FOT.: Oleksiy Mark/Svietlanan/Dreamstime.com

Klucz do serwera

Dzięki plikom .htaccess możemy w ogromnym zakresie modyfikować działanie serwera Apache. To bardzo przydatne, szczególnie gdy firma hostingowa nie zapewnia nam bezpośredniego dostępu do konfiguracji Apache'a. Jak korzystać z .htaccess?

.htaccess to podstawa
Michał Kobiela
Autor

Wartość plików .htaccess doceniłem, uruchamiając własną witrynę internetową. Bez nich (i modułu mod_rewrite) nie byłbym w stanie tworzyć przyjaznych odnośników i tym samym zwiększyć pozycji strony w wynikach wyszukiwarek. Zmniejszyłem także znacznie zużycie transferu, modyfikując nagłówki ważności plików. Zawsze też zastanawiałem się, jak działają dynamicznie generowane obrazy, które nieraz miałem okazję oglądać w podpisach użytkowników – teraz wiem, że to wszystko zasługa .htaccess.

Według danych statystycznych za lipiec 2011 roku blisko 66 procent stron internetowych udostępnianych jest w sieci za pomocą serwera Apache. Udział Apache'a w rynku ciągle rośnie i nic nie wskazuje, aby w przyszłości się to zmieniło. Warto więc zapoznać się z tym popularnym serwerem, a przede wszystkim z jego możliwościami konfiguracyjnymi. Większość opcji znajduje się w pliku **httpd.conf**, w głównym katalogu Apache. Jednak w przypadku wykupionej usługi hostingowej zazwyczaj nie mamy dostępu do powłoki systemowej, a tym bardziej do konta administratora, tym samym nie mamy większego wpływu na to, jak działa nasz serwer. Rozwiązaniem tej kwestii jest edycja plików **.htaccess**. Możemy w nich zapisać dodatkowe ustawienia, pozwalające na dostosowanie funkcjonowania serwera do naszych indywidualnych potrzeb.

Przydatny .htaccess

Polecenia zawarte w głównym pliku konfiguracyjnym serwera są ładowane tylko podczas uruchamiania Apache'a. Natomiast pliki .htaccess są wyszukiwane i ładowane za każdym razem, gdy otwieramy dowolny folder spośród katalogów, do których dostęp ma serwer. Zawarte w nich dyrektywy wprowadzają zmiany także we wszystkich podkatalogach względem lokalizacji .htaccess.

Potrzebne oprogramowanie

W artykule będziemy korzystali z gotowego pakietu XAMPP (serwer Apache, baza MySQL, interpreter języka PHP), który należy zainstalować z dołączonej płyty DVD KOD 102 lub pobrać ze strony www.apachefriends.org/en/xampp.html. Należy mieć na uwadze, że wszystkie poniższe przykłady zawartości plików **.htaccess** działają tylko i wyłącznie na serwerze Apache, ponieważ jedynie on je obsługuje. Administrator ma jednak możliwość całkowitego wyłączenia ich wczytywania. Jedyne, co wówczas możemy zrobić, to napisać prośbę o włączenie ich obsługi.

W ARTYKULE

NA DVD

Kompletne kody źródłowe 001
plik znajduje się na krążku Eksperta
plik w całości znajduje się na krążku, pokazany tu jest tylko jego fragment
Drupal 7.7 GPL 022
Notepad++ 5.9 GPL 060
WordPress 3.2.1 PL GPL 099
WordPress Exec-PHP 4.9 FREEWARE 100
XAMPP 1.7.4 GPL 102

Aby szybko uzyskać dostęp do programu, w interfejsie płyty wpisujemy KOD i wciskamy enter.

