Возможности .htaccess. Синтаксис и примеры

Опубликовано muff в Чт, 2010-07-08 19:13

Итак, для начала определимся, что это за зверь такой ".htaccess" и "с чем его едят"...

.htaccess (от. англ. hypertext access) — файл дополнительной конфигурации веб-сервера Apache, а также подобных ему серверов. Позволяет задавать большое количество дополнительных параметров и разрешений для работы веб-сервера в отдельных каталогах (папках), таких как управляемый доступ к каталогам, переназначение типов файлов и т.д., без изменения главного конфигурационного файла.

.htaccess является подобием httpd.conf с той разницей, что действует только на каталог, в котором располагается, и на его дочерние каталоги. Возможность использования .htaccess в том или ином каталоге указывается в httpd.conf (директива AllowOverride).

Файл .htaccess может быть размещён в любом каталоге. Директивы этого файла действуют на все файлы в текущем каталоге и во всех его подкаталогах (если эти директивы не переопределены директивами нижележащих файлов .htaccess). Для того чтобы эти файлы .htaccess можно было использовать, необходимы соответствующие настройки главного конфигурационного файла (значение директивы AllowOverride должно быть установлено All). Как правило, подавляющее большинство хостеров разрешают использовать свои файлы .htaccess.

Синтаксис .htaccess

Вот обязательной синтаксис, несоблюдение которого приводит к ошибкам сервера:

- ***пути к файлам (директориям) указываются от корня сервера.*** Пример: /home/www/htdocs/config/.htpasswords;
- ***домены с указанием протокола.*** Пример: Redirect / http://www.site.com: [1]
- ***Файл имеет название именно "точка" htaccess***;
- ***Должен быть записан в UNIX-формате***. Для оболочки FAR, достигается F4 (редактирование файла), Shift+F2 (выбрать "сохранить как UNIX-текст").

Директивы простого перенаправления (редирект)

Наиболее часто используемые, и наиболее сложные директивы **.htaccess**. Предположим мы хотим при запросе нашего сайта переадресовать пользователя на другой URL, для этого нам необходимо в корневую директорию сайта добавить файл **.htaccess** со следующим содержимым:

```
Redirect / http://www.example.com [2]
# http://www.example.com [2] - URL На который мы перенаправляем запросы
```

Более сложный пример, мы хотим определенные страницы нашего сайта переадресовывать на другие сайты

```
Redirect /linux http://www.linux.org [3]
Redirect /linux/download.html http://www.linux.org/dist/download_info.html [4]
Redirect 301 /kernel http://www.linux.org
```

Теперь при наборе **http://mysite.ru/linux** будут открываться **http://www.linux.org**. В последнем примере WEB сервер будет передавать код 301, что означает "документ перемещен постоянно".
Синтаксис команды Redirect выглядит следующим образом:

```
Redirect [status] URL_LOCAL URL_REDIRECT

status : необязательное поле, определяет код возврата, допустимые значения:


 * permanent (301 – ???????? ????????? ?????????) * temp (302 – ???????? ????????? ?
???????) * seeother (303 – ???????? ??????) * gone (410 – ?????)

URL_LOCAL : локальная часть URL запрашиваемого документа.

URL_REDIRECT : URL куда должен быть выполнен редирект.
```

Директива **RedirectMatch** аналогична директиве **Redirect** за исключением того, что в **RedirectMatch** возможно использование регулярных выражений, что несомненно может быть удобно в некоторых условиях. Например для организации передачи параметров скрипту в теле URL.

```
RedirectMatch /(.*)/(.*)/index.html$ http://mysite.ru/script.php?par1=$1&par2=$2
```

Хотя данный пример и вызовет перезагрузку страницы в дальнейшем его можно будет улучшить. Здесь необходимо сделать небольшое лирическое отступление и поговорить о **синтаксисе регулярных выражений**.

В регулярном выражении можно использовать любые печатные символы и пробел, но часть символов имеет особое значение:

- Круглые скобки **()** используются для выделения групп символов. В дальнейшем к ним можно обращаться по номеру.
- Символ **^** обозначает начало строки.
- Символ **$** обозначает конец строки.
- Символ **.** обозначает любой символ.
- Символ **|** обозначает альтернативу. Например, выражения "A|B" означают "A или B".
- Символ **?** ставится после символа (группы), который может как присутствовать, так и отсутствовать.
- Символ * ставится после символа (группы), который может отсутствовать или присутствовать неограниченное число раз подряд.
- Символ **+** действует аналогично символу * с той лишь разницей, что предшествующий ему символ обязательно должен присутствовать хотя бы один раз.
- Квадратные скобки **[]** используются для перечисления допустимых символов.
- Квадратные скобки **[^]** используются для перечисления недоступных символов.
- Символ **** ставится перед спецсимволами, если они нужны в своем первозданном виде.
- Все, что расположено после символа **'#'**, считается комментарием.

