helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

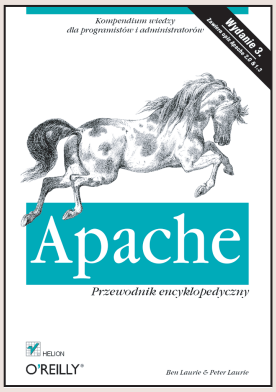
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

Apache. Przewodnik encyklopedyczny. Wydanie III



Autorzy: Ben Laurie, Peter Laurie
Tłumaczenie: Tomasz Sadowski
ISBN: 83-7361-124-X
Format: B5, stron: 704
Przykłady na ftp: 1249 kB

Udostępniany nieodpłatnie serwer WWW Apache obsługuje dziś ponad połowę wszystkich witryn w internecie i systematycznie zwiększa swój udział w rynku.
Książka „Apache. Przewodnik encyklopedyczny. Wydanie III" autorstwa dwóch kluczowych członków Zespołu Apache, opisuje sposób pobrania, instalacji i zabezpieczania tego serwera oraz omawia popularne rozszerzenia, umożliwiające konstruowanie na jego podstawie aplikacji WWW.

Serwer Apache osiągnął rangę kompletnego systemu i skutecznie konkuruje z wszystkimi pozostałymi serwerami HTTP niezależnie od tego, czy będziemy porównywać je pod kątem oferowanych możliwości, efektywności, czy też szybkość działania. Apache jest przy tym dostępny dla wielu platform systemowych, w tym dla różnego rodzaju systemów Unix i systemów z rodziny Windows.

Prezentowana Czytelnikom trzecia już edycja książki opisuje najpopularniejsze wersje 1.3 i 2.0 serwera Apache dla systemów Windows i Unix kładąc szczególny nacisk na:

- pobranie i kompilację oprogramowania serwera,
- konfigurację i uruchamianie serwera w systemach Windows i Unix (obejmując też zagadnienia związane ze strukturami katalogów serwera i serwerami wirtualnymi),
- omówienie interfejsu programowego serwera (w wersjach 1.3 i 2.0),
- szczegółowy opis zagadnień związanych z zabezpieczeniem serwera Apache i wdrożeniem go w rozbudowanych witrynach,
- prezentacje pełnej listy dyrektyw konfiguracyjnych,
- informacje na temat instalacji i testowania skryptów języka Perl uruchamianych w trybie CGI oraz instalacji i korzystania z rozszerzeń, takich jak mod_perl, PHP, JServ, Tomcat i Cocoon.

Dzięki książce „Apache. Przewodnik encyklopedyczny" administratorzy witryn WWW nie mający dotychczas do czynienia z serwerem Apache mogą zapoznawać się z jego działaniem stopniowo, analizując i wdrażając przykładowe witryny prezentujące kolejne etapy konfiguracji serwera. Doświadczeni administratorzy i programiści (niezależnie od tego, czy ich środowiskiem roboczym jest system Windows, czy Unix) docenią natomiast te fragmenty książki, które składają się na kompletną i zwięzłą dokumentację całego serwera.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Przedmowa *9*

Rozdział 1. *Wprowadzenie* *19*

Co robi serwer WWW? 19
Jak działa Apache? 23
Apache i sieci 24
Jak działa klient? 30
Co dzieje się po stronie serwera? 32
Planowanie instalacji serwera Apache 33
Windows? 36
Która wersja Apache? 36
Instalowanie serwera Apache 37
Kompilacja serwera Apache 1.3.x w systemie Unix 42
Nowe funkcje Apache 2 53
Instalacja Apache 2.0 w systemie Unix 56
Apache w systemach Windows 57

Rozdział 2. *Konfiguracja serwera Apache — odsłona pierwsza* *63*

Co to właściwie jest witryna WWW? 63
Pierwsza witryna — site.toddle 66
Uruchomienie serwera w Uniksie 67
Uruchomienie serwera w Windows 81
Dyrektywy 85
Obiekty współużytkowane 87

Rozdział 3. *Wielkie otwarcie* ***91***
Więcej i lepiej, czyli site.simple 91
Zaczynamy na poważnie 95
Dyrektywy blokowe 98
Pozostałe dyrektywy 102
Nagłówki odpowiedzi HTTP 112
Restart serwera 117
Pliki .htaccess 118
Metapliki w standardzie CERN 118
Określanie terminu ważności dokumentu 119

Rozdział 4. *Serwery wirtualne* ***123***
Implementacja dwóch witryn 123
Implementacja serwerów wirtualnych 123
Dwie kopie serwera Apache 128
Serwery wirtualne konfigurowane dynamicznie 132

Rozdział 5. *Uwierzytelnianie* ***137***
Protokół uwierzytelniania 137
Dyrektywy sterujące uwierzytelnianiem 139
Hasła w systemie Unix 144
Hasła w systemie Windows 146
Hasła w sieci WWW 146
Punkt widzenia klienta 146
Skrypty CGI 147
Co by tu jeszcze… 147
Dyrektywy order, allow i deny 147
Pliki DBM w Uniksie 151
Uwierzytelnianie oparte na skrótach wiadomości 155
Dostęp anonimowy 159
Kilka ćwiczeń 162
Automatyczne przekazywanie danych o użytkowniku 163
Jak korzystać z plików .htaccess? 164
Priorytety dyrektyw lokalnych 166

Rozdział 6. ***Opis i negocjacja zawartości dokumentów*** ***169***

Typy MIME 169
Uzgadnianie zawartości 177
Uzgadnianie języka 179
Mapy typów 183
Przeglądarki a protokół HTTP/1.1 185
Mechanizm filtrów 186

Rozdział 7. ***Indeksowanie katalogów*** ***191***

Lepszy indeks — ale jak? 191
Rozszerzenia indeksów tworzonych przez użytkownika 202
Mapy graficzne 205
Dyrektywy związane z mapami graficznymi 210

Rozdział 8. ***Przeadresowywanie*** ***213***

Dyrektywa Alias 214
Translacja adresów URL 222
Korygowanie adresów 230

Rozdział 9. ***Apache jako serwer pośredniczący*** ***233***

Bezpieczeństwo 233
Dyrektywy sterujące serwerem pośredniczącym 234
Czyżby błąd? 239
Wydajność serwera 239
Nasza konfiguracja 242

Rozdział 10. ***Co jest grane?*** ***249***

Rejestrowanie za pośrednictwem skryptu i bazy danych 249
Dzienniki serwera Apache 250
Rejestrowanie konfiguracji 260
Status serwera 263

Rozdział 11. ***Bezpieczeństwo informacji*** ***267***

Użytkownicy wewnętrzni i zewnętrzni 269
Podpisy cyfrowe i pieniądz elektroniczny 271
Certyfikaty cyfrowe 276
Zapory sieciowe 278

Zagadnienia prawne ... 282
Secure Sockets Layer (SSL) ... 283
Podstawowe mechanizmy bezpieczeństwa w serwerze Apache ... 283
Dyrektywy sterujące SSL ... 301
Zestawy szyfrów ... 321
Bezpieczeństwo w praktyce ... 328
Przyszłość zabezpieczeń ... 333

Rozdział 12. *Duża witryna WWW* ... 335
Konfiguracja komputera ... 335
Bezpieczeństwo serwera ... 335
Zarządzanie dużą witryną ... 340
Oprogramowanie dodatkowe ... 343
Skalowalność ... 350
Równoważenie obciążenia ... 352

Rozdział 13. *Piszemy aplikacje* ... 367
Witryny WWW jako aplikacje ... 367
Definiowanie logiki aplikacji ... 372
Języki XML i XSLT w aplikacjach WWW ... 377

Rozdział 14. *Polecenia wstawiane SSI* ... 379
Informacja o rozmiarze pliku ... 382
Informacja o czasie modyfikacji pliku ... 383
Wstawianie treści plików ... 384
Wykonywanie skryptów CGI ... 384
Zmienne w poleceniach SSI ... 385
Filtry SSI w Apache 2.0 ... 385

Rozdział 15. *PHP* ... 389
Instalacja języka PHP ... 390
Site.php ... 391

Rozdział 16. *Skrypty CGI i język Perl* ... 397
Świat CGI ... 397
Udostępnienie skryptu serwerowi Apache ... 399
Ustawianie wartości zmiennych środowiskowych ... 417

Ciasteczka ... 418
Dyrektywy serwera Apache związane z obsługą skryptów ... 430
suEXEC w Uniksie ... 433
Procedury obsługi ... 440
Akcje ... 442
Przeglądarki ... 444

Rozdział 17. *Moduł mod_perl* ... *447*
Moduł mod_perl — jak to działa? ... 449
Dokumentacja modułu mod_perl ... 450
Instalacja modułu mod_perl — wariant prostszy ... 450
Dostosowanie skryptów do wymagań modułu mod_perl ... 454
Zmienne globalne ... 454
Dyrektywa strict ... 457
Odświeżanie pamięci serwera ... 457
Otwieranie i zamykanie plików ... 458
Konfiguracja serwera dla modułu mod_perl ... 458

Rozdział 18. *Kontenery apletów: JServ i Tomcat* ... *463*
Moduł mod_jserv ... 464
Tomcat ... 476
Tomcat i Apache ... 482

Rozdział 19. *XML i serwlet Cocoon* ... *487*
Język XML ... 487
Język XML a Perl ... 491
Cocoon ... 492
Cocoon 1.8 i JServ ... 492
Cocoon 2.0.3 i Tomcat ... 496
Testowanie serwletu Cocoon ... 497

Rozdział 20. *Interfejs programowy serwera Apache* ... *501*
Dokumentacja ... 502
Biblioteka APR ... 502
Pule ... 502
Globalna struktura konfiguracyjna ... 504
Lokalna struktura konfiguracyjna ... 507

Opis żądania ... 510
Dostęp do danych konfiguracyjnych i opisu żądania ... 514
Zaczepy, zaczepy opcjonalne i funkcje opcjonalne ... 514
Filtry, kubełki i zespoły ... 524
Moduły ... 536

Rozdział 21. ***Piszemy własny moduł serwera Apache*** ***... 539***

Wprowadzenie ... 540
Kody stanu ... 541
Struktura module ... 543
Przykład od A do Z ... 583
Wskazówki ogólne ... 602
Przystosowywanie kodu do wersji 2.0 ... 602

Dodatek A ***Interfejs API 1.x*** ***... 607***

Pule ... 607
Globalna struktura konfiguracyjna ... 609
Lokalna struktura konfiguracyjna ... 610
Opis żądania ... 610
Dostęp do danych konfiguracyjnych i opisu żądania ... 613
Funkcje API ... 613

Skorowidz ***... 669***

3

Wielkie otwarcie

Mając już działający w oparciu o podstawową konfigurację serwer, można pokusić się o głębszą analizę jego możliwości, z większą liczbą szczegółów. Na szczęście różnice pomiędzy wersjami serwera Apache dla systemów uniksowych i systemów Windows kończą się po przebrnięciu przez początkową konfigurację; później można się już skupić na tworzeniu działającej strony WWW.