Co warto wiedzieć o .htaccess

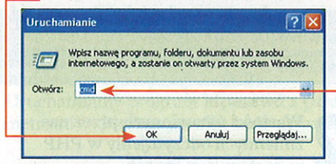
Generowanie przyjaznych adresów URL, własnych stron błędów, blokowanie dostępu do ważnych katalogów i plików, zabezpieczanie ich hasłem, a także zmienianie ustawień interpretera PHP – to tylko niektóre korzyści, jakie przynosi stosowanie plików **.htaccess**. Najbardziej docenią je użytkownicy wielu popularnych CMS-ów, takich jak WordPress, Drupal czy też Joomla, gdyż domyślnie korzystają one z wszystkich możliwości, jakie niosą pliki .htaccess.

Tworzenie pliku .htaccess

W przeciwieństwie do Linuksa, w systemie Windows można mieć problem z utworzeniem pliku .htaccess. Niedogodność tę omijamy, postępując według poniższych kroków.

1 Tworzymy na dysku C:\ nowy plik tekstowy plik.txt.

2 Jeżeli korzystamy z Windows 7, klikamy na przycisk , w pole wyszukiwania wpisujemy , a następnie klikamy na cmd.exe. W przypadku starszej wersji systemu Windows klikamy na Start, z menu wybieramy Uruchom... . W oknie, które się pojawi, wpisujemy i klikamy na przycisk .



3 W Wierszu polecenia komendą `cd C:\` przechodzimy na dysk twardy C. Zmieniamy nazwę pliku, w tym wypadku plik **.txt**, na **.htaccess**, wydając polecenie `ren plik.txt .htaccess`. Nie należy zapomnieć, aby przenieść go później do odpowiedniego katalogu.

Włączanie ładowania modułów

Niektore funkcje, które oferują pliki **.htaccess**, wymagają aktywacji odpowiedniego modułu. Zazwyczaj potrzebne opcje są aktywne. Jeżeli tak nie jest, możemy włączyć je ręcznie. W tym celu przechodzimy do katalogu C:\xampp\apache\conf i dwukrotnym kliknięciem otwieramy w edytorze tekstu plik httpd.conf. Szukamy w nim następujących linii .

```
121  #LoadModule proxy_http_module modules/mod_proxy_http.so
122  LoadModule rewrite_module modules/mod_rewrite.so
123  LoadModule setenvif_module modules/mod_setenvif.so
```

Moduł włączamy, usuwając znak komentarza (#) znajdujący się obok jego ścieżki – w tym wypadku interesuje nas **mod_rewrite** , który został opisany w poradzie umieszczonej niżej po lewej. Na koniec nie możemy zapomnieć o restarcie serwera, w przeciwnym wypadku zmiany nie zostaną aktywowane.

Własne strony błędów

1 Jedną z ciekawszych zalet plików **.htaccess** jest możliwość użycia samodzielnie zaprojektowanych stron błędów, na przykład 404 .

2 Zadanie to nie należy do skomplikowanych i ogranicza się jedynie do umieszczenia w pliku .htaccess następujących dyrektyw .

```
ErrorDocument 404 /blad404.php  1
ErrorDocument 500 /blad500.php  2
ErrorDocument 403 /zabronione.jpg  3
```

W naszym przypadku, gdy serwer zwróci błąd 404, zostanie załadowany plik 1, gdy będzie to błąd 500 2, a gdy 403 – 3.

3 Aby dodać następne spersonalizowane strony, po prostu dodajemy kolejne wiersze, zamieniając numer obok ErrorDocument na interesujący nas kod błędu (możemy je wszystkie znaleźć na stronie http://pl.wikipedia.org/wiki/Kod_odpowiedzi_HTTP) i odpowiednią nazwę pliku, który ma się wówczas wyświetlić.

Moduł mod_rewrite

1 Najczęściej używanym i uniwersalnym rozszerzeniem serwera Apache jest moduł **mod_rewrite**. Jak sama jego nazwa wskazuje, pozwala on na przepisywanie adresów według określonych reguł. Jeżeli na przykład adresy wyglądają w następujący sposób: http://localhost/artykul.php?id=ekspert_nr1, to możemy zamienić je na postać bardziej przyjazną – http://localhost/artykuly/ekspert_nr1. Będzie ona dużo bardziej czytelna zarówno dla użytkowników naszego serwisu, jak i wyszukiwarek internetowych, co przełoży się także na jej lepsze pozycjonowanie.

