Поговорим о:

htaccess и работа с modrewrite.
Основа сайта.

htaccess

.htaccess (с точкой в начале имени) - это файл-конфигуратор Apache-серверов, который дает возможность конфигурировать работу сервера в отдельных директориях (папках), не предоставляя доступа к главному конфигурационному файлу (apache/conf/httpd.conf).

Особенности

1. В нем можно переопределить большое количество директив, прописанных в главном файле httpd.confg
2. Если расположить htaccess-файл в корневой каталог, он распространится на весь сайт (исключения составят только те каталоги, в которых расположен собственный конфигурационный файл, и каталоги, расположенные ниже в древовидной структуре)
3. Разместить htaccess-файл можно в любой каталог, а его директивы будут применены ко всем подкаталогам
4. Htaccess не доступен пользователю для просмотра из браузера, так как относится к категории «системные».

Возможности

Директивы простого перенаправления (редирект);
Директивы сложного перенаправления (mod_rewrite);
Индексные страницы;
Обработка ошибок;
Определение кодировки;
Управление доступом к директориям и файлам;
Паролирование директорий;
Опции PHP.

.htaccess – перенаправление/изменение

Если на сайте были перемещены страницы на новые адреса, то пользователь или поисковый робот, обратившись по старому адресу, наверняка их не увидит. Чтобы склеить старый и новый адреса страницы можно применить простой 301 редирект. При этом url в адресной строке браузера изменится.
Для этого в файле конфигурации htaccess необходимо прописать следующий код:

Redirect 301 /staraya.html http://vash-sait.ua/novaya.html

Если нужно **изменение** – прозрачное перенаправление без изменения адресной строки, то:

```
RewriteEngine on
RewriteRule ^old\.html$ new.html
```

.htaccess – RewriteCond

Директива **RewriteCond** определяет условия для какого-либо правила. Перед директивой **RewriteRule** располагаются одна или несколько директив **RewriteCond**. Следующее за ними правило преобразования используется только тогда, когда URI соответствует условиям этой директивы и также условиям этих дополнительных директив.

Может использовать переменные сервера - это переменные вида %{NAME_OF_VARIABLE}

где NAME_OF_VARIABLE может быть строкой взятой из следующего списка:

HTTP заголовки:	**соединение & запрос:**	-
HTTP_USER_AGENT	REMOTE_ADDR	
HTTP_REFERER	REMOTE_HOST	
HTTP_COOKIE	REMOTE_USER	
HTTP_FORWARDED	REMOTE_IDENT	
HTTP_HOST	REQUEST_METHOD	
HTTP_PROXY_CONNECTION	SCRIPT_FILENAME	
HTTP_ACCEPT	PATH_INFO	
	QUERY_STRING	
	AUTH_TYPE	

внутренние сервера:	**системные:**	специальные:
DOCUMENT_ROOT	TIME_YEAR	
SERVER_ADMIN	TIME_MON	API_VERSION
SERVER_NAME	TIME_DAY	THE_REQUEST
SERVER_ADDR	TIME_HOUR	REQUEST_URI
SERVER_PORT	TIME_MIN	REQUEST_FILENAME
SERVER_PROTOCOL	TIME_SEC	IS_SUBREQ
SERVER_SOFTWARE	TIME_WDAY	
	TIME	

Что бы вы могли увидеть свои - а они у каждого сервера и домуента свои - необходимо например создать вот такой файлик info.php:

```
<?php
phpinfo () ;
?>
```

.htaccess – RewriteCond (Условия)

Условие это шаблон условия, т.е., какое-либо регулярное выражение применяемое к текущему экземпляру СравниваемаяСтрока, т.е., СравниваемаяСтрока просматривается на поиск соответствия Условие. Условие это perl совместимое регулярное выражение с некоторыми дополнениями:

- '!' (восклицательный знак) для указания несоответствия шаблону.
- '<Условие' (лексически меньше)
- '>Условие' (лексически больше)
- '=Условие' (лексически равно)
- '-d' (является ли каталогом)
- '-f' (является ли обычным файлом)
- '-s' (является ли обычным файлом с ненулевым размером)
- '-l' (является ли символической ссылкой)
- '-F' (проверка существования файла через подзапрос)
- '-U' (проверка существования URL через подзапрос)

.htaccess – RewriteCond (Флаги)

Дополнительно вы можете устанавливать специальные флаги для Условие добавляя **[flags]**

- 'nocase|NC' (регистронезависимо)

Регистр не имеет значение, т.е., нет различий между 'A-Z' и 'a-z' как в дополнении СравниваемаяСтрока так и Условие. Этот флаг эффективен только для сравнений между СравниваемаяСтрока и Условие. Он не работает при проверках в файловой системе и в подзапросах.