Więcej i lepiej, czyli site.simple

W chwili obecnej możemy już przystąpić do konstruowania przykładowych witryn WWW, których zawartość można znaleźć również na załączonej do książki płycie CD. Aby zachować jakiś związek z realnym światem, oprzemy nasze eksperymenty na przykładzie fikcyjnej firmy Butterthlies, Inc., a dokładniej jej polskiej filii — Butterthlies Polska, sp. z o.o. Nasza firma zajmuje się sprzedażą artystycznych pocztówek i ma zamiar zaistnieć w Internecie. W związku z powyższym musimy przydzielić jej jakieś adresy IP; jako że wszystko to dzieje się na niby i ogranicza się do eksperymentu, adresy te będą zawierały się w obrębie naszej sieci lokalnej. Dzięki temu komputery biorące udział w doświadczeniu nie będą musiały łączyć się z Internetem. Techniczna strona przydzielenia adresów sprowadza się do zmodyfikowania zawartości plików *\windows\hosts* (na komputerze „windowsowym" pełniącym rolę klienta) i */etc/hosts* (na maszynie uniksowej będącej serwerem), tak aby zawierały następujące wpisy:

```
127.0.0.1 localhost
192.168.123.2 www.butterthlies.com.pl
192.168.123.2 sales.butterthlies.com.pl
192.168.123.3 sales-IP.butterthlies.com.pl
192.168.124.1 www.gdzies-tam.com
```

Pierwszy wpis (*localhost*) jest obowiązkowym elementem pliku i nie należy go usuwać; nie powinieneś również kierować do niego żadnych żądań HTTP, gdyż wyniki mogą być dziwne.

W kwestii pozostałych wpisów najlepiej skonsultuj się z administratorem swojej sieci lokalnej.

Witryna *site.simple* jest lekko zmodyfikowanym wariantem *site.toddle*. Skrypt *go* powinien działać bez problemów. Przypomnijmy sobie procedury uruchamiania i zatrzymywania serwera (w różnych systemach operacyjnych):

UNIX W Uniksie:

```
test -d logs || mkdir logs
httpd -d 'pwd' -f 'pwd'/conf/httpd.conf
```

Zatrzymanie polega na wykonaniu polecenia `kill` z odpowiednim identyfikatorem.

WIN32 W Windows:

Otwórz okno DOS i w wierszu poleceń wpisz:

```
> cd "\program files\apache group\apache"
> apache -k start
Apache/1.3.26 (Win32) running ...
```

Aby zatrzymać serwer, otwórz drugie okno DOS i wpisz:

```
> apache -k stop
> cd logs
```

i ewentualnie:

```
> edit error.log
```

Procedury te będą obowiązywać dla wszystkich witryn demonstrowanych w ramach przykładów do książki, dlatego nie będziemy już przytaczać procedur sterujących uruchamianiem serwera.

Również inne różnice w opisywanych dalej konfiguracjach serwera Apache dla Uniksa i Windows powinny być minimalne. O ile nie będzie to wyraźnie zaznaczone, należy zakładać, że wszystkie opisy stosują się w tej samej formie do obu wersji.

Nie od rzeczy byłoby rejestrować w jakimś pliku poczynania naszego serwera. Podczas pracy nad pierwszym wydaniem książki mieliśmy ułatwione zadanie, gdyż używana przez nas wersja Apache automatycznie tworzyła w katalogu *…/site.simple/logs* plik dziennika o nazwie *access_log*. Z sobie tylko wiadomych przyczyn projektanci serwera Apache postanowili jednak zerwać z przeszłością i kolejne wersje wymagają już jawnego określenia położenia dziennika w pliku konfiguracji serwera. Służy do tego dyrektywa `TransferLog`.

W swojej obecnej postaci plik *httpd.conf* powinien wyglądać następująco:

```
User webuser
Group webgroup
ServerName localhost
DocumentRoot /usr/www/APACHE3/site.simple/htdocs
TransferLog logs/access_log
```

Katalog *…/site.simple/htdocs* zawiera, jak poprzednio, tylko jeden plik o nazwie *1.txt*:

```
Witamy w witrynie site.simple!
```

W chwili obecnej możesz już uruchomić serwer poleceniem `go`, przesiąść się do komputera-klienta i za pomocą przeglądarki WWW zajrzeć pod adres *http://www.butterthlies.com.pl*. Powinieneś ujrzeć coś takiego:

```
Index of /
* Parent Directory
* 1.txt
```

Kliknięcie łącza do pliku *1.txt* wyświetli zawarty w tym ostatnim komunikat.

Wszystko ładnie, ale jedna rzecz jest tu cokolwiek zagadkowa: ten sam wynik uzyskujemy, łącząc się z adresem *http://sales.butterthlies.com.pl*. Jakim cudem *w ogóle* uzyskujemy jakąś odpowiedź, skoro ani jeden, ani drugi adres (ani też odpowiadające im numery IP) nie został wpisany do pliku konfiguracji zawartego w katalogu *site.simple*?

Odpowiedź jest prosta. Konfigurując komputer pełniący rolę serwera, nakazaliśmy jego interfejsowi sieciowemu reagować na komunikaty skierowane pod adresy:

```
192.168.123.2
192.168.123.3
```

Serwer Apache przez domniemanie prowadzi nasłuch na wszystkich adresach zdefiniowanych dla danego systemu, a dla wybranych również udziela odpowiedzi. Jeśli w systemie zdefiniowano serwery wirtualne (w naszym przypadku nie zdefiniowano), Apache przegląda ich listę w poszukiwaniu adresu IP, który odpowiada adresowi, pod którym odebrano nadesłane żądanie obsługi. Ustaliwszy, o który serwer wirtualny chodzi, Apache używa odpowiadającego mu bloku w pliku konfiguracyjnym, a jeśli blok taki nie istnieje — głównego bloku konfiguracji serwera. Do zagadnienia tego wrócimy w dalszej części rozdziału, omawiając dyrektywy `BindAddress`, `Listen` i `<VirtualHost>`, dające administratorowi skuteczną kontrolę nad serwerami wirtualnymi.

Należy zauważyć, że prezentowany tu sposób pracy, polegający na częstej zmianie wykorzystywanych konfiguracji, jest w stanie skutecznie zdezorientować przeglądarkę, co stwierdziliśmy zarówno w przypadku programu Netscape, jak i Internet Explorera. W przypadku przeglądarki Netscape przekonanie się o poprawnym działaniu serwera wymagało na ogół odświeżania przeglądanych plików przez klikanie przycisku *Reload* przy wciśniętym klawiszu *Ctrl*. W skrajnych przypadkach konieczne było wyłączenie buforowania stron w pamięci podręcznej poprzez wydanie poleceń *Edit-Preferences-Advanced-Cache*, wyzerowanie rozmiarów bufora dyskowego (*Disk Cache*) i pamięciowego (*Memory Cache*) i wymuszenie każdorazowego badania aktualności dokumentu (zazna-

czenie przycisku opcji *Every Time*). W przypadku Internet Explorera konieczne okazało się ustawienie opcji sterującej częstotliwością porównywania zawartości bufora z treścią odpowiedzi na *Przy każdej wizycie na tej stronie* (w oknie ustawień *Tymczasowych plików internetowych*) w ramach okna *Opcje internetowe*. Jeśli nie wykonasz tych zabiegów, musisz liczyć się z możliwością wyświetlenia przez przeglądarkę mieszanki kilku ostatnio odebranych z serwera stron. Dzieje się tak oczywiście dlatego, iż w naszych doświadczeniach bez przerwy żonglujemy różnymi wersjami witryn, katalogów i plików, czego trzeźwo myślący administrator ani użytkownik na ogół nie robi. Jeśli jakiś plik zostanie zastąpiony starszą wersją, przeglądarka dochodzi do skądinąd słusznego wniosku, że zawartość pamięci podręcznej jest bardziej aktualna i oczywiście ją wyświetla.

Wróćmy jednak do rzeczy. Zakończ pracę programu Apache, naciskając na klawiaturze serwera klawisze *Ctrl*+*C* (lub inną kombinację służącą do przerywania działania programu) i obejrzyj zawartość pliku *…/logs/access_log*. Powinien on zawierać coś w rodzaju

```
192.168.123.1 - - [<data i godzina>] "GET / HTTP/1.0" 200 256
```

Liczba `200` jest kodem odpowiedzi zwracanym przez serwer (ang. *HTTP response code*) i oznacza, że operacja się udała, zaś `256` — liczbą przesłanych podczas transakcji bajtów. Plik *…/error_log* powinien być pusty, ponieważ cała operacja przebiegła bez błędów. Warto oczywiście zaglądać tam od czasu do czasu, chociaż skojarzenie daty i czasu wpisu z błędem, który wystąpił jakiś czas temu, bywa kłopotliwe i nieraz trzeba w tym celu mocno wysilić pamięć.

Życie niestety bywa mniej przyjemne i czasami coś się psuje. Dla przykładu, klient może zażądać od serwera nieistniejącego dokumentu. Sytuację taką można obsłużyć za pomocą dyrektywy `ErrorDocument`.

ErrorDocument

Dyrektywa `ErrorDocument` pozwala na określenie czynności podejmowanej w sytuacji, kiedy klient odwołuje się do nieistniejącego dokumentu.

```
ErrorDocument kod-błędu nazwa-dokumentu
Zastosowanie: konfiguracja główna, serwery wirtualne, katalogi, pliki .htaccess
```

W przypadku wystąpienia błędu, Apache może zrobić jedną z czterech rzeczy:

1. Przekazać klientowi elementarny komunikat o błędzie, o sztywnej, niezmiennej treści.
2. Przekazać klientowi komunikat zdefiniowany przez administratora.
3. Przeadresować żądanie klienta do lokalnego adresu URL przeznaczonego do obsługi błędu.
4. Przeadresować żądanie klienta do zewnętrznego adresu URL.

