters that are left. (after `media/`)

2)

that means the first word left after `_export/`

3)

that means everything that's left (after `_export/raw/`)