Это все основные примитивы с помощью которых можно построить любое регулярное выражение.

Директивы сложного перенаправления (mod_rewrite)

Модуль **mod_rewrite** имеющийся в составе **Apache** — это мощнейшее, интеллектуальное средство преобразования **URL** адресов. С ним возможны почти все типы преобразований, которые могут выполняться или нет в зависимости от разных условий, факторов.

Данный модуль представляет собой основанный на правилах механизм (синтаксический анализатор с применением регулярных выражений), выполняющий **URL** преобразования на лету. Модуль поддерживает неограниченное количество правил и связанных с каждым правилом условий, реализуя действительно гибкий и мощный механизм управления **URL**. **URL** преобразования могут использовать разные источники данных, например переменные сервера, переменные окружения, **HTTP** заголовки, время и даже запросы к внешним базам данных в разных форматах, — для получения **URL** нужного вам вида.

Директива **RewriteCond** - определяет условие при котором происходит преобразование. **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.

Под **обратной связью** подразумевается использование частей сравниваемых **URL** для дальнейшего использования, т.е. как передачи параметров или для построения нового **URL**.

`$N`	(0 <= N <= 9) предоставляющие доступ к сгруппированным частям (в круглых скобках!) шаблона из соответствующей директивы RewriteRule (единственной, следующей сразу за текущим набором директив RewriteCond).
`%N`	(1 <= N <= 9) предоставляющие доступ к сгруппированным частям (в круглых скобках!) шаблона из соответствующей директивы RewriteCond в текущем наборе условий.
`%{NAME_OF_VARIABLE}`	где NAME_OF_VARIABLE может быть одной из ниже приведенных переменных

Ниже приводится список всех доступных переменных **%{NAME_OF_VARIABLE}** с их кратким описанием.

HTTP_USER_AGENT	Содержит информацию о типе и версии браузера и операционной системы посетителя.
HTTP_REFERER	Приводится адрес страницы, с которой посетитель пришёл на данную страницу.
HTTP_COOKIE	Список COOKIE передаваемых браузером
HTTP_FORWARDED	Страница непосредственно с которой перешел пользователь
HTTP_HOST	Адрес сервера, например host.ru
HTTP_ACCEPT	Описываются предпочтения клиента относительно типа документа.
REMOTE_ADDR	IP-адрес посетителя.
REMOTE_HOST	Адрес посетителя в нормальной форме — например, rt99.net.ru
REMOTE_IDENT	Имя удаленного пользователя. Имеет формат

	имя.хост, например, kondr.www.rtt99.net.ru
REMOTE_USER	То-же, что и REMOTE_IDENT, но содержит только имя. Пример: **kondr**
REQUEST_METHOD	Позволяет определить тип запроса (GET или POST). Должен обязательно анализироваться, т.к. определяет дальнейший способ обработки информации
SCRIPT_FILENAME	Полный путь к вебстранице на сервере.
PATH_INFO	Содержит в себе все, что передавалось в скрипт.
QUERY_STRING	Содержит строчку, переданную в качестве запроса при вызове **CGI** скрипта.
AUTH_TYPE	Используется для идентификации пользователя
DOCUMENT_ROOT	Содержит путь к корневой директории сервера.
SERVER_ADMIN	Почтовый адрес владельца сервера, указанный при установке.
SERVER_NAME	Адрес сервера, типа kondr.host.ru
SERVER_ADDR	IP-адрес вашего сайта.
SERVER_PORT	Порт на котором работает Apache.
SERVER_PROTOCOL	Версия **HTTP** протокола.
SERVER_SOFTWARE	Название сервера, например, **Apache/1.3.2 (Unix)**
TIME_YEAR **TIME_MON** **TIME_DAY** **TIME_HOUR** **TIME_MIN** **TIME_SEC** **TIME_WDAY** **TIME**	Переменные предназначены для работы со временем в разных форматах.
API_VERSION	Это версия **API** модуля **Apache** (внутренний интерфейс между сервером и модулем) в текущей сборке сервера, что определено в **include/ap_mmn.h**.
THE_REQUEST	Полная строка **HTTP** запроса отправленная браузером серверу (т.е., **«GET /index.html HTTP/1.1»**). Она не включает какие-либо дополнительные заголовки отправляемые браузером.
REQUEST_URI	Ресурс, запрошенный в строке HTTP запроса.
REQUEST_FILENAME	Полный путь в файловой системе сервера к файлу или скрипту соответствующим этому запросу.
IS_SUBREQ	Будет содержать текст «true» если запрос выполняется в текущий момент как подзапрос, «false» в другом случае. Подзапросы могут быть сгенерированы модулями которым нужно иметь дело с дополнительными файлами или URI для того чтобы выполнить собственные задачи.