Domyślnie realizowana jest czynność pierwsza, natomiast trzy pozostałe opcje mogą być wymuszone za pomocą dyrektywy `ErrorDocument`, której argumentami są kod odpowiedzi serwera i komunikat o błędzie lub adres URL. Komunikat różni się od adresu

tym, iż jest poprzedzony znakiem cudzysłowu ("), który nie jest oczywiście przesyłany do klienta. W niektórych przypadkach Apache dołącza do komunikatu dodatkowe informacje i objaśnienia.

Używane w dyrektywie `ErrorDocument` adresy URL mogą być adresami lokalnymi (rozpoczynają się wówczas znakiem ukośnika — „/") lub pełnymi adresami zawierającymi nazwę węzła. A oto kilka przykładów:

```
ErrorDocument 500 http://gdzies.tam.com/cgi-bin/test
ErrorDocument 404 /cgi-bin/zlyadres.pl
ErrorDocument 401 /rejestracja.html
ErrorDocument 403 "Nieczynne z powodu urlopu"
```

Zwróćmy uwagę, że użycie w dyrektywie `ErrorDocument` pełnego adresu URL (rozpoczynającego się od prefiksu metody, np. „http") powoduje przesłanie do klienta polecenia przeadresowania nawet wtedy, gdy dokument identyfikowany adresem znajduje się na tym samym serwerze. Fakt ten ma kilka konsekwencji, z których najważniejszą jest niemożność odwołania się do nielokalnego dokumentu w przypadku obsługi błędu numer `401` (co wynika z zasady działania podstawowego schematu uwierzytelniania (*Basic*) w protokole HTTP).

Zaczynamy na poważnie

Plik konfiguracji serwera *httpd.conf* znajdujący się w katalogu *…/site.first* ma postać:

```
User webuser
Group webgroup
ServerName localhost
DocumentRoot /usr/www/site.first/htdocs
TransferLog logs/access_log
```

W pierwszej edycji książki omawialiśmy w tym podrozdziale również dyrektywy `AccessConfig` i `ResourceConfig`. Użycie w nich jako argumentu urządzenia pustego (*/dev/null* w Uniksie, *NUL* w Windows) pozwalało na zablokowanie odwołań do plików *srm.conf* i *access.conf*, co było niezbędne w przypadku nieobecności tych ostatnich. Ponieważ nowsze wersje Apache po prostu nie przejmują się brakiem wspomnianych plików, użycie obu dyrektyw jest obecnie zbędne. W przypadku ich użycia wskazywane przez nie pliki zostaną włączone do pliku konfiguracyjnego serwera. Od wersji 1.3.14 dyrektywy te mogą wskazywać nie pliki, a katalogi — wtedy do pliku konfiguracyjnego serwera włączona zostanie zawartość wszystkich plików przechowywanych w tych katalogach.

W Apache 2.0 dyrektywy `AccessConfig` i `ResourceConfig` zostały zniesione, więc ich umieszczenie w pliku spowoduje zgłoszenie przez serwer komunikatu o błędzie konfiguracji. Zamiast nich można zastosować dyrektywy `Include`: `Include conf/srm.conf` i `Include conf/access.conf` (w podanej kolejności) na końcu głównego pliku konfiguracyjnego.

Ponadto Apache 2.0, dość konsekwentnie, wymaga zdefiniowania dyrektywy `Listen`. Jej brak spowoduje przerwanie działania serwera i wyświetlenie komunikatu o błędzie następującej treści:

```
... no listening sockets available, shutting down.
```

WIN32 W systemach z rodziny Windows Apache ignoruje dyrektywy `User` i `Group`, w związku z czym mogą one zostać usunięte z pliku konfiguracyjnego.

Zadaniem serwera Apache jest dostarczanie klientom dokumentów w języku HTML, a jak do tej pory nie miał on specjalnie czego dostarczać. Spróbujmy zatem utworzyć prostą stronę WWW zawierającą ofertę firmy Butterthlies Polska i informacje o sposobie zakupu towaru.

Nieco teorii na temat projektowania WWW można znaleźć np. w systemie pomocy przeglądarki Netscape pod hasłem „Creating Net Sites". Po odbyciu elementarnego kursu języka HTML możemy już wyprodukować surową wersję wirtualnej broszurki:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
<title>Katalog firmy Butterthlies</title>
</head>
<h1> Witamy w firmie Butterthlies Polska</h1>
<h2>Nasza oferta na lato</h2>
<p> Wszystkie kartki można zamawiać w paczkach po 20 sztuk, po 8 złotych za
paczkę.
Przy zamówieniach powyżej 100 sztuk udzielamy 10% rabatu.
Ceny nie zawierają podatku VAT.
</p>
<hr>
<p>
Wzór 2315
<p align=center>
<img src="bench.jpg" alt="Ławeczka">
<p align=center>
Ławeczka dla zakochanych
<hr>
<p>
Wzór 2316
<p align=center>
<img src="hen.jpg" ALT="Kurnik a'la chińska pagoda">
<p align=center>
Chiński Kurnik
<HR>
<p>
Wzór 2317
<p align=center>
<img src="tree.jpg" alt="Bardzo ładne drzewo">
<p align=center>
Świątynia Dumania
<hr>
<p>
Wzór 2318
<p align=center>
<img src="bath.jpg" alt="Cokolwiek niecodzienne zdjęcie wanny">
<p align=center>
Wanna Surrealistyczna
```

```
<hr>
<p align=right>
Projekt pocztówek: Harriet@alart.demon.co.uk
<hr>
<br>
Butterthlies Polska sp. z o.o. * 99-000 Helionowo
</br>
</body>
</html>
```

UNIX Nasza strona pojawi się po raz pierwszy w katalogu *…/site.first/htdocs*, ale w dalszej części książki będziemy ją również wykorzystywać w wielu innych witrynach. W takiej sytuacji można ulokować odpowiednie pliki w jednym, „centralnym" katalogu i utworzyć do nich dowiązania z innych katalogów, wykorzystując uniksowe polecenie `ln`. Co więcej, każda zmiana „oryginalnego" pliku będzie natychmiast widoczna we wszystkich dowiązaniach. Mamy więc katalog */usr/www/APACHE3/main_docs* i plik dokumentu *catalog_summer.html*. Plik ten odwołuje się do kilku oryginalnych zdjęć, przechowywanych w postaci czterech plików *.jpg*. Wszystkie te pliki znajdują się w katalogu *…/main_docs* i zostaną dowiązane do odpowiednich plików w katalogu *htdocs*:

```
% ln /usr/www/APACHE3/main_docs/catalog_summer.html
% ln /usr/www/APACHE3/main_docs/bench.jpg
```

W ten sam sposób należy wykonać dowiązania do pozostałych plików. Powyższe polecenie powinno być wykonywane z wnętrza katalogu *…/site.first/htdocs*.

Po wykonaniu w tym katalogu polecenia `ls` okaże się, że jest on pełen potrzebnych nam plików.

WIN32 W systemie Windows nie istnieje pojęcie dowiązania, toteż będziemy musieli za każdym razem kopiować pliki.

Domyślny indeks witryny

Uruchom ponownie serwer poleceniem `go`. Przesiądź się do komputera-klienta i jeszcze raz otwórz stronę *http://www.butterthlies.com.pl*. Powinieneś zobaczyć coś takiego:

```
Index of /
* Parent Directory
* bath.jpg
* bench.jpg
* catalog_summer.html
* hen.jpg
* tree.jpg
```

Plik index.html

Powyższy wydruk zawiera domyślny indeks zawartości katalogu, będący swego rodzaju protezą, generowaną automatycznie przez program Apache w przypadku braku „prawdziwego" indeksu. Spróbujmy zatem stworzyć porządny indeks i zapisać go w pliku *…/htdocs/index.html*:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
<title>Indeks ofert firmy Butterthlies</title>
</head>
<body>
<p>Witamy w firmie Butterthlies Polska</p>
<ul>
<li><A href="catalog_sumer.html">Oferta - lato</A>
<li><A href="catalog_summer.html">Katalog - jesień</A>
</ul>
<hr>
<br>
Butterthlies Polska sp. z o.o. * 99-000 Helionowo
</br>
</body>
</html>
```

Aby wszystko wyglądało poważniej, dorzuciliśmy przy okazji jeszcze jeden plik, *catalog_autumn.html* (idąc po linii najmniejszego oporu, skopiowaliśmy plik *catalog_summer.html*, zmieniając w nim tylko porę roku), tak więc w indeksie ostatecznie znalazły się dwa łącza.

Jeśli klient (przeglądarka) w nadesłanym przez siebie żądaniu przekaże adres URL katalogu zawierającego plik *index.html*, Apache automatycznie prześle ten plik klientowi, traktując go jako domyślny indeks katalogu (zachowanie to można zmodyfikować za pomocą dyrektywy `DirectoryIndex`). Kolejne odwołanie do adresu *http://www.butterthlies.com.pl* powinno więc dać następujący efekt:

```
Witamy w firmie Butterthlies Polska
* Katalog - lato
* Katalog - jesień
--------------------------------------------------
Butterthlies Polska sp. z o.o. * 99-000 Helionowo
```

Oczywiście jako doświadczeni marketingowcy nie możemy zapomnieć o rejestracji naszej witryny w wyszukiwarkach internetowych. Dzięki temu już wkrótce nasze strony zaczną odwiedzać pierwsi klienci (zostawiając po sobie ślady w pliku *…/logs/access_log*), a kiedy zapoznają się z naszą fantastyczną ofertą… rozdzwonią się telefony z zamówieniami, a my wkrótce zostaniemy milionerami.