- 'ornext|OR' (либо следующее условие)

Используйте для комбинирования условий в правилах OR вместо AND. Типичный пример:

```
RewriteCond %{REMOTE_HOST} ^host1.* [OR]
RewriteCond %{REMOTE_HOST} ^host2.* [OR]
RewriteCond %{REMOTE_HOST} ^host3.*
RewriteRule ...some special stuff for any of these hosts...
```

Без этого флага вы должны были бы написать это условие/правило три раза.

.htaccess – RewriteRule

Директива **RewriteRule** и есть настоящая рабочая лошадка преобразований.
Эта директива может встречаться более одного раза. Каждая директива, в этом случае, определяет одно правило преобразования. Порядок определений этих правил важен, потому что этот порядок используется при обработке правил во время работы.

Шаблон это perl совместимое регулярное выражение которое применяется к текущему URL. Здесь под «текущим» подразумевается значение URL когда применяется это правило. Этот URL не обязательно совпадает с первоначально запрошенным URL, потому что любое количество правил возможно уже были применены к нему и соответственно преобразовали его.

.htaccess – RewriteRule (синтаксис рег. выражений)

Текст:

. Любой одиночный символ
[chars] Класс символов: Один из символов
[^chars] Класс символов: Ни один из символов
text1|text2 Альтернатива: text1 или text2

Кванторы (символы для обозначения количественных отношений):
? 0 или 1 из предшествующего текста
* 0 или N из предшествующего текста (N > 0)
+ 1 или N из предшествующего текста (N > 1)

макрос "$1" обозначает ту часть исходного пути, которая расположена внутри первой пары скобок "RewriteRule ^(.*)....." , $2 – внутри второй пары и так далее.

.htaccess – RewriteRule (синтаксис рег. выражений)

Группировка: (text) Группировка текста

Маркеры: ^ Маркер начала строки и $ Маркер конца строки

Экранирование:
\char экранирование конкретного символа (к примеру для указания символов ".[]()" и т.д.)

Кроме того, в mod_rewrite символ отрицания (NOT) ('!') — допускаемый префикс в шаблоне. Это даёт вам возможность инвертировать действие шаблона; ну к примеру скажем: "если текущий URL не совпадает с этим шаблоном". Это может быть использовано в особых случаях, когда проще найти шаблон для несоответствия, или в качестве последнего правила, работающего по умолчанию.

Примечание
При использовании символа NOT (не) для инвертирования действия шаблона вы не можете иметь сгруппированные части групповых символов в шаблоне. Это невозможно потому что когда нет соответстви

.htaccess – RewriteRule (флаги)

Третий аргумент директивы RewriteRule. Флаги — это разделённый запятыми, заглавные спец символы заключенные в квадратные скобки. Список флагов модуля преобразований:

'redirect|R [=code]' (вызывает редирект)

Префикс в Подстановке вида http://thishost[:thisport]/ (создающий новый URL из какого-либо URI) запускает внешний редирект (перенаправление). Если нет накакого кода в подстановке ответ будет с HTTP статусом 302 (ВРЕМЕННО ПЕРЕМЕЩЕН). Если вы хотите использовать дркгие коды ответов в диапазоне 300-400, просто напишите их в виде числа или используйте одно из следующих символических имён: temp (по-умолчанию), permanent, seeother. Используйте это в директивах, которые должны преобразовывать некие виртуальные URL в реальные и возвращать их клиенту, например, преобразовывать «/~» в «/u/» или всегда добавлять слэш к /u/user, и т.д.

Для остановки процесса преобразования, вам также нужно написать флаг 'L'. Итого запоминаем:

- ☑️ При внешнем редиректе меняется урл адреса в строке браузера - "[R=301,L]"
- ☑️ При внутреннем - не меняет урл адреса в строке браузера - "[R=301] или [L]"

'forbidden|F' (делает URL запрещенным)

Это делает текущий URL запрещённым, например, клиенту немедленно отправляется ответ с HTTP статусом 403 (ЗАПРЕЩЕНО). Используйте этот флаг в сочетании с соответствующими RewriteConds для блокирования URL по некоторым критериям.