Условие это шаблон условия, т.е., какое-либо регулярное выражение применяемое к текущему экземпляру "Сравниваемая Строка", т.е., "Сравниваемая Строка" просматривается на поиск соответствия Условие.

Помните: Условие это perl совместимое регулярное выражение с некоторыми дополнениями:

- Вы можете предварять строку шаблона префиксом **'!'** 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)

Все эти проверки также могут быть предварены префиксом восклицательный знак ('!') для инвертирования их значения.

RewriteEngine включает или выключает работу механизма преобразования. Если она установлена в положение **off** этот модуль совсем не работает. Отметьте, что по-умолчанию, настройки преобразований не наследуются. Это означает что вы должны иметь RewriteEngine on директиву для каждого виртуального хоста в котором вы хотите использовать этот модуль.
Синтаксис **RewriteEngine** выглядит следующим образом:

```
RewriteEngine on | off

# По умолчанию RewriteEngine off
```

Используйте для комбинирования условий в правилах **OR** вместо **AND**. Типичный пример - перенавравление запросов на поддомены в отдельные каталоги.

```
RewriteEngine on

RewriteCond %{REMOTE_HOST} ^mysubdomain1.* [OR]
RewriteCond %{REMOTE_HOST} ^mysubdomain2.* [OR]
RewriteCond %{REMOTE_HOST} ^mysubdomain3.*
RewriteRule ^(.*)$ ^mysubdomain_public_html/$1

RewriteCond %{REMOTE_HOST} ^mysubdomain4.*
RewriteRule ^(.*)$ ^mysubdomain4_public_html/$1
```

Для выдачи главной страницы какого-либо сайта согласно **«User-Agent:»** заголовку запроса, вы можете использовать следующие директивы:

```
RewriteEngine on

RewriteCond %{HTTP_USER_AGENT} ^Mozilla.*
RewriteRule ^/$ /homepage.max.html [L]

RewriteCond %{HTTP_USER_AGENT} ^Lynx.*
RewriteRule ^/$ /homepage.min.html [L]
```

```
RewriteRule ^/$ /homepage.std.html [L]
```

Для выдачи разных сайтов для разных браузеров согласно **«User-Agent:»** заголовку запроса, вы можете использовать следующие директивы:

```
RewriteEngine on

RewriteCond %{HTTP_USER_AGENT} ^Mozilla.*
RewriteRule ^(.*)$ /mozilla/$1 [L]

RewriteCond %{HTTP_USER_AGENT} ^Lynx.*
RewriteRule ^(.*)$ /lynx/$1 [L]

RewriteRule ^(.*)$ /default/$1 [L]
```

Общий синтаксис директивы **RewriteRule** выглядит следующим образом:

```
RewriteRule Шаблон Подстановка [flag]

# flag - необязательное поле указывающее дополнительные опции
```

В подстановке вы можете использовать, в том числе, и специальные флаги путем добавления в качестве третьего аргумента директивы **RewriteRule**. Флаги — это разделённый запятыми, следующий список флагов:

'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 никакого кода в подстановке ответ будет с **HTTP** статусом 302 (ВРЕМЕННО ПЕРЕМЕЩЕН). Для остановки процесса преобразования, вам также нужно написать флаг **'L'**.

'forbidden|F [=code]'
(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 [=code]'
(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.

'proxy|P [=code]'
(вызывает прокси)
Этот флаг помечает подстановочную часть как внутренний запрос прокси и немедленно (т.е., процесс преобразования здесь останавливается) пропускает его через прокси модуль. Используйте этот флаг для того, чтобы добиться более мощной реализации диркетивы **ProxyPass**, интегрирующей некоторое содержимое на удаленных серверах, в пространство имён локального сервера.