Dyrektywy blokowe

Apache udostępnia cały szereg tak zwanych *dyrektyw blokowych* (ang. *block directives*). Dyrektywy te pozwalają ograniczyć zasięg działania innych, zawartych w nich dyrektyw do określonych serwerów wirtualnych, katalogów czy też plików. Dyrektywy blokowe są niezwykle istotne w praktyce, bowiem to właśnie one — a w szczególności dyrektywa `<VirtualHost>` — umożliwiają administratorowi uruchomienie większej liczby niezależnych serwerów WWW poprzez pojedyncze wywołanie programu Apache. Stwierdzenie to nabierze większego sensu, gdy przejdziemy do omawiania obsługi kilku witryn (zobacz podrozdział „Implementacja dwóch witryn” w rozdziale 4.).

Obecnie zajmiemy się składnią dyrektyw blokowych.

<VirtualHost>

```
<VirtualHost węzeł[:port]>
...
</VirtualHost>
Zastosowanie: konfiguracja główna
```

Działanie dyrektywy <VirtualHost> w pliku konfiguracji serwera jest podobne do funkcji pełnionej przez znaczniki języka HTML. Nagłówek <VirtualHost> otwiera blok tekstu zawierający inne dyrektywy, odnoszące się do konkretnego serwera wirtualnego; koniec takiego bloku oznaczany jest ciągiem </VirtualHost>. Oto przykład:

```
...
<VirtualHost www.butterthlies.com.pl>
ServerAdmin sales@butterthlies.com.pl
DocumentRoot /usr/www/APACHE3/site.virtual/htdocs/customers
ServerName www.butterthlies.com.pl
ErrorLog /usr/www/APACHE3/site.virtual/name-based/logs/error_log
TransferLog /usr/www/APACHE3/site.virtual/name-based/logs/access_log
</VirtualHost>
...
```

Dyrektywa <VirtualHost> pozwala ponadto określić nazwę domenową lub adres IP (i ewentualnie numer portu) przypisany danemu serwerowi wirtualnemu. Jeśli port nie zostanie podany, domyślnie wykorzystywany jest port numer 80 (standardowy port używany w protokole HTTP) lub port określony za pomocą dyrektywy Port. Użycie w miejscu argumentu *węzeł* ciągu _default_ powoduje wreszcie, że zawartość danego bloku będzie odnosiła się do wszystkich serwerów nie uwzględnionych w innych blokach <VirtualHost>. W rzeczywistym systemie *węzeł* będzie oczywiście nazwą domenową lub adresem IP naszego serwera.

Oprócz dyrektywy <VirtualHost> Apache udostępnia jeszcze trzy inne dyrektywy pozwalające na ograniczenie zasięgu działania innych dyrektyw:

- <Directory>
- <Files>
- <Location>

Powyższe dyrektywy wymieniliśmy w kolejności rosnącego priorytetu. Oznacza to, że działanie dyrektywy <Directory> może być zmodyfikowane przez nadrzędną wobec niej <Files>, a ta z kolei musi „ustąpić pierwszeństwa" dyrektywie <Location>. Dyrektywy przetwarzane są grupami w następującej kolejności:

1. <Directory> (nie zawierające wyrażeń regularnych), równolegle z zawartością plików *.htaccess*[1], przy czym te ostatnie mają priorytet.
2. <DirectoryMatch> i <Directory> zawierające wyrażenia regularne.
3. <Files> i <FilesMatch> (równolegle).
4. <Location> i <LocationMatch> (równolegle).

[1] I oczywiście dla każdego katalogu zawartego w ścieżce dostępu.

W pierwszym przypadku katalogi przetwarzane są w kolejności od najmniejszego do największego[2]. Kolejność przetwarzania dla pozostałych grup określona jest porządkiem zapisu w pliku konfiguracji serwera. Dyrektywy zawarte wewnątrz bloków `<VirtualHost>` realizowane są po wykonaniu odpowiadających im bloków położonych na zewnątrz.

<Directory> i <DirectoryMatch>

```
<Directory katalog>
...
</Directory>
Zastosowanie: konfiguracja główna, serwery wirtualne
```

Dyrektywa `<Directory>` pozwala na ograniczenie zasięgu działania bloku dyrektyw do wybranego katalogu lub grupy katalogów. Należy tu podkreślić, że specyfikacja katalogu traktowana jest jako *bezwzględna*, tj. dyrektywa `<Directory />` obejmie swoim działaniem nie katalog `DocumentRoot` (i jego podkatalogi), ale *cały system plików*, poczynając od katalogu głównego. Nazwa katalogu może zawierać symbole wieloznaczne (`?`) (dowolny znak) i (`*`) (dowolny ciąg dowolnych znaków), a także nawiasy kwadratowe (`[ ]`), służące do definiowania zbiorów znaków. Dla przykładu, specyfikacja `[a-d]` oznacza „dowolny ze znaków a, b, c lub d". Umieszczenie znaku tyldy (`~`) na początku nazwy katalogu pozwala na użycie w niej kompletnych wyrażeń regularnych[3].

Dyrektywa `<DirectoryMatch>` działa tak samo jak `<Directory ~ >`, tj. akceptuje definicję nazwy katalogu w postaci wyrażenia regularnego, tak więc zapisy:

```
<Directory ~ /[a-d].*>
```

i

```
<DirectoryMatch /[a-d].*>
```

są identyczne i odnoszą się do wszystkich katalogów, których nazwy rozpoczynają się od liter „a", „b", „c" lub „d".

<Files> i <FilesMatch>

```
<Files plik>
...
</Files>
Zastosowanie: konfiguracja główna, serwery wirtualne, pliki .htaccess
```

Dyrektywa `<Files>` pozwala ograniczyć działanie bloku dyrektyw do plików o nazwie zadanej parametrem *`plik`*. Nazwa ta określana jest względem katalogu `DocumentRoot` i może zawierać symbole wieloznaczne oraz wyrażenia regularne poprzedzone znakiem tyldy (`~`). Dyrektywa `<FilesMatch>` używana jest wraz z wyrażeniami regularnymi

[2] Określenia te odnoszą się do liczby elementów katalogu, a nie ich łącznej objętości.

[3] Szczegółowe omówienie wyrażeń regularnych znajdziesz w książce Jeffreya E. F. Friedla *Mastering Regular Expressions* (O'Reilly & Associates; jak na razie książka ta nie doczekała się niestety polskiego tłumaczenia).

nie poprzedzonymi znakiem tyldy. Aby zatem ograniczyć działanie bloku do trzech najpopularniejszych w Internecie typów plików graficznych, musimy użyć dyrektywy:

```
<FilesMatch "\.gif|jpe?g|png$">
```

zaś aby zapewnić specjalne traktowanie katalogom produktów firmy Butterthlies Polska, możemy użyć konstrukcji:

```
<FilesMatch catalog.*>
```

W odróżnieniu od `<Directory>` i `<Location>`, dyrektywa `<Files>` może być umieszczana w plikach *.htaccess*.

<Location> i <LocationMatch>

```
<Location adres-URL>
...
</Location>
Zastosowanie: konfiguracja główna, serwery wirtualne
```

Użycie dyrektywy `<Location>` pozwala na ograniczenie zasięgu działania bloku dyrektyw do zadanych *adresów URL*. Podobnie jak poprzednio, adresy mogą zawierać symbole wieloznaczne oraz wyrażenia regularne poprzedzone znakiem tyldy. Zgodnie z regułami interpretacji wyrażeń regularnych wprowadzonymi w programie Apache 1.3, symbole (`*`) i (`?`) nie spowodują dopasowania znaku ukośnika (`/`). Argumentami dyrektywy `<LocationMatch>` są wyrażenia regularne nie poprzedzone znakiem tyldy (`~`).

Większość dyrektyw używanych w bloku `<Directory>` może być również stosowana w bloku `<Location>`. Należy jednak pamiętać, że użycie w nim dyrektywy `AllowOverride`, chociaż poprawne z formalnego punktu widzenia, jest pozbawione sensu.

<IfDefine>

```
<IfDefine nazwa>
...
</IfDefine>
```

Dyrektywa `<IfDefine>` pozwala na warunkowe uaktywnienie bloku dyrektyw w przypadku uruchomienia programu Apache z opcją `-D nazwa`[4]. Pozwala to na zamknięcie kilku wariantów konfiguracji w pojedynczym pliku *httpd.conf*. Możliwość ta przydaje się głównie podczas testowania i tworzenia wersji dystrybucyjnych, jest natomiast rzadziej stosowana w przypadku „regularnych" witryn o ustalonej strukturze.

<IfModule>

```
<IfModule [!]nazwa-pliku-modułu>
...
</IfModule>
```

[4] Opcja ta definiuje symbol o zadanej nazwie; zob. też „Opcje wywołania programu Apache" w rozdziale 2. — *przyp. tłum.*

Dyrektywa <IfModule> pozwala na warunkowe uaktywnienie bloku dyrektyw w zależności od tego, czy moduł o danej *nazwie* został dołączony do programu Apache (w trakcie kompilacji bądź też dynamicznie, poprzez załadowanie w trakcie pracy pliku DLL). Poprzedzenie nazwy modułu znakiem wykrzyknika (!) powoduje uaktywnienie bloku w przypadku, gdy moduł *nie został* dołączony. Bloki ograniczone dyrektywami <IfModule> mogą być zagnieżdżane. Parametr *nazwa-pliku-modułu* powinien odpowiadać nazwie pliku źródłowego modułu, na przykład *mod_log_conf.c*.

Pozostałe dyrektywy

Pozostało nam do omówienia jeszcze kilka dyrektyw o charakterze administracyjnym.

ServerName

```
ServerName nazwa-domenowa
Zastosowanie: konfiguracja główna, serwery wirtualne
```

Dyrektywa ServerName definiuje nazwę domenową serwera używaną w adresach URL wykorzystywanych do przeadresowywania żądań. Brak dyrektywy spowoduje wykonywanie przez serwer próby samodzielnego określenia nazwy domenowej serwera na podstawie własnego adresu IP; jednakże metoda ta może nie zadziałać lub spowodować wybranie nazwy innej niż preferowana *nazwa domenowa* węzła. Przykładowo, jeżeli kanoniczna nazwa domenowa serwera to *simple.example.com*, ale docelowo klienci mają się odwoływać do węzła *www.example.com*, dyrektywa ServerName powinna mieć następującą postać:

```
ServerName www.przyklad.com
```

UseCanonicalName

```
UseCanonicalName on|off
Wartość domyślna: on
Zastosowanie: konfiguracja główna, serwery wirtualne, katalogi, pliki .htaccess
```

Dyrektywa ta steruje sposobem tworzenia przez Apache adresów wskazujących „na siebie", co ma miejsce np. w przypadku przeadresowania odwołania adresu *http://www.firma.com/jakis/katalog* do poprawnej formy *http://www.firma.com/jakis/katalog/* (różnica tkwi w końcowym znaku ukośnika). W przypadku włączenia dyrektywy UseCanonicalName (stan domyślny), do przeadresowania zostanie użyta nazwa serwera i numer portu zadane dyrektywami ServerName i Port (ale nie w przypadku serwera Apache 2.0). Jej wyłączenie spowoduje użycie nazwy i numeru portu określonej w oryginalnym żądaniu.