2 Podany obok kod zamienia adresy w sposób, jaki został zaprezentowany, w poprzednim punkcie. Jak widać, jest tutaj wymagana znajomość wyrażeń regularnych (KŚ Ekspert numer 3/2004, str. 46). Poleceniem 1 aktywujemy uruchomienie mechanizmu służącego do przepisywania. Początek ciągu znaków, od którego rozpoczniemy dopasowywanie, oznaczamy `^` . Zapis `[^/]` oznacza pojedynczy znak różny od ukośnika. Nawiasy `( )` otaczające wyrażenie powodują zapamiętanie wartości, która się między nimi pojawiła, i przypisanie jej do zmiennej, znak `+` mówi, że może się ono kilkakrotnie powtórzyć, a znak zapytania `?` dodaje warunek „lub". W tym wypadku drugi warunek jest pusty (nie został określony), co oznacza po prostu brak `/` na końcu ciągu. Znak dolara `$` kończy dopasowywanie ciągu. Dalej podajemy adres, na jaki będziemy „przepisywali" wpisywany przez użytkownika URL. Zapis **$1** 2 oznacza wstawienie wcześniej zapamiętanego ciągu znaków, a `[L]` daje sygnał do zakończenia procedury przepisywania.

```
RewriteEngine On  1
RewriteRule ^artykuly/([^/]+)/?$ artykul.
php?id=$1 [L]
```

2

3 Przy wykorzystaniu modułu **mod_rewrite** możliwe jest także wymuszenie używania lub też nie

przedrostka www. Dzięki zastosowaniu tego rozwiązania eliminowany jest problem wyszukiwania zdublowanych treści przez wyszukiwarki internetowe, co spowodowane jest indeksowaniem treści strony twojastrona.pl i www.twojastrona.pl. W przykładzie zezwolono tylko na używanie przedrostka www, w przeciwnym wypadku robotowi zostanie zwrócony błąd 301 – strona przeniesiona permanentnie. Zapis 2 oznacza, że warunek 3 jest niewrażliwy na to, czy adres wpisany był z dużych czy małych liter, a 4 odczytuje adres, jaki został zażądany (zmienne w przypadku **RewriteCond** poprzedzamy znakiem %, a w **RewriteRule** ^).

```
RewriteEngine On
RewriteCond %{HTTP_HOST} ^twojastrona.pl [NC]   2
RewriteRule ^(.*)$ http://www.twojastrona.pl/$1
[R=301,L]   1
```

```
RewriteEngine On
RewriteCond %{HTTP_USER_AGENT} ^GoogleBot [OR]   1
RewriteCond %{HTTP_USER_AGENT} ^HTTrack   2
RewriteRule ^(.*)$ - [F,L]
```

4 Gdy chcemy pozbyć się nieproszonych gości w postaci niektórych pająków sieciowych zbierających informacje o naszej stronie internetowej, wystarczy zastosować kod. Jak widać na poniższym przykładzie, pokusiliśmy się o zablokowanie robota sieciowego Google 1, a także znanego programu do tworzenia luster (tak zwanych mirrorów) stron internetowych HTTrack 2. Zapis oznacza warunek „lub", a wysyła informację zwrotną o zabronionym dostępie i kończy przepisywanie.

Zmniejszanie obciążenia serwera

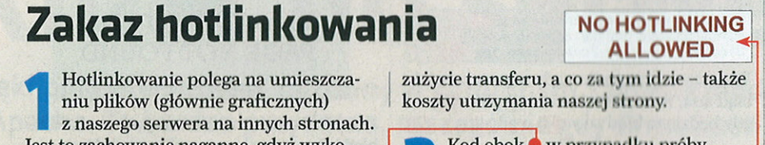
1 Moduł **mod_expires** pozwala na odciążenie serwera poprzez manipulację nagłówkami, które informują przeglądarkę internetową, jak często zmienia się zawartość naszej witryny. Zamiast na przykład pobierać za każdym razem te same grafiki, ładuje je z pamięci podręcznej, przez co znacznie zmniejsza się transfer plików oraz czas potrzebny na wczytanie strony.