'last|L [=code]'
(последнее правило)
Остановить процесс преобразования на этом месте и не применять больше никаких правил преобразований. Используйте этот флаг для того, чтобы не преобразовывать текущий **URL** другими, следующими за этим, правилами преобразований.

'next|N [=code]'
(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.Используйте этот флаг для перезапуска процесса преобразований, т.е., безусловному переходу на начало цикла.

'chain|C [=code]'
(связь со следующим правилом)
Этот флаг связывает текущее правило со следующим (которое, в свою очередь, может быть связано со следующим за ним, и т.д.). 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 [=code]'
(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»**.

'nosubreq|NS [=code]'
(используется только в случае не внутреннего подзапроса)
Этот флаг дает команду механизму преобразований пропустить директиву если текущий подзапрос является внутренним подзапросом. К примеру, внутренние подзапросы в **Apache** происходят тогда, когда **mod_include** пытается получить информацию о возможных файлах по-умолчанию для каталогов (index.xxx). При подзапросах это не всегда полезно и даже иногда вызывает проблему в работе набора директив преобразований. Используйте этот флаг для исключения некоторых правил.

'nocase|NC [=code]'
(не учитывать регистр)
Это делает Шаблон нечуствительным к регистру, т.е., нет различий между 'A-Z' и 'a-z' когда Шаблон применяется к текущему **URL**.

'qsappend|QSA [=code]'
(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 [=code]'
(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.

Если в подкаталогах в **.htaccess** нет ни одной директивы модуля **mod_rewrite**, то все правила преобразования наследуются из родительского каталога.

При наличии в файле **.htaccess** каких либо директив модуля **mod_rewrite** не наследуется

ничего, а состояние по умолчанию выставляется таким же, как в главном конфигурационном файле веб-сервера (по умолчанию "off"). Поэтому, если нужны правила преобразования для конкретного каталога, то нужно еще раз вставить директиву **"RewriteEngine on"** в **.htaccess** для конкретного каталога.

При наследовании правил из верхних каталогов и добавлении к ним новых свойственных только данному каталогу - необходимо выставить в начале следующее: **"RewriteEngine on"** и **"RewriteOptions inherit"** - последняя директива сообщает серверу о продолжении.

Примеры

Посетители веб-сайта авторизуются при помощи стандартной авторизации (**AuthType BasicAuth**). Необходимо по ссылке **/home/первая буква логина/** показывать содержимое их домашних каталогов.

```
RewriteEngine on
RewriteCond %{REMOTE _ USER} != ""
RewriteCond /home/(.)/(%{REMOTE_USER}) -d
RewriteRule (.*) /home/%2/$1
```

Жесткий запрет посещений нашего веб-сайта для робота поисковой системы Google

```
RewriteEngine on

RewriteCond %{USER_AGENT} Googlebot
RewriteRule .* - [F]

# Другой вариант возвращает вместо ошибки 403 ( FORBIDDEN ) ошибку 404 ( NOT _ FOUND )
RewriteCond %{USER_AGENT} Googlebot
RewriteRule .* - [R=404]
```

Закрыть доступ к веб-сайту в рабочее время

```
RewriteEngine on

RewriteCond %{TIME_HOUR}%{TIME_MIN} > 900
RewriteCond %{TIME_HOUR}%{TIME_MIN} < 1800
RewriteRule .* - [ F ]
```

Если на вашем сайте есть очень ценные картинки или архивы и вы не хотите чтобы кто-то размещал их (если архивы, то ссылки на них) на своих страницах, создавая таким образом бесполезный трафик для вашего сайта, вы можете запретить скачивание ресурсов, проверяя поле заголовка **HTTP_REFERER**. для каталога:

```
RewriteEngine on

RewriteBase /img/
RewriteCond %{HTTP_REFERER} !^$
RewriteRule .* - [ F ]
```

и для определенных типов файлов

```
RewriteEngine on

RewriteBase /img/
RewriteCond %{HTTP_REFERER} !^$
RewriteRule \.(jpe?g|gif|png|css|swf)$ - [ F ]
```

В связи с неоднозначностью записи расширения HTML файлов (.htm или .html), некоторые пользователи могут ошибочно набрать адрес страницы. Для автоматического исправления такого рода ошибок, можно воспользоваться mod_rewrite.

```
RewriteEngine on

RewriteBase /
RewriteRule ^(.*)\.htm$ $1.html [R=permanent]
```