Dyrektywa UseCanonicalName przydaje się między innymi w sytuacji, w której komputery użytkowników należą do tej samej domeny co serwer WWW (ma to miejsce np. w sieciach typu intranet). W takim przypadku użytkownik może odwoływać się do serwera za pomocą nazwy skróconej (np. „www"), oszczędzając sobie w ten sposób wpi-

sywania pełnej nazwy domenowej (np. *www.firma.com*). Jeśli dyrektywa `UseCanonicalName` jest aktywna, podanie przez użytkownika adresu bez końcowego ukośnika (np. *http://www/katalog*) spowoduje przeadresowanie żądania do adresu URL *http://www.firma.com/katalog/*, podczas gdy w przypadku wyłączenia dyrektywy żądanie trafiłoby pod adres *http://www/katalog/*. Ma to oczywiste zastosowanie w sytuacji korzystania dostępu autoryzowanego — ponowne użycie nazwy serwera zwalnia użytkownika od konieczności powtórnego przechodzenia przez procedurę autoryzacji, co stałoby się w momencie zorientowania się przez przeglądarkę, że nazwa serwera uległa zmianie. Inne, bardziej złożone zastosowania wiążą się z translacją nazw i adresów związaną z pewnymi aspektami użycia zapór sieciowych.

ServerAdmin

```
ServerAdmin adres-e-mail
Zastosowanie: konfiguracja główna, serwery wirtualne
```

Dyrektywa `ServerAdmin` pozwala zdefiniować adres poczty elektronicznej umieszczany automatycznie przez Apache na stronach generowanych w przypadku wystąpienia błędu. Warto użyć w tym celu dedykowanego adresu, np. *problemy-WWW@butterthlies.com.pl*.

ServerSignature

```
ServerSignature off|on|email
Wartość domyślna: off
Zastosowanie: katalogi, pliki .htaccess
```

Dyrektywa `ServerSignature` umożliwia zidentyfikowanie serwera, który faktycznie obsłużył dane żądanie (co ma znaczenie np. w przypadku użycia serwerów pośredniczących). Jej włączenie (`ServerSignature on`) powoduje automatyczne dołączanie do generowanych przez serwer dokumentów stopki (sygnatury), zawierającej numer wersji programu serwera i nazwę serwera wirtualnego zdefiniowaną w dyrektywie `ServerName`. Użycie formy `ServerSignature email` dodaje do stopki łącze `mailto:` zawierające adres określony dyrektywą `ServerAdmin`.

ServerTokens

```
ServerTokens productonly|min[imal]|OS|full
Wartość domyślna: full
Zastosowanie: konfiguracja główna
```

Dyrektywa `ServerTokens` ustala zakres informacji, jakimi „przedstawia się" serwer Apache. Administrator troszczący się o bezpieczeństwo serwera może ograniczyć ilość informacji o serwerze, jaka może trafić do potencjalnych włamywaczy.

`productonly` (od wersji 1.3.14)
: Serwer zwraca wyłącznie nazwę `Apache`.

`min[imal]`
: Serwer zwraca tylko swoją nazwę i numer wersji, np. `Apache v1.3`.

`OS`
: Serwer zwraca swoją nazwę i numer wersji oraz nazwę macierzystego systemu operacyjnego, np. `Apache v1.3 (Unix)`.

`full`
: Serwer zwraca dane opisane powyżej oraz informacje o wchodzących w skład jego kodu modułach, np. `Apache 1.3 (Unix) PHP/3.0 MojModul/1.2`.

ServerAlias

```
ServerAlias nazwa1 nazwa2 nazwa3 ...
Zastosowanie: serwery wirtualne
```

Dyrektywa `ServerAlias` umożliwia zdefiniowanie listy aliasów, czyli synonimów nazw identyfikujących dany serwer wirtualny. Protokół HTTP/1.1 umożliwia odwołanie się do serwera poprzez nazwę za pomocą pola `Host:` nagłówka HTTP. Podana w nagłówku nazwa powinna odpowiadać którejś z nazw zdefiniowanych w dyrektywach `ServerName`, `ServerAlias` lub `VirtualHost`.

ServerPath

```
ServerPath ścieżka
Zastosowanie: serwery wirtualne
```

Protokół HTTP/1.1 umożliwia związanie z tym samym adresem IP kilku nazw domenowych serwerów. W takiej sytuacji klient identyfikuje odpowiedni serwer poprzez przesłanie w nagłówku żądania pola `Host:` *nazwa-serwera*. Chociaż migracja do protokołu HTTP została już prawie zakończona, niektóre przeglądarki będą zapewne jeszcze przez jakiś czas używały protokołu HTTP/1.0, nie przesyłając pól `Host`[5]. Z tego też względu stworzono dyrektywę `ServerPath` pozwalającą na dostęp do witryny poprzez podanie ścieżki.

Należy tu podkreślić, że użycie dyrektywy `ServerPath` jest nieco kłopotliwe, wymaga bowiem ogromnej dyscypliny w kwestii spójności zapisu łączy używanych wewnątrz witryny. Wszystkie one muszą być zapisywane w postaci względnej, tylko wtedy bowiem będą prawidłowo współpracowały z różnymi adresami URL. Jeśli jednak musisz zapewnić w witrynie obsługę przeglądarek nie korzystających z protokołu HTTP/1.1 i nie przesyłających pól `Host`, w zasadzie nie masz wyboru.

Załóżmy, dla przykładu, że utworzyliśmy dwie witryny o nazwach *www.firma.com* i *sklep.firma.com*, przypisując im obu ten sam adres IP, np. `192.168.123.2`. Nasz plik *httpd.conf* wygląda tak:

```
<VirtualHost 192.168.123.2>
ServerName www.firma.com
DocumentRoot /usr/www/firma
```

[5] Z drugiej strony warto zauważyć, że ów okres przejściowy w znacznym stopniu zakończył się jeszcze przed wprowadzeniem protokołu HTTP/1.1 — wiele przeglądarek wysyłało pola `Host:` również w przypadku korzystania z wersji 1.0. Tym niemniej w sporadycznych przypadkach omawiana dyrektywa może okazać się przydatna.

```
ServerPath /firma
</VirtualHost>

<VirtualHost 192.168.123.2>
ServerName sklep.firma.com
DocumentRoot /usr/www/sklep
ServerPath /sklep
</VirtualHost>
```

Przeglądarka korzystająca z protokołu HTTP/1.1 może w takiej sytuacji rozróżnić obie witryny, przesyłając po prostu adresy URL *http://www.firma.com/* i *http://sklep.firma.com/*. Ponieważ protokół HTTP/1.0 pozwala na rozróżnianie witryn wyłącznie na podstawie adresów IP, oba powyższe adresy URL będą w tym przypadku tożsame. Przeglądarki korzystające z protokołu HTTP/1.0 mogą jednak odwoływać się do obu witryn indywidualnie, podając adresy *http://www.firma.com/firma* i *http://www.firma.com/sklep*. Warto zauważyć, że ten sam efekt miałyby odwołania do adresów URL *http://www.sklep.com/firma* i *http://www.sklep.com/sklep*, bowiem obu występującym w nich nazwom domenowym odpowiada ten sam adres IP (a zatem z punktu widzenia protokołu HTTP/1.0 są one identyczne).

ScoreBoardFile

```
ScoreBoardFile nazwa-pliku
Wartość domyślna: logs/apache_status
Zastosowanie: konfiguracja główna
```

Dyrektywa `ScoreBoardFile` jest w niektórych systemach wymagana do poprawnego utworzenia pliku tymczasowego, wykorzystywanego przez serwer do zarządzania procesami potomnymi (tzw. plik tablicy procesów potomnych, *scoreboard file*). Najprostszą metodą przekonania się, czy w danym systemie plik taki jest wymagany, jest uruchomienie programu Apache i sprawdzenie, czy w wyniku tego działania został utworzony plik o nazwie zadanej w dyrektywie. Jeśli okaże się, że plik taki jest niezbędny, należy podjąć kroki w celu zapewnienia, by nie był on równocześnie używany przez więcej niż jeden proces główny serwera Apache.

W przypadku konieczności użycia pliku tablicy procesów potomnych można spróbować umieścić go na dysku pamięciowym (wirtualnym, *RAM disk*). Powinno to poprawić osiągi serwera, jednak wiąże się z ryzykiem utraty danych.

UNIX Używając serwera Apache w systemach Linux 1.x i Unix System V Release 4, można próbować zablokować użycie pliku tablicy procesów potomnych, dołączając opcje `-DHAVE_SHMGET -DUSE_SHMGET_SCOREBOARD` do zmiennej `EXTRA_CFLAGS` w pliku konfiguracyjnym kompilacji serwera *Configuration*. Rozwiązanie to powinno działać przynajmniej w części konfiguracji Linuksa 1.x (w wersjach Apache wcześniejszych od 1.3b4 wystarczało użycie opcji `HAVE_SHMGET`).

UNIX

CoreDumpDirectory

```
CoreDumpDirectory katalog
Wartość domyślna: <ServerRoot>
Zastosowanie: konfiguracja główna
```

Dyrektywa ta określa położenie *katalogu*, w którym w razie awarii Apache zapisze plik zrzutu pamięci procesu (ang. *core dump file*). Plik ten można następnie poddać analizie za pomocą debugera. Domyślnie plik ten powinien być zapisywany w katalogu określonym dyrektywą `ServerRoot`, jednak przypisane mu prawa dostępu nie umożliwiają zapisywania tam danych przez proces serwera uruchomiony przez zwykłego użytkownika. Użycie dyrektywy `CoreDumpDirectory` ma sens tylko w Uniksie, gdyż Windows nie obsługuje funkcji zrzutu pamięci w przypadku załamania systemu.