2 Poniższy kod prezentuje zastosowanie modułu w praktyce. W pierwszej linii 1 sprawdzamy, czy moduł został załadowany, jeżeli okazałoby się, że nie – wówczas cały następny kod nie zostanie wykonany. Uruchamiamy moduł poleceniem 2, a następnie ustawiamy 2-miesięczny czas „ważności" dla wszystkich innych plików 3 niż zadeklarowane w dalszej części. Jeżeli chcemy ustalić go dla konkretnych typów plików, wówczas zapisujemy to w taki sposób 4 (w przykładzie dotyczy to plików jpg).

```
<ifmodule mod_expires.c>   1
ExpiresActive On   2
ExpiresDefault „access plus 2 months"   3
ExpiresByType image/jpg „access plus 1 week"   4
ExpiresByType image/x-icon „access plus 2 years „
ExpiresByType text/html „access plus 2 weeks 2 hours 45 minutes"   5
</ifmodule>
```

3 Możemy także łączyć kilka określeń czasu 5, na przykład ustawiając czas wygaśnięcia plików html na 2 tygodnie, 2 godziny i 45 minut.

Zakaz hotlinkowania



1 Hotlinkowanie polega na umieszczaniu plików (głównie graficznych) z naszego serwera na innych stronach. Jest to zachowanie naganne, gdyż wykorzystuje cenne zasoby naszego serwera, zmniejsza jego przepustowość, zwiększa zużycie transferu, a co za tym idzie – także koszty utrzymania naszej strony.

2 Kod obok w przypadku próby pobrania plików 1 z adresu innego niż nasz 2 wyświetla grafikę informującą, że hotlinkowanie jest zabronione.

```
RewriteEngine On
RewriteCond %{HTTP_REFERER} !^http://(www\.)?twojastrona\.pl [NC]
RewriteRule \.(jpg|gif|wmv|mov)$ /wykrytohotlink.jpg [L]
```

Deklarowanie własnych zmiennych serwera

1 Moduł **mod_setenvif** umożliwia nam tworzenie własnych zmiennych, które później można odczytać, na przykład wykorzystując skrypt PHP. Dzięki takiemu rozwiązaniu jesteśmy w stanie wyświetlać daną stronę w zależności od użytej przeglądarki czy też systemu operacyjnego. Poniższy kod przypisuje zmiennej **system_operacyjny** wartość Linux lub Windows w zależności od używanego systemu.

```
BrowserMatchNoCase linux system_operacyjny=Linux
BrowserMatchNoCase win system_operacyjny=Windows
```

2 Wartości ustawionych przez nas zmiennych odczytujemy w PHP w następujący sposób:

```
<?php
echo $_SERVER['system_operacyjny'];
?>
```

Dzięki temu uzyskane informacje możemy umieścić w treści strony lub wykorzystać w tworzonych dynamicznie obrazach typu userbar (patrz porada Zmiany w traktowaniu typów plików na następnej stronie).

Listowanie katalogu

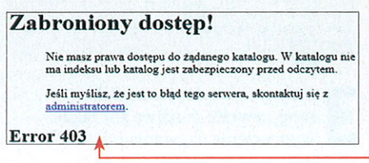
1 Istnieje możliwość zadeklarowania wczytywania innego pliku niż domyślny (index.php, index.html i tym podobne), czego dokonujemy wpisem.

```
DirectoryIndex startowa.php inna.php jeszczeinna.html
```

Kolejność jest tutaj niezwykle istotna, ponieważ w takim porządku nastąpi próba wyświetlenia poszczególnych plików (najpierw startowa.php, jeżeli nie zostanie odnaleziony, to inna.php i tak dalej).