Необходимо запрос любой страницы сайта отправлять на одну (будет написано что сайт временно ен доступен), но в то же время нужно оставить его открытым для поисковых машин. То есть для клиентов сайт закрыт, а для индексации - открыт.

```
RewriteEngine on

RewriteBase /
RewriteCond %{HTTP_USER_AGENT} !^yandex.* [NC,OR]
RewriteCond %{HTTP_USER_AGENT} !^googlebot.* [NC,OR]
RewriteCond %{HTTP_USER_AGENT} !^gaisbot.* [NC,OR]
RewriteCond %{HTTP_USER_AGENT} !^rambler.* [NC,OR]
RewriteCond %{HTTP_USER_AGENT} !^aport.* [NC,OR]
RewriteCond %{HTTP_USER_AGENT} !^metacrawler.* [NC,OR]
RewriteCond %{HTTP_USER_AGENT} !^msnbot.* [NC,OR]
RewriteCond %{HTTP_USER_AGENT} !^crawler.* [NC,OR]
RewriteRule ^/$ - [L,R]
```

Перенаправление несуществующих **URL** на другой веб-сервер

```
RewriteEngine on

RewriteBase /
RewriteCond /your/docroot/%{REQUEST_FILENAME} !-f
RewriteRule ^(.+) http://webserverB.dom/$1
```

Проблема здесь в том, что это будет работать только для страниц находяшихся внутри **DocumentRoot**. Тогда как вы можете добавить больше условий (например ещё и для управления домашними каталогами, и т.д.) есть лучший вариант:

```
RewriteEngine on

RewriteBase /
RewriteCond /your/docroot/%{REQUEST_FILENAME} !-U
```

```
RewriteRule ^(.+) http://webserverB.dom/$1
```

Сайт перенесен с одного домена на другой с domain.com на domain2.com

```
RewriteEngine on

RewriteCond %{HTTP_HOST} ^www\.domain\.com$ [R=301,1]
RewriteRule ^(.*)$ http://www.domain2.com/$1
```

Часто хостинговые провайдеры накладывают ограничение на количество уикальных сайтов, и зачастую под это ограничение попадают и поддомены. Данное ограничение можно обойти средствами **.htaccess**. Например мы хотим направить поддомен forum.yourdomain.net на директорию **~/forum** для этого необходимо направить нужные поддомены на папку с остновным сайтом и создать вложенную директорию например **forum**

```
RewriteEngine on

RewriteCond %{HTTP_HOST} ^forum\.yuordomain\.net$ [NC]
RewriteCond %{REQUEST_URI} !^/forum/$ [NC]
RewriteRule (.*) /forum/$1 [L]
```

Индексные страницы

Когда пользователь заходит на хост например **http://gentoo.org** принято, что открывается индексный файл index.* при его отсутствии - либо содержимое каталога, либо ошибку 403 (FORBIDDEN) если опция отключена - запрещен просмотр директорий.

За листинг файлов отвечает директива **Indexes** (показывать посетителю список файлов, если в выбранном каталоге нет файла **index.html** или его аналога).

Иногда нужно сделать так, чтобы в случае отсутствия в каталоге файла, который показывается по умолчанию, листинг, то есть список файлов в каталоге, не выдавался. В этом случае добавим в **.htaccess** такую строчку:

```
# Запрет выдачи листинга пустого каталога
Options -Indexes
```

А чтобы выдавал листинг, нужно:

```
Options Indexes
```

Если же понадобиться разрешить просматривать список файлов, но чтобы при этом часть файлов определенного формата не отображалась, то запишем:

```
IndexIgnore *.php* *.pl
```

Выдает листинг каталога, т.е. его содержание со всем содержанием, за исключением файлов-скриптов **PHP** и **Perl**.

Если ваш веб-сайт построен на скриптах, то в качестве индексных часто могут использоваться файлы с другими расширениями - указать эти файлы можно с помощью директивы **DirectoryIndex** .

```
DirectoryIndex index.html index.shtml index.pl index.cgi index.php
```

Если же вы хотите что бы при обращении к каталогу открывался не **index.html**, а например, файл **htaccess.php** или **/cgi-bin/index.pl**:

```
DirectoryIndex htaccess.php /cgi-bin/index.pl
```

Обработка ошибок

В ходе работы сервера иногда возникают ошибки, скорее это будет правильней назвать не сбоями в работе сервера, а стандартными кодами возврата оговоренными в стандарте HTTP_RFC2616. Вообще, в RFC ошибки называются "Status Codes", но мы их будем называть именно ошибками - так привычнее.