'gone|G' (делает URL «мёртвым»)

Этот флаг делает текущий URL «мертвым», т.е., немедленно отправляется HTTP ответ со статусом 410 (GONE). Используйте этот флаг для маркировки «мертвыми» не существующие более страницы.

.htaccess – RewriteRule (флаги)

'proxy|P' (вызвает прокси)

Этот флаг помечает подстановочную часть как внутренний запрос прокси и немедленно (т.е., процесс преобразования здесь останавливается) пропускает его через прокси модуль.
Примечание: Для того чтобы это использовать убедитесь что у вас есть работающий прокси модуль на вашем сервере Apache и строка подстановки это реальный URI .

'last|L' (последнее правило)

Остановить процесс преобразования на этом месте и не применять больше никаких правил преобразований. Это соответствует оператору last в Perl или оператору break в языке C. Используйте этот флаг для того, чтобы не преобразовывать текущий URL другими, следующими за этим, правилами преобразований.

'next|N' (следуюший раунд)

Перезапустить процесс преобразований (начав с первого правила). В этом случае URL снова сопоставляется неким условиям, но не оригинальный URL, а URL вышедший из последнего правила преобразования. Это соответствует оператору next в Perl или оператору continue из языка C. Используйте этот флаг для перезапуска процесса преобразований, т.е., безусловному переходу на начало цикла.
Однако будьте осторожны, для того чтобы не сделать бесконечный цикл!

'chain|C' (связь со следующим правилом)

Этот флаг связывает текущее правило со следующим. Это имеет следующий эффект: если есть соответствие правилу, процесс продолжается как обычно, т.е., флаг не производит никакого эффекта. Если правило не соответствует условию, все следующие, связанные правила, пропускаются.

'type|T=MIME-тип' (принудительно установить MIME тип)

Принудительно установить MIME-тип целевого файла в MIME-тип. К примеру, это можно использовать для имитации mod_alias директивы ScriptAlias которая принудительно устанавливает для всех файлов внутри отображаемого каталога MIME тип равный «application/x-httpd-cgi».

.htaccess – RewriteRule (флаги)

'nocase|NC' (не учитывать регистр)

'qsappend|QSA' (добавлять строку запроса)
Этот флаг указывает механизму преобразований на добавление а не замену, строки запроса из URL к существующей, в строке подстановки. Используйте это когда вы хотите добавлять дополнительные данные в строку запроса с помощью директив преобразований.

'noescape|NE' (не экранировать URI при выводе)
Этот флаг не даёт mod_rewrite применять обычные правила экранирования URI к результату преобразования. Обычно, специальные символы (такие как '%', '$', ';', и так далее) будут экранированы их шестнадцатиричными подстановками ('%25', '%24', и '%3B', соответственно); этот флаг не дает это делать. Это позволяет символам процента появлятся на выходе , как в
RewriteRule /foo/(.*) /bar?arg=P1\%3d$1 [R,NE]
для которого '/foo/zed' преобразовывалось бы в безопасный запрос '/bar?arg=P1=zed'.

.htaccess – сложное перенаправление

Перенаправление домена с www на без www:

```
RewriteEngine On
RewriteCond %{HTTP_HOST} ^www.mysite.ua$ [NC]
RewriteRule ^(.*)$ http://mysite.ua/$1 [R=301,L]
```

Перенаправление посетителей на разные старницы в зависимости от IP-адреса посетителя:

```
RewriteEngine On
SetEnvIf REMOTE_ADDR 183.11.101.1 REDIR="redir"
RewriteCond %{REDIR} redir
RewriteRule ^/$ /kontakt.html
```

Перенаправление на время обновления веб-ресурса:

```
RewriteEngine on
RewriteCond %{REQUEST_URI} !/info.html$        //зашел не на info.html
RewriteCond %{REMOTE_HOST} !^14.124.354.80   //и не с 14.124.354.80
RewriteRule $ http://vash-sait.ua/info.html [R=302,L]
```

Изменение адресов URL к более простой форме для посетителей и поисковых роботов.

```
RewriteRule ^products/([^/]+)/([^/]+)/([^/]+)
product.php?cat=$1&brand=$2&prod=$3
```

Данное правило позволяет использовать посетителю адрес URL наподобие products/turntables/technics/sl1210, который будет трансформироваться в product.php?cat=turntables&<WBR>brand=technics&prod=sl1210.