SendBufferSize

```
SendBufferSize liczba
Wartość domyślna: ustalana przez system operacyjny
Zastosowanie: konfiguracja główna
```

Dyrektywa ta pozwala powiększyć wielkość bufora transmisji używanego przez procedury obsługi protokołu TCP/IP ponad domyślną wartość określaną przez system operacyjny. W specyficznych przypadkach pozwala to na poprawę wydajności, jednak nie radzimy Ci zmieniać domyślnego ustawienia parametrów TCP/IP, o ile nie znasz się na rzeczy naprawdę dobrze.

UNIX

LockFile

```
LockFile plik
Wartość domyślna: logs/accept.lock
Zastosowanie: konfiguracja główna
```

Jeśli Apache zostanie skompilowany z opcją `USE_FCNTL_SERIALIZED_ACCEPT` lub `USE_FLOCK_SERIALIZED_ACCEPT`, przed uruchomieniem musi on zapisać na dysku lokalnym plik blokady (ang. *lock file*). Operacja ta może okazać się niemożliwa w przypadku umieszczenia katalogu *logs* w woluminie NFS. Nie jest również wskazane umieszczanie pliku blokady w katalogu dostępnym dla wszystkich do zapisu, gdyż jego przypadkowe utworzenie zablokuje możliwość uruchomienia serwera. Mechanizm blokady jest niezbędny w niektórych systemach operacyjnych, które nie tolerują równoczesnego wywołania funkcji `accept()` przez kilka procesów dla pojedynczego gniazda (to właśnie w niej „siedzi" Apache w trakcie oczekiwania na połączenie). Z tego też względu konieczne jest użycie jakiegoś mechanizmu szeregowania wywołań funkcji `accept()`. Można tego dokonać poprzez użycie pliku blokady, jednak rozwiązanie to nie daje się zastosować w przypadku użycia systemu NFS.

AcceptMutex

```
AcceptMutex default|metoda
Wartość domyślna: default
Zastosowanie: Konfiguracja główna
```

Dyrektywy `AcceptMutex` służą do ustalenia metody wykorzystywanej następnie przez Apache do szeregowania wielokrotnych wywołań funkcji `accept()` w stosunku do gniazda sieciowego, inicjowanych przez procesy potomne serwera. W wersjach poprzedzających Apache 2.0 metodę szeregowania można było określić wyłącznie na etapie

kompilacji. Wybór optymalnej metody uzależniony jest oczywiście od cech systemu operacyjnego i architektury serwera. Szczegółowe informacje na ten temat można znaleźć pod adresem *http://httpd.apache.org/docs-2.0/misc/perf-tuning.html*.

W przypadku rezygnacji z określenia wartości dyrektywy `AcceptMutex` lub ustawienia jej na wartość domyślną, serwer skorzysta z ustawień określonych w trakcie kompilacji. Pozostałe dostępne metody zostały wyliczone poniżej. Warto odnotować, że nie wszystkie metody są odpowiednie dla każdej platformy. Jeżeli dana metoda nie jest na konkretnej platformie obsługiwana, próba uruchomienia serwera spowoduje zapisanie w pliku dziennika błędów odpowiedniego komunikatu i listy dostępnych metod szeregowania.

`flock`
: Synchronizacja dostępu do pliku określonego wartością dyrektywy `LockFile` realizowana jest za pośrednictwem systemowego wywołania `flock()`.

`fcntl`
: Synchronizacja dostępu do pliku określonego wartością dyrektywy `LockFile` realizowana jest za pośrednictwem systemowego wywołania `fcntl()`.

`sysvsem`
: Synchronizacja dostępu do pliku realizowana jest przy użyciu semaforów zgodnych ze specyfikacją *System V*.

`pthread`
: Synchronizacja dostępu do pliku realizowana jest przy użyciu semaforów zgodnych ze specyfikacją POSIX Threads (*PThreads*).

KeepAlive

```
KeepAlive liczba
Wartość domyślna: 5
Zastosowanie: konfiguracja główna
```

Skoro dany użytkownik raz odwołał się do witryny, istnieją spore szanse, że za chwilę uczyni to ponownie. Minimalizację niepożądanych opóźnień można uzyskać poprzez podtrzymanie otwartego połączenia, jednak liczbę odwołań w trakcie tak wydłużonego połączenia warto ograniczyć, aby zapobiec nadmiernej konsumpcji zasobów serwera. Ustalenie dopuszczalnej liczby odwołań realizowane jest za pomocą dyrektywy `KeepAlive`; wartość domyślna, równa 5, może zostać powiększona w przypadku, gdy struktura witryny zawiera więcej poziomów drzewa katalogów. Warto wspomnieć, że przeglądarka Netscape Navigator 2 zawierała błąd zakłócający działanie podtrzymywania połączeń; począwszy od wersji 1.2, Apache automatycznie wykrywa użycie tej przeglądarki, lokalizując w nagłówkach odebranych żądań ciąg `Mozilla/2`. Problem można również wyeliminować poprzez użycie dyrektywy `BrowserMatch` (zobacz rozdział 13.).

KeepAliveTimeout

```
KeepAliveTimeout czas-w-sekundach
Wartość domyślna: 15
Zastosowanie: konfiguracja główna
```

Dyrektywa ta pozwala na ograniczenie czasu oczekiwania na kolejne żądanie poprzez określenie limitu czasu (w sekundach), przez który połączenie będzie podtrzymywane w stanie otwarcia. W chwili odebrania żądania (przed upływem limitu) uaktywniona zostaje dyrektywa `TimeOut`.

TimeOut

```
TimeOut czas-w-sekundach
Wartość domyślna: 300[6]
Zastosowanie: konfiguracja główna
```

Dyrektywa ta określa maksymalną długość czasu transmisji pojedynczego bloku danych w trakcie transakcji. We wcześniejszych wersjach Apache dyrektywa ta miała dość nieprzyjemne efekty uboczne, powodowała bowiem błędy przeterminowania w przypadku transmisji dużych plików łączami o niewielkiej przepustowości. W związku z tym tak zmieniono jej działanie, by określała nie maksymalną długość trwania całej transakcji, ale czas przeznaczony na przesłanie pojedynczego bloku danych[7].

HostNameLookups

```
HostNameLookups on|off|double
Wartość domyślna: off[8]
Zastosowanie: konfiguracja główna, serwery wirtualne, katalogi
```

Uaktywnienie tej dyrektywy poprzez użycie argumentu `on` powoduje, że każde odebrane żądanie poddawane jest operacji odwrotnego odwzorowania adresu (ang. *reverse DNS resolution*). Oznacza to, że wydzielony z żądania adres IP klienta jest wykorzystywany do ustalenia jego nazwy domenowej, pobieranej z odpowiedniego serwera DNS. Tak uzyskana nazwa domenowa jest następnie rejestrowana w plikach dzienników. Wyłączenie dyrektywy (użycie argumentu `off`) powoduje użycie w tym miejscu adresu IP. Ponieważ procedura odwrotnego odwzorowania adresu potrafi być bardzo czasochłonna, dla poprawy wydajności (zwłaszcza w bardziej obciążonych systemach) lepiej jest zablokować tę funkcję, ustawiając dyrektywę `HostNameLookups` na `off`. Warto tu wspomnieć, iż pakiet Apache zawiera program narzędziowy o nazwie *logresolve*, pozwalający na dokonanie operacji odwrotnego rozwinięcia adresów już zapisanych w plikach dzienników[9].

Począwszy od wersji 1.3, dyrektywa `HostNameLookups` może również przyjmować wartość `double`. Jej użycie powoduje wykonanie dwukrotnego odwrotnego odwzorowania adresu (*double reverse DNS lookup*), czyli powtórnego przetworzenia na adres IP

[6] W wersjach Apache wcześniejszych niż 1.2 wartość ta wynosiła 1200 sekund — *przyp. tłum.*

[7] Ewentualnie pomiędzy odebraniem kolejnych pakietów TCP — *przyp. tłum.*

[8] W wersjach Apache wcześniejszych niż 1.3 wartość ta wynosiła `on`, o czym warto pamiętać podczas aktualizacji serwera.

[9] W przypadku adresów IP przydzielanych dynamicznie nie ma gwarancji, iż ich odwrotne odwzorowanie w terminie późniejszym pozwoli ustalić nazwę domenową, która faktycznie odpowiadała im w momencie połączenia. Jeśli znajomość nazwy domenowej klienta jest istotna, należy włączyć dyrektywę `HostNameLookups`.

nazwy domenowej uzyskanej z odwzorowania DNS oryginalnego adresu. Zgodność obu adresów oznacza pozytywny wynik testu. Niezależnie od ustawienia dyrektywy `HostNameLookups`, w przypadku implementacji kontroli dostępu poprzez moduł *mod_access* i listy dostępu, każdy klient żądający dostępu do zasobów serwera musi pomyślnie przejść test dwukrotnego odwrotnego odwzorowania adresu.

Include

```
Include plik
Zastosowanie: konfiguracja główna
```

Dyrektywa ta powoduje wstawienie zawartości *pliku* w miejscu jej wystąpienia w pliku konfiguracji serwera. Począwszy od wersji 1.3.14, jeżeli *plik* wskazuje katalog, do pliku konfiguracyjnego włączana jest zawartość wszystkich plików znajdujących się w katalogu i jego podkatalogach.

Limit

```
<Limit metoda1 metoda2 ...>
...
</Limit>
```

Dyrektywa blokowa `<Limit metoda>` definiuje blok zawierający dyrektywy adekwatne dla poszczególnych metod nadchodzących żądań HTTP. Przykładowo:

```
<Limit GET POST>
... dyrektywy ...
</Limit>
```

spowoduje ograniczenie zastosowania dyrektyw zdefiniowanych wewnątrz bloku `<Limit>` do tych żądań, które odwoływały się do metod `GET` lub `POST` protokołu HTTP. Zwykle kontrola dostępu realizowana jest identycznie dla wszystkich metod protokołu HTTP. Jednak w ogólnym przypadku dyrektywy kontroli dostępu nie powinno się umieszczać wewnątrz dyrektywy blokowej `<Limit>`.