Код возврата - это трехзначное число, на основании которого можно судить о том, насколько успешно был обработан запрос. Код возврата начинающиеся на 1,2,3 считаются успешными, остальные причисляются к разряду ошибок.

```
Вот список ошибок 4xx и 5xx :
```

400 - Bad Request
401 - Unauthorized
402 - Payment Required
403 - Forbidden
404 - Not Found
405 - Method Not Allowed
406 - Not Acceptable
407 - Proxy Authentication Required
408 - Request Time-out
409 - Conflict
410 - Gone
411 - Length Required
412 - Precondition Failed
413 - Request Entity Too Large
414 - Request-URI Too Large
415 - Unsupported Media Type
500 - Internal Server Error
501 - Not Implemented
502 - Bad Gateway
503 - Service Unavailable
504 - Gateway Time-out
505 - HTTP Version not supported

При возникновении ошибки 4xx или 5xx посетитель Вашего сайта увидит в браузере

сообщение от сервера, которое врядlли можно назвать предельно понятным рядовому пользователю. **Apache** предоставляет возможность выдать вместо аскетичного технического текста, не изобилующего деталями, свою страницу, где Вы можете человеческим языком объяснить пользователю, что произошло и что делать.

Пример переопределения страниц ошибок приведен ниже:

```
# содержание файла .htaccess:
ErrorDocument 404 http://bg10.ru/error/404.htm [5]
ErrorDocument 403 http://bg10.ru/error/403.htm [6]
ErrorDocument 400 http://bg10.ru/error/400.htm [7]
ErrorDocument 500 http://bg10.ru/error/500.htm [8]

# в случае ошибки "FORBIDDEN" показывается текстовое сообщение, которое
# обязательно должно начинаться с кавычки, кавычка в сообщении не выводится:

ErrorDocument 403 "Sorry can't allow you access today, 403 Status Codes Apache"
```

Кодировка

Иногда браузер пользователя не может корректно определить тип кодировки выдаваемого документа. Для решения этой проблемы используемая кодировка указывается в настройках **Web** сервера **Apache** и заголовке передаваемого документа. Причем для корректного распознания эти кодировки должны совпадать. На наших серверах по умолчанию используется кодировка **cp1251**

В HTML для указания кодировки используется тег:

```
<meta http-equiv="content-type" content="text/html; charset=Windows-1251">
```

Наиболее часто встречаются типы кодировки для русского языка передаваемые в заголовке документа:

- Windows-1251 - Кириллица (Windows).
- KOI8-r - Кириллица (КОИ8-Р)
- cp866 - Кириллица (DOS).
- Windows-1252 - Западная Европа (Windows).
- Windows-1250 - Центральная Европа (Windows).
- UTF-8 - двух байтовая кодировка

Теперь рассмотрим указание кодировки по умолчанию через **.htaccess**. **AddDefaultCharset** задает дефолтную таблицу символов (кодировку) для всех выдаваемых страниц на веб сервере **Apache**. Указываем кодировку на все файлы, в которой по умолчанию получает документы браузер:

```
AddDefaultCharset WINDOWS-1251
```

При загрузке файла на сервер, возможна перекодировка, его - указываем, что все получаемые файлы будут иметь кодировку windows-1251, для того что бы указать кодировку на загружаемые файлы напишем:

```
CharsetSourceEnc WINDOWS-1251
```

Если необходимо отменить перекодировку сервером файлов:

```
CharsetDisable on
```

Управление доступом

Очень часто возникает необходимость запретить доступ к определенным файлам или папкам для определенных груп пользователей. В **Web** сервере **Apache** есть встроенные средства для решения данной проблемы.

Для запрета или разрешения доступа ко всем файлам и папкам в текущей и во всех вложенных директориях используется директива **Order** синтаксис ее очень прост:

```
Order [Deny,Allow] | [Allow,Deny]
# По умолчанию Deny,Allow
```

В зависимости от того в каком порядке указаны директивы меняется логика работы сервера. В случае если Deny,Allow то запрещается доступ со всех IP кроме оговоренных, в случае если Allow,Deny разрешается доступ со всех IP кроме оговоренных. Далее должны идти секции описания для доступа и запрета. Ключевое слово **all** означает со всех **IP**