Круглые скобки между слешами в регулярном выражении примера выполняют объединение в группы – мы можем использовать каждую из них как $1, $2 и $3 соответственно.

Комбинация [^/]+ в скобках соответствует любому символу, кроме слеша, в любых количествах.

.htaccess – защита от «хотлинков»

Не для кого не секрет, что сегодня все чаще воруют информацию с сайтов. Иногда только текстовую, а иногда и вместе с графическими изображениями. И каждый раз когда на сторонний сайт будет приходить посетитель эти изображения будут грузится с вашего хостинга создавая нагрузку и сжигая трафик. Чтобы это предотвратить добавляем следующий код:

```
RewriteEngine On
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !^http://([ -a-z0-9] .)?my_site.ua [NC]
RewriteRule .(gif|jpe?g|png)$ - [F,NC,L]
```

В приведенном выше примере на сайте грузящем изображение будет появляться ошибка 403, если желаете чтобы вместо картинки отображалась определенная картинка, то последнюю строку замените на следующую:

```
RewriteRule .(jpg|png|gif)$ http://my_site.ua/images/imageinfo.jpg [NC,R,L]
```

.htaccess – Блокировка пользователей пришедших с определенного сайта

Если владелец сайта не хочет, чтобы его ресурс посещали пользователи, зашедшие с определенного домена и требуется им закрыть доступ - htaccess также готов помочь. Вы можете перекрыть трафик с определенных сайтов используя страницу 403 или «запрет доступа». Полезна данная настройка тогда, когда на ваш сайт появились ссылки с сайтов с запрещенным контентом и по ним идет трафик на ваш сайт.

```
RewriteEngine on
RewriteCond %{HTTP_REFERER} zapretnui-sait.com [NC,OR]
RewriteCond %{HTTP_REFERER} zapretnui-sait.com [NC,OR]
RewriteRule .* - [F]
```

.htaccess – доп. возможности

Блокировка с определенного сайта:

Если владелец сайта не хочет, чтобы его ресурс посещали пользователи, зашедшие с определенного домена и требуется им закрыть доступ - htaccess также готов помочь. Вы можете перекрыть трафик с определенных сайтов используя страницу 403 или «запрет доступа».
Полезна данная настройка тогда, когда на ваш сайт появились ссылки с сайтов с запрещенным контентом и по ним идет трафик на ваш сайт.

```
RewriteEngine on
RewriteCond %{HTTP_REFERER} zapretnui-sait.com [NC,OR]
RewriteCond %{HTTP_REFERER} zapretnui-sait.com [NC,OR]
RewriteRule .* - [F]
```

Изменить время исполнения скрипта:

```
php_value max_execution_time 30
```

(30 - время исполнения скрипта в секундах.)

Создание своих страниц с описанием ошибок:

```
ErrorDocument 401 /errors401.html
ErrorDocument 403 /errors403.html
ErrorDocument 404 /errors404.html
ErrorDocument 500 /errors505.html
```

Напомню:

- 401 — Требуется авторизация (Authorization Required)
- 403 — пользователь не прошел аутентификацию, запрет на доступ (Forbided)
- 404 — запрашиваемый документ (файл, директория) не найден (Not Found)
- 500 — внутренняя ошибка сервера (Internal Server Error)

Определение кодировки:

```
AddDefaultCharset UTF-8
```

Ограничение размера загружаемого файла для PHP:

```
php_value upload_max_filesize 20M
```

(где 20M - это ращмер файла в мегабайтах)

Изменить максимальный размер запроса для загрузки в PHP:

```
php_value post_max_size 10M
```

Изменение времени на разбор введенных данных:

```
php_value max_input_time 60
```

Управление доступом к директориям и файлам

Запретить доступ ко всем файлам:
deny from all

Разрешить доступ с определенного IP:
order allow deny
deny from all
allow from 192.112.12.198

Запретить доступ с определенного IP:
order allow deny
deny from all
deny from 192.112.12.198

Запретить/разрешить просмотр директории без индексных файлов:
Options -Indexes (+ для разрешения)

Можно **закрыть доступ к любому из файлов**, тем не менее скрипты, если возникнет необходимость, смогут продолжить его использовать. Для этого используется следующий код:

```
<Files set.php>
deny from all
</Files>
```

Ограничить доступ к определенному типу файлов:

```
<Files "\.(htm|sql|...указать еще расширения...)$">
order allow,deny
deny from all
</Files>
```