2 Jeżeli chcemy całkowicie zablokować pokazywanie zawartości katalogów, wyłączamy opcję indeksowania. Gdy któryś z użytkowników będzie chciał wyświetlić zawartość katalogu, w którym nie będzie domyślnego pliku do załadowania, wówczas wyświetli się błąd 403 – dostęp zabroniony.

```
Options -Indexes
```

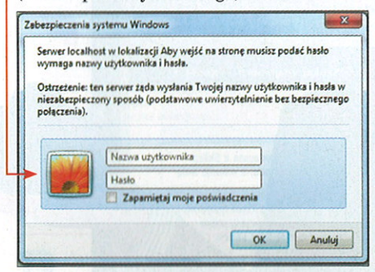



```
IndexIgnore *.jpg   1
IndexIgnore *.jpeg   1
IndexIgnore niewidoczne.html   2
```

3 Możemy także ukrywać wybrane przez nas typy plików 1 lub też konkretne pliki 2. Dzięki temu zabiegowi zostaną one pominięte podczas prezentowania zawartości katalogu.

Zabezpieczanie folderów hasłem

1 Aby zabezpieczyć folder i wszystkie jego podkatalogi przed niepowołanym dostępem, dopisujemy do pliku **.htaccess** (w zabezpieczanym katalogu) zawartość.

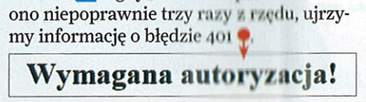



```
AuthName „Aby wejść na stronę musisz podać hasło"
AuthType Basic
AuthUserFile C:/xampp/htdocs/.htpasswd   1
Require valid-user   2
```

2 Kolejnym krokiem jest utworzenie pliku **.htpasswd** (nazwa może być dowolna, lecz przyjęło się właśnie takie jego nazywanie), który będzie odpowiedzialny za przechowywanie nazw użytkowników oraz przypisanych im haseł.

3 W systemie Windows z zainstalowanym pakietem XAMPP uruchamiamy Wiersz polecenia, powtarzając krok 2 z podrozdziału Tworzenie plików **.htaccess**. Wydając komendę **cd**, przechodzimy do następującego katalogu `C:\xampp\apache\bin>_` (w przypadku standardowej instalacji XAMPP). Uruchamiamy program **htpasswd** z parametrami. Pierwszy z nich każe utworzyć nowy plik z hasłem [redacted], drugi wskazuje jego nazwę [redacted], a trzeci użytkownika `ekspert`.

4 Po podaniu parametrów wciskamy klawisz [enter]. Program prosi nas o podanie hasła, które wpisujemy, a następnie o jego powtórzenie. Każdorazowo potwierdzamy je klawiszem Enter. Po zakończeniu pracy programu w katalogu **C:\xampp\apache\bin** został utworzony plik **.htpasswd** z nazwą użytkownika i jego hasłem. Przenosimy go do katalogu, który chcemy zabezpieczyć. W linijce 1 wpisujemy ścieżkę do niego i umożliwiamy zalogowanie się tylko tym użytkownikom, którzy podali poprawny login i hasło 2 – gdy zostanie wprowadzone ono niepoprawnie trzy razy z rzędu, ujrzymy informację o błędzie 401.



5 Aby dodatkowo zwiększyć poziom takiego zabezpieczenia, możemy stworzyć białą listę adresów IP, które mają prawo dostępu do chronionego katalogu (w przeciwnym wypadku każdy może próbować odgadnąć hasło).