Например мы хотим запретить (блокировать) доступ с **IP** 81.222.144.12 и 81.222.144.20 и разрешить всем остальным нам необходимо добавить в **.htaccess** следующий код:

```
Order Allow,Deny
Allow from all
Deny from 81.222.144.12, 81.222.144.20
```

Для обратной ситуации когда мы хотим запретить доступ со всех IP кроме 81.222.144.12 и 81.222.144.20 нам необходимо добавить в .htaccess следующий код:

```
Order Deny,Allow
Deny from all
Allow from 81.222.144.12, 81.222.144.20
```

Запрет или разрешение на доступ можно указывать не только на все файлы, но так же можно указывать на отдельный файл или группы файлов. Например мы хотим запретить доступ всех пользователей кроме **IP** 81.222.144.12 к файлу **passwd.html** который расположен в текущей директории.

```
<Files "passwd.html">
Order Deny,Allow
Deny from all
Allow from 81.222.144.12
</Files>
```

Так же можно запретить или разрешить доступ к определенной группе файлов. Например к файлам с расширением **".key"**:

```
<Files "\.(key)$">
Order Deny,Allow
Deny from all
Allow from 81.222.144.12
</Files>
```

Паролирование директорий

.htaccess можно так же использовать для установки пароля на доступ к определенным папкам, файлам и группам файлов. Приведем рабочий пример, а потом поясним все содержимое:

```
AuthName "Protected area, need authorization"
AuthType Basic
AuthUserFile /home/t/test/.authfile
require valid-user
```

Данный файл нужно положить в ту директории, на которую мы хотим поставить пароль.

Директива AuthName выводит сообщение при запросе пароля, все сообщение необходимо писать в одну строчку, синтаксис директивы тривиален:

```
AuthName "SEE TEXT"
```

Директива **AuthType** выбирает тип аутентификации. Возможны следующие типы: **Basic** или **Digest**. Второй может не поддерживаться некоторыми браузерами, поэтому пользоваться им не рекомендуется.

```
AuthType Basic | Digest
```

AuthUserFile указывает имя файла с паролями для аутентификации пользователей (пароли в этом файле будут шифрованными). Путь к файлу с паролями задается относительно корня веб-сервера. Храните файл с паролями в папке, доступ к которой закрыт для пользователей - (желательно поместить этот файл вне иерархии вашего веб-сайта).

Запустив **htpasswd** без параметров мы увидим:

```
# htpasswd

Usage: htpasswd [-cmdpsD] passwordfile username htpasswd -b[cmdpsD] passwordfile use
rname password

 htpasswd -n[mdps] username htpasswd -nb[mdps] username password-c Create a new file
.-n Don't update file; display results on stdout.-m Force MD5 encryption of the pass
word.-d Force CRYPT encryption of the password (default).-p Do not encrypt the passw
ord (plaintext).-s Force SHA encryption of the password.-b Use the password from the
 command line rather than prompting for it.-D Delete the specified user.On Windows,
```

```
NetWare and TPF systems the '-m' flag is used by default.On all other systems, the '
-p' flag will probably not work.
```

Здесь не будут рассматриваться все параметры этой команды, но вы можете сами прочитать подробности, запустив **htpasswd** в **unix shell** или ознакомившись с соответствующей страницей документации по **Apache**.

Итак, изначально у нас еще нет файла с паролями и нам нужно его создать:

```
# htpasswd -c authfile test1
New password:
Re-type new password
Adding password for user test1
```

Здесь не будут рассматриваться все параметры этой команды, но вы можете сами прочитать подробности, запустив **htpasswd** в **unix shell** или ознакомившись с соответствующей страницей документации по Apache.
После выполнения данной операции **htpasswd** создаст файл **passwords**, в котором окажется пользователь **test1** и его пароль в зашифрованном виде:

```
# cat .authfile
test1:zgco1KREjBY8M
```

А теперь мы хотим добавить еще одного пользователя. Так как файл с паролями у нас уже есть, мы просто не будем использовать ключ **'-c'** :

```
# htpasswd .authfile test2
New password:
Re-type new password:
Adding password for user test2

# cat .authfile
test1:zgco1KREjBY8M
test2:eN3uA6t0kzV1c
```

Вернемся к описанию директив паролирования директорий. Директива **Require** определяет пользователей, которые могут получить доступ:

```
Require USER_NAME | valid-user
```

Указывая valid-user вы разрешаете доступ всем пользователям, перечисленным в файле паролей.