Dyrektywa `<Limit>` służy głównie do nakładania ograniczeń związanych z kontrolą dostępu i wpływających na realizację żądań zawierających wskazane w nagłówku bloku metody protokołu HTTP. Żądania nie zawierające wymienionych tam metod nie podlegają żadnym ograniczeniom. Ograniczenia zdefiniowane wewnątrz dyrektywy grupowej `<Limit>` nie wpływają w żaden sposób na realizację żądań nie zawierających wskazanych metod. Poniższy przykład wyznacza sposób kontrolowania żądań `POST`, `PUT` i `DELETE`, nie wpływając na obsługę żądań odwołujących się do pozostałych metod HTTP:

```
<Limit POST PUT DELETE>
Require valid-user
</Limit>
```

Nazwy metod, wymieniane w nagłówku bloku `<Limit>`, muszą należeć do zbioru `GET`, `POST`, `PUT`, `DELETE`, `CONNECT`, `OPTIONS`, `TRACE`, `PATCH`, `PROPFIND`, `PROPPATCH`, `MKCOL`, `COPY`, `MOVE`, `LOCK` i `UNLOCK`. Wielkość liter w nazwie metody jest istotna. Wyspecyfikowanie w bloku `<Limit>` metody `GET` powoduje objęcie restrykcjami dostępu zdefiniowanymi w tym bloku również żądań `HEAD`.

Zasadniczo nie zaleca się stosowania dyrektywy <Limit>, jeśli nie jest to bezwzględnie potrzebne (na przykład w przypadku, kiedy konfiguracja serwera obejmuje implementację metody PUT, która jednak z oczywistych względów powinna podlegać ściślejszym ograniczeniom niż metoda GET) i obeszliśmy się bez niej w przykładzie *site.authent*. Niestety, dostępna w sieci dokumentacja serwera Apache zachęca do niewłaściwego stosowania tej dyrektywy, więc jest ona często nadużywana.

<LimitExcept>

```
<LimitExcept metoda1 metoda2 ...>
...
</LimitExcept>
```

Blok utworzony dyrektywami <LimitExcept> i </LimitExcept> służy do wydzielania tych dyrektyw kontroli dostępu, które stosowane będą wobec żądań zawierających metody HTTP nie wymienione w nagłówku bloku. Jest to dyrektywa komplementarna wobec <Limit> i może zawierać dyrektywy aplikowane zarówno do standardowych, jak i niestandardowych (lub nierozpoznanych) metod. Patrz opis dyrektywy <Limit>.

LimitRequestBody

```
LimitRequestBody liczba-bajtów
Wartość domyślna: 0
Zastosowanie: konfiguracja główna, serwery wirtualne, katalog, pliki .htaccess
```

Za pomocą wartości definiowanej w ramach dyrektywy LimitRequestBody możliwe jest ograniczenie maksymalnego rozmiaru żądania (0 oznacza brak limitu; maksymalna dopuszczalna wartość to 2 147 483 647). Wartość domyślna jest ustalana podczas kompilacji serwera za pośrednictwem stałej DEFAULT_LIMIT_REQUEST_BODY.

Głównym zastosowaniem tej dyrektywy jest ograniczanie maksymalnego rozmiaru komunikatu przekazywanego w ciele żądania HTTP; ograniczenie realizowane jest wyłącznie w ramach kontekstu, w którym występuje dyrektywa (czyli w kontekście serwera, katalogu, pliku bądź lokalizacji). Jeżeli żądanie przekroczy ustalony rozmiar, serwer, zamiast zrealizować żądanie, odeśle do klienta komunikat z informacją o błędzie. Dopuszczalny rozmiar komunikatu jest uzależniony głównie od natury zasobu, do którego odwołuje się żądanie i metody związanej z obsługą tego zasobu. Z ciała komunikatu korzystają intensywnie skrypty CGI, przekazując do serwera informacje przesłane w formularzu. Implementacje metody PUT wymagają z kolei ustalenia limitu rozmiaru komunikatu na poziomie przewyższającym rozmiar komunikatów zawierających zasoby przesyłane do serwera w ramach metody PUT.

Dyrektywa ta daje administratorowi serwera możliwość sterowania obsługą ponadnormatywnych żądań klientów, co może okazać się przydatne w unikaniu niektórych form ataku odmowy obsługi.

LimitRequestFields

```
LimitRequestFields liczba
Wartość domyślna: 100
Zastosowanie: konfiguracja główna
```

Liczba jest liczbą całkowitą, z zakresu od 0 (co oznacza brak ograniczenia) do 32 767.Wartość domyślna (100) ustalana jest podczas kompilacji za pośrednictwem stałej DEFAULT_REQUEST_LIMIT_FIELDS.

Dyrektywa LimitRequestFields daje administratorowi serwera WWW możliwość sterowania maksymalną dopuszczalną liczbą nagłówków w żądaniach HTTP. Oczywiście limit musi być większy od liczby nagłówków przesyłanych przez zwykłych klientów w czasie normalnej działalności witryny. W praktyce liczba ta rzadko przekracza 20, ale może być większa i zależy głównie od implementacji przeglądarki oraz od stopnia dostosowania jej przez użytkownika do własnych wymagań i związaną z tym obsługą negocjacji zawartości komunikatów HTTP. Za pomocą pól nagłówka protokołu HTTP określane są też często najróżniejsze rozszerzenia protokołu.

Administrator serwera może za pośrednictwem tej dyrektywy kontrolować obsługę ponadnormatywnych żądań klientów, w tym żądań nadsyłanych w ramach ataku odmowy obsługi. W razie otrzymywania od zwykłych użytkowników sygnałów o pojawianiu się w przeglądarce komunikatów informujących o zbyt dużej liczbie pól w żądaniu, wartość LimitRequestFields powinna zostać zwiększona.

LimitRequestFieldSize

```
LimitRequestFieldSize liczba-bajtów
Wartość domyślna: 8190
Zastosowanie: konfiguracja główna
```

Dyrektywa ta określa maksymalny rozmiar pojedynczego pola nagłówka żądania HTTP jako *liczbę-bajtów* z zakresu od 0 do wartości ustalonej podczas kompilacji (za pośrednictwem stałej DEFAULT_LIMIT_REQUEST_FIELD_SIZE).

Administrator serwera WWW może za pomocą tej dyrektywy zmniejszyć dopuszczalny rozmiar pól nagłówka żądań HTTP napływających do serwera tak, aby ich rozmiar był mniejszy od rozmiaru bufora wejściowego, którego wielkość jest ustalana na etapie kompilacji. Wartość ta nie może być zbyt mała, aby możliwa była poprawna obsługa zwykłych żądań HTTP przesyłanych do serwera w ramach normalnych odwiedzin witryny. Rozmiar pól nagłówków żądania HTTP zależy głównie od implementacji przeglądarki, w tym od stopnia jej dostosowania do potrzeb użytkownika w sensie intensywności korzystania z funkcji negocjacji protokołu HTTP.

Dyrektywa ta daje administratorowi serwera możliwość sterowania obsługą ponadnormatywnych żądań klientów, co może okazać się przydatne w zapobieganiu niektórym formom ataku odmowy obsługi. W normalnych warunkach nie ma potrzeby zmiany wartości domyślnej.

LimitRequestLine

```
LimitRequestLine liczba-bajtów
Wartość domyślna: 8190
Zastosowanie: konfiguracja główna
```

Liczba-bajtów jest liczbą całkowitą, z zakresu od 0 (co oznacza brak ograniczenia) do 8 190 (wartość ta jest ustalana podczas kompilacji na podstawie wartości stałej DEFAULT_LIMIT_REQUEST_LINE i wynosi zwykle 8 190).

Dyrektywa LimitRequestFields daje administratorowi serwera WWW sposobność ograniczenia maksymalnego dopuszczalnego rozmiaru poszczególnych wierszy żądania HTTP. Rozmiar ten powinien być dostosowany do rozmiaru bufora wejściowego serwera. Wiersze żądania zawierają metodę HTTP, URI i numer protokołu, dyrektywa LimitRequestLine ma oczywiście na celu ograniczenie rozmiaru URI przekazywanego do serwera. Wartość ta musi być na tyle duża, aby serwer nie odrzucał poprawnych żądań odwołujących się do udostępnianych przez serwer zasobów, przy uwzględnieniu narzutu rozmiaru identyfikatora URI związanego z przechowywaniem parametrów metody GET.

Administrator serwera może za pośrednictwem tej dyrektywy kontrolować obsługę ponadnormatywnych żądań klientów, w tym żądań nadsyłanych w ramach ataku odmowy obsługi. W normalnych warunkach nie ma potrzeby modyfikowania wartości domyślnej.

Nagłówki odpowiedzi HTTP

Administrator serwera WWW może samodzielnie ustawiać treść nagłówków HTTP odpowiedzi odsyłanych klientom, tak aby przystosować działanie serwera do specjalnych warunków działania witryny, udostępnić metainformacje wyszukiwarkom i robotom indeksującym oraz przekazywać etykiety klasyfikujące standardu PICS. Warto jednak pamiętać, że Apache nie sprawdza poprawności i zasadności określonych w ten sposób nagłówków, dlatego każdorazowo konieczne jest samodzielne upewnienie się, czy dany nagłówek nie powoduje czasem u klienta niespodziewanych i niepożądanych efektów.

Header

```
Header set|append|add nagłówek "wartość"
Header unset nagłówek
Zastosowanie: konfiguracja główna, serwery wirtualne, pliki access.conf
i .htaccess
```

Dyrektywa ta pozwala na zastąpienie, dołączenie bądź usunięcie nagłówków HTTP odpowiedzi serwera. Rodzaj działania w stosunku do nagłówka jest określany pierwszym parametrem dyrektywy. Może on przyjąć jedną z następujących wartości:

`set`
: Nagłówek odpowiedzi jest ustawiany. Jeśli wcześniej zdefiniowany został nagłówek o identycznej nazwie, zostanie on zastąpiony nagłówkiem definiowanym bieżąco.

`append`
: Wartość nagłówka odpowiedzi jest dołączana do wartości nagłówka o tej samej nazwie zdefiniowanego wcześniej. Nowa wartość oddzielana jest od poprzedniej przecinkiem. Jest to standardowa metoda definiowania wielowartościowych nagłówków HTTP.