Запрещаем просмотр нежелательным User-Agent:

```
SetEnvIfNoCase user-Agent ^FrontPage [NC,OR]
SetEnvIfNoCase user-Agent ^Java.* [NC,OR]
SetEnvIfNoCase user-Agent ^Microsoft.URL [NC,OR]
SetEnvIfNoCase user-Agent ^MSFrontPage [NC,OR]
Order Allow,Deny
Allow from all
Deny from env=bad_bot
```

Список юзерагентов - http://www.user-agents.org

Мой пример:

```
AddDefaultCharset UTF-8
RewriteEngine On
RewriteBase /
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_URI} ^(.+)/$
RewriteRule ^(.+)/$ /$1 [R=301,L]
RewriteCond %{HTTP_HOST} ^www\.(.*) [NC]        //  NC - регистронезависимость
RewriteRule ^(.*)$ http://%1/$1 [R=301,L]
RewriteCond %{THE_REQUEST} ^[A-Z]{3,9}\ /index\ HTTP/
RewriteRule ^index\.html$ / [R=301,L]
RewriteCond %{THE_REQUEST} ^[A-Z]{3,9}\ /index\.php\ HTTP/
RewriteRule ^index\.php$ / [R=301,L]
RewriteRule ^sitemap.xml$ index.php?route=feed/google_sitemap [L]
RewriteRule ^googlebase.xml$ index.php?route=feed/google_base [L]
RewriteCond %{REQUEST_FILENAME} !-f                    // не каталог
RewriteCond %{REQUEST_FILENAME} !-d                    // не файл
RewriteCond %{REQUEST_URI} !.*\.(ico|gif|jpg|jpeg|png|js|css)       // не картинка
RewriteRule ^([^?]*) index.php?_route_=$1 [L,QSA]
```

Схема простого сайта

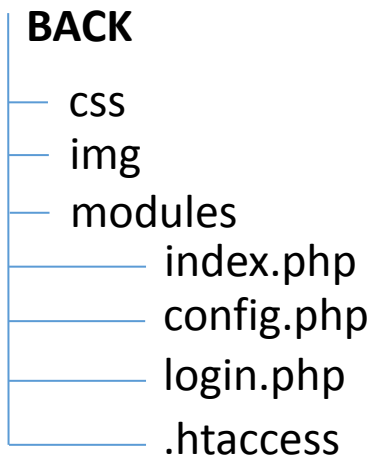
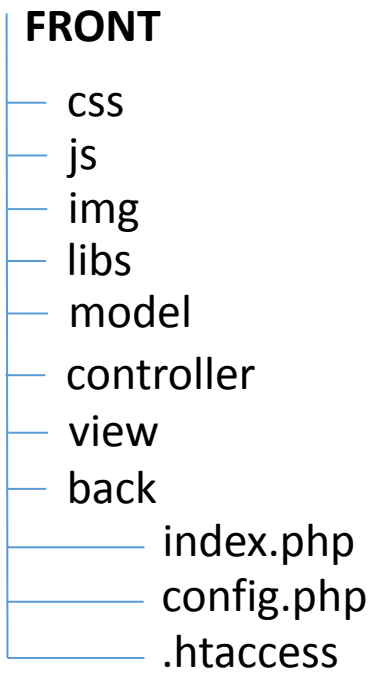
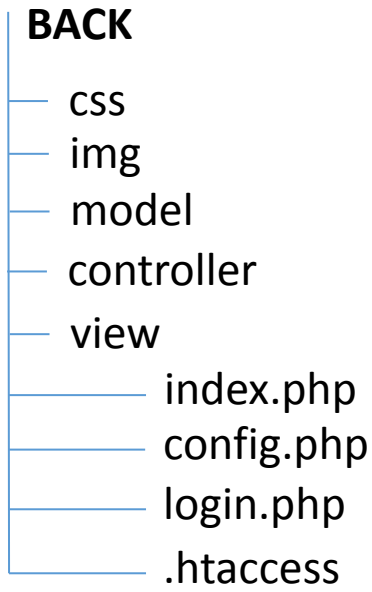


Схема сайта с MVC





Полезные ссылки:

http://www.htaccess.net.ru/doc/mod_rewrite/index.php - документация по htaccess
https://htmlweb.ru/php/htaccess.php - очень много про htaccess
http://htaccess.net.ru/doc/mod_rewrite/Primer-RewriteRule.php - получение ЧПУ