6 W tym celu do powyżej stworzonego pliku dopisujemy następujące dyrektywy. Ważne jest, aby dobrze zadeklarować kolejność aplikowania poleceń **deny** oraz **allow** 1. Gdybyśmy dokonali tego w odwrotny sposób, najpierw nastąpiłoby zezwolenie na dostęp z adresów 2, a potem przez nadpisanie zablokowalibyśmy każdy adres IP 3 – w rezultacie nikt nie uzyskałby możliwości zalogowania się. Zapis 4 oznacza, że zezwalamy na łączenie się osób z adresem IP rozpoczynającym się od 192.167.98, a pozostała część może mieć dowolną wartość, na przykład 192.167.98.134, 192.167.98.12.

```
Order deny, allow   1
Deny from all   3
Allow from 192.167.98.   4
Allow from 192.178.94.56   2
```

7 Można w ten sposób zabezpieczać pojedyncze pliki 1 lub pasujące do odpowiedniego wzorca 2, czyli w tym wypadku dotyczyć to będzie plików o rozszerzeniach .gif i .jpeg. W poniższym przykładzie zezwalamy na dostęp jedynie komputerowi lokalnemu (127.0.0.1 lub localhost).

```
<Files cennyplik.html>   1
Order deny, allow
Deny from all
Allow from 127.0.0.1
</Files>
<FilesMatch „\. (gif|jpeg)$">   2
Order deny, allow
Deny from all
Allow from 127.0.0.1
</FilesMatch>
```

Zmiany w traktowaniu typów plików

1 Dzięki zastosowaniu odpowiednich dyrektyw jesteśmy w stanie dokonywać zmiany w traktowaniu typów plików przez serwer Apache. Aby na przykład zapobiec otwieraniu plików zawierających muzykę lub klipy wideo przez domyślny program, dopisujemy następujące polecenie, które wymusi ich ściągnięcie przez przeglądarkę internetową.

```
AddType application/octet-stream .mp3
.wmv .mp4 .mov
```

2 Istnieje też pewna ciekawa sztuczka, która umożliwia nam traktowanie plików z rozszerzeniem .jpg jako plików .php. Wykorzystuje się to w przypadku pasków użytkowników umieszczanych w podpisach, które wyświetlają dynamicznie treść, na przykład adres IP użytkownika oraz nazwę jego systemu operacyjnego lub inne informacje. Aby skorzystać z takiej możliwości, wystarczy dopisać do pliku .htaccess następującą treść. Od tej pory pliki z rozszerzeniem .jpg będą przetwarzane przez interpreter PHP.

```
AddType application/x-httpd-php .jpg
```

3 Na dołączonej płycie znaleźć można prosty kod PHP (**podpis.php**), który w sposób dynamiczny wyświetla aktualny czas, system operacyjny oraz adres IP osoby, która go wywołuje. Wystarczy jedynie zmienić jego rozszerzenie z **.php** na **.jpeg** i przepisać kod z poprzedniego punktu do pliku **.htaccess**. W skrypcie użyto także zmiennej serwera z poprzedniej strony, która przechowuje informację o systemie operacyjnym używanym przez użytkownika (jej deklarację należy umieścić w pliku **.htaccess**).

Ustawienia parsera PHP

1 Wpisując odpowiednie dyrektywy do pliku **.htaccess**, jesteśmy w stanie manipulować zmiennymi, które umożliwiają zmiany w zachowaniu interpretera PHP. W przykładzie wyłączamy pokazywanie błędów 1 oraz uniemożliwiamy tworzenie zmiennych globalnych 2, zmieniamy wersję PHP na 5, a także ustawiamy maksymalny rozmiar transferowanego przez protokół http pliku na 8 MB.

```
php_flag display_errors off   1
php_flag register_globals off   2
SetEnv PHP_VER 5   3
php_value upload_max_filesize 8M   4
```

2 Listę wszystkich zmiennych, których wartości możemy modyfikować, znajdziemy na stronie www.php.net/manual/pl/ini.list.php, są one oznaczone w polu **Changeable** jako **PHP_INI_ALL** lub **PHP_INI_PERDIR**. W przypadku opcji, które mogą przyjmować jedynie wartości logiczne – **true**, **false** lub **0,1** (prawda, fałsz) – używamy 3, w pozostałych przypadkach – czyli takich, w których można przypisać zmiennej konkretną wartość – 4.

MK