Приведем пример для доступа определенных пользователей из файла с паролями **.htpasswd**

```
AuthName "Protected area, need authorization"
AuthType Basic
AuthUserFile /home/t/test/.authfile
require muff alex
```

Так же как и с запретом доступа по IP здесь можно использовать расширение **<Files>** ниже приведены два примера: установки пароля на группу файлов и на один определенный файл.

```
<Files "passwd.html">
AuthName "Protected area, need authorization"
AuthType Basic
AuthUserFile /home/t/test/.authfile
require valid-user
</Files>
```

```
<Files "\.(key)$">
AuthName "Protected area, need authorization"
AuthType Basic
AuthUserFile /home/t/test/.authfile
require valid-user
</Files>
```

Следует помнить, что при таком ограничении доступа пароли передаются по каналам связи в открытом виде и при определенных обстоятельствах могут быть перехвачены злоумышленниками. Поэтому в целях безопасности рекомендуется организовывать доступ к закрытым областям веб-сайта через защищенное **SSL-соединение**.

Указание опций PHP

Директивы для конфигурирования **PHP** можно размещать не только в файле **php.ini**, но также и в конфигурационных файлах Apache для вашего сайта – **.htaccess**. Это позволяет проводить тонкую настройку **php** для разных директорий.

Для работы с **PHP** в конфигурационных файлах **Apache** доступны 4 директивы: **php_value**, **php_flag**, **php_admin_value**, **php_admin_flag**, которые отличаются значимостью, типом устанавливаемых значений и местом применения.

Директивы **php_admin_value**, **php_admin_flag** выставляются только в файле **httpd.conf**, так что нам они не интересны. По сути данные директивы переопределяют значение остальных директив.

Директивыа **php_flag** служит для установки логических значений директив в **php.ini**. В то время как директива **php_value** служит для установки строковых и числовых значений директив **php.ini**, т.е. любых типов значений, за исключением логических.

Синтаксис директив очень прост:

```
php_flag имя директивы on | off
php_value имя директивы VALUE
```

Приведем перечень наиболее часто используемых директив

mysql.default_host	Устанавливает имя хоста базы данных. **Пример**: php_value mysql.default_host localhost
mysql.default_user	Устанавливает имя пользователя базы данных **Пример**: php_value mysql.default_user alexey
mysql.default_password	Устанавливает пароль пользователя базы

	данных **Пример**: php_value mysql.default_password Hry5Gw2
display_errors	Разрешает вывод ошибок и предупреждений в браузер. **Пример**: php_flag display_errors 0
display_startup_errors	Включает отображение ошибок, возникающих при запуске PHP. **Пример**: php_flag display_startup_errors 0
error_reporting	Определяет типы (уровени важности) фиксируемых ошибок. **Пример**: php_value error_reporting “E_ALL & ~E_NOTICE”
auto_prepend_file	Определение файла, который будет выводится в начале каждого php-скрипта. Путь указывается от корня файловой системы сервера. **Пример**: php_value auto_prepend_file /www/server/prepend.php
auto_append_file	Определение файла, который будет выводится в конце каждого php-скрипта. **Пример**: php_value auto_append_file /www/server/append.php
sendmail_from	Устанавливает e-mail отправителя, который применяется при отправке почтовых сообщений с помощью PHP. **Пример**: php_value sendmail_from root [at] beget [dot] ru
user_agent	Устанавливает строку User-agent, которая используется PHP при обращении к удаленным серверам. **Пример**: php_value user_agent “Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)”

Например для вывода всех сообщений об ошибках генерируемых **php** в **.htaccess** нужно прописать следующие строки:

```
php_flag display_errors 1
php_flag display_startup_errors 1
php_value error_reporting “E_ALL & ~E_NOTICE”
```

Для запрещения выполнения **php** в текущей директории и во всех вложенных необходиомо в **.htaccess** прописать следующие строки:

```
php_flag engine off
```

Использовано материалы:

- Волшебный файл .htaccess [9]

Источник (получено *2026-10-30 17:53*):

http://muff.kiev.ua/content/vozmozhnosti-htaccess-sintaksis-i-primery

Ссылки:
[1] http://www.site.com/;
[2] http://www.example.com
[3] http://www.linux.org
[4] http://www.linux.org/dist/download_info.html
[5] http://bg10.ru/error/404.htm
[6] http://bg10.ru/error/403.htm
[7] http://bg10.ru/error/400.htm
[8] http://bg10.ru/error/500.htm
[9] http://beget.ru/art9.html#simple_redirect