`add`
: Nagłówek odpowiedzi jest dodawany do istniejącego zbioru nagłówków, nawet jeśli nagłówek o tej samej nazwie już występuje w zbiorze nagłówków. Tak więc w wyniku stosowania tej opcji możliwe jest utworzenie odpowiedzi zawierającej kilka identycznych nagłówków o różnych wartościach. Konsekwencje interpretacji takiej odpowiedzi przez klienta są nieokreślone, dlatego zaleca się stosowanie zamiast opcji `add` opcji `append`.

`unset`
: Nagłówek odpowiedzi o wskazanej nazwie jest usuwany z odpowiedzi. Jeśli w odpowiedzi zdefiniowano kilka nagłówków o tej samej nazwie, każdy z nich zostanie usunięty ze zbioru nagłówków odpowiedzi.

Po opcji definiującej rodzaj operacji na zbiorze nagłówków podawana jest nazwa nagłówka (może ona zawierać tradycyjny znak dwukropka, ale nie jest to wymagane). Wielkość liter parametrów dyrektywy nie ma znaczenia. W przypadku opcji `add`, `append` i `set` wymagane jest określenie trzeciego parametru, interpretowanego jako wartość nagłówka. Jeśli wartość ta zawiera znaki spacji, powinna zostać ujęta w znaki cudzysłowu. Nie można podać wartości nagłówka w przypadku opcji `unset`.

Kolejność przetwarzania

Dyrektywa `Header` może być definiowana w niemal dowolnym miejscu pliku konfiguracyjnego, zarówno w kontekście konfiguracji głównej, w blokach dyrektyw dla serwerów wirtualnych, wewnątrz bloków `<Directory>`, `<Location>` i `<Files>` oraz wewnątrz plików *.htaccess*.

Kolejność przetwarzania dyrektyw `Header` jest następująca:

- konfiguracja główna;
- serwer wirtualny;
- bloki `<Directory>` i pliki *.htaccess*;
- `<Location>`;
- `<Files>`.

Istotna jest kolejność występowania dyrektyw. Poniższy przykład ilustruje zniesienie zdefiniowanej wcześniej dyrektywy:

```
Header append Author "Jan Nowak"
Header unset Author
```

Efektem przetworzenia obu dyrektyw jest usunięcie ze zbioru nagłówków odpowiedzi nagłówka `Author:`. Gdyby zamienić dyrektywy miejscami, zbiór nagłówków odpowiedzi zawierałby nagłówek `Author:` o wartości `"Jan Nowak"`.

Dyrektywy `Header` są jeszcze przed wysłaniem odpowiedzi przetwarzane przez przypisane do nich procedury obsługi. Oznacza to, że niektóre z nagłówków, dodane bezpośrednio przed odesłaniem odpowiedzi do klienta, nie mogą zostać usunięte czy nadpisane. Dotyczy to między innymi nagłówków `Date:` i `Server:`.

Options

```
Options opcja opcja ...
Wartość domyślna: All
Zastosowanie: konfiguracja główna, serwery wirtualne, katalogi, pliki
.htaccess
```

Dyrektywa `Options` ma niezwykle szerokie zastosowanie i nie daje się przypisać do pojedynczego kontekstu, dlatego zostanie omówiona szerzej niż inne. Administrator WWW dysponuje za pośrednictwem dyrektywy `Options` daleko idącą kontrolą nad możliwościami, jakimi dysponować będą właściciele poszczególnych witryn obsługiwanych przez serwer. Parametry *`opcja`* mogą zostać zastąpione pojedynczym ciągiem `None`, co spowoduje zablokowanie wszelkich dodatkowych funkcji serwera, mogą też przyjąć jedną z następujących wartości:

`All`
: Uaktywnione zostaną wszystkie opcje serwera z wyjątkiem opcji `MultiViews` (dla zachowania zgodności z wersjami wcześniejszymi).

`ExecCGI`
: Brak określenia tej opcji uniemożliwia wykonywanie skryptów CGI.

`FollowSymLinks`
: Określenie tej opcji pozwala serwerowi rozwijać dowiązania symboliczne znajdujące się w danym katalogu.



Rozwijanie przez serwer dowiązań symbolicznych nie wyłącza plików, do których klient odwołuje się za pośrednictwem dowiązań, spod rygorów dyrektyw zdefiniowanych w bloku `<Directory>`, jeśli wyrażenie zdefiniowane w nagłówku tego bloku obejmuje katalog, w którym znajdują się te dowiązania. Wewnątrz sekcji `<Location>` opcja ta jest ignorowana.

`Includes`
: Brak określenia tej opcji uniemożliwia przetwarzanie przez serwer poleceń wstawianych SSI.

`IncludesNOEXEC`
: Określenie tej opcji oznacza przyzwolenie na przetwarzanie poleceń wstawianych, z wyjątkiem poleceń `exec` i `include` w odniesieniu do skryptów CGI.

`Indexes`
: Jeżeli klient odwoła się w żądaniu do URL-a, który zostanie odwzorowany do katalogu, w którym nie ma pliku *index.html*, określenie tej opcji pozwoli na zastosowanie zestawu poleceń indeksowania zwracających sformatowany wydruk zawartości katalogu.

`MultiViews`
: Uaktywnia obsługę *wielowidokowej treści dokumentów*. Negocjacja zawartości dokumentów obejmuje m.in. wybór wersji językowej (dyrektywa `AddLanguage`) i formatu plików graficznych. Szersze omówienie tego tematu zawiera rozdział 6.

`SymLinksIfOwnerMatch`
: Serwer korzysta z rozwinięć dowiązań symbolicznych wyłącznie w przypadku, kiedy plik uzyskany po rozwinięciu dowiązania należy do użytkownika, do którego należy również samo dowiązanie.



Opcja ta jest ignorowana wewnątrz bloku `<Location>`.

Poszczególne opcje mogą być poprzedzane znakami (+) lub (-), oznaczającymi odpowiednio: uaktywnienie bądź zablokowanie opcji. Zgodnie z tym, poniższa dyrektywa spowoduje uaktywnienie indeksowania, ale zablokuje możliwość wykonywania skryptów CGI:

```
Options +Indexes -ExecCGI
```

Jeśli w dyrektywie nie zostaną określone żadne opcje, dyrektywa jest interpretowana tak, jakby towarzyszył jej specyfikator `All` (co jednak nie oznacza uaktywnienia funkcji `MultiViews`). Określenie dowolnej opcji znosi domyślne założenie specyfikatora `All`.

Powoduje to przynajmniej jeden, niepożądany efekt, który zademonstrujemy na przykładzie pliku *…/site.options*. Ilustracja tego efektu wymaga też nieznacznej modyfikacji skryptu `go`. Jego nowa zawartość to:

```
test -d logs || mkdir logs
httpd -f 'pwd'/conf/httpd$1.conf -d'pwd'
```

Załóżmy, że w katalogu *…/htdocs* znajduje się katalog pozbawiony pliku *index.html* i prosty plik konfiguracyjny serwera o następującej zawartości:

```
User webuser
Group webgroup
ServerName www.butterthlies.com.pl
DocumentRoot /usr/www/APACHE3/site.options/htdocs
```

Po uruchomieniu serwera (jak zwykle poleceniem `./go`) i odwołaniu się do niego za pośrednictwem przeglądarki WWW zobaczymy indeks katalogu *…/htdocs*. Jeśli teraz skopiujemy plik konfiguracyjny do pliku *…/conf/httpd1.conf* i dodamy do niego wiersz:

```
Options ExecCGI
```

przeładujmy serwer, zatrzymując go i ponownie uruchamiając za pośrednictwem polecenia `./go 1`. Po ponownym odwołaniu się do katalogu w oknie przeglądarki pojawi się dość nieprzyjemny komunikat:

```
FORBIDDEN
You don't have permissions to access / on this serwer
```

(lub podobny; jego treść uzależniona jest od konfiguracji przeglądarki). Przyczyną odmowy obsługi żądania jest to, że kiedy w pliku konfiguracyjnym nie określi się dyrektywy `Options`, domyślnie przyjmowana jest dyrektywa `Options All`. Natomiast po jawnym zadeklarowaniu opcji `ExecCGI` wszelkie pozostałe opcje zostały po zniesieniu dyrektywy domyślnej zablokowane. Należy wtedy zmienić treść pliku konfiguracyjnego (*…/conf/httpd2.conf*) tak, aby zawierał wiersz:

```
Options +ExecCGI
```

Jeżeli określeniu opcji nie towarzyszy znak (+) lub (–) i w pliku zadeklarowanych jest kilka dyrektyw, serwer bierze pod uwagę ostatnią. Przykładowo (*…/conf/httpd3.conf*):

```
Options ExecCGI
Options Indexes
```

spowoduje uaktywnienie tylko opcji `Indexes`. Skrypty CGI nie będą działać, choć przyczyna tego stanu rzeczy nie jest dla początkujących administratorów oczywista. Podobny efekt występuje też przy wielokrotnym definiowaniu dyrektywy `Options` w osobnych blokach `<Directory>`:

```
<Directory /web/docs>
Options Indexes FollowSymLinks
</Directory>
<Directory /web/docs/specs>
Options Includes
</Directory>
```

Dla katalogu */web/docs/spec* aktywna będzie wyłącznie opcja `Includes`.

Opcje FollowSymLinks, SymLinksIfOwnerMatch

Kiedy wcześniej w tym rozdziale, kierując się oszczędnością przestrzeni dyskowej, zdecydowaliśmy, że zasoby witryny (pliki zdjęć *bench.jpg*, *hen.jpg*, *bath.jpg* i *tree.jpg*) przechowywane będą w katalogu */usr/www/APACHE3/main_docs*, a poszczególne kopie witryny korzystać będą z dowiązań do tych plików, mówiliśmy o dowiązaniach twardych (ang. *hard links*). Korzystanie z takich dowiązań nie zawsze jest jednak najlepszym pomysłem, ponieważ po usunięciu pliku, do którego odwoływały się dowiązania i utworzeniu jego nowej wersji, dowiązania twarde nie będą wskazywać nowej wersji pliku. Inaczej jest w przypadku dowiązania symbolicznego — takie dowiązanie odwoływałoby się do nowej wersji pliku. Dowiązanie symboliczne tworzone jest poleceniem `ln` z opcją `-s`, według szablonu `ln -s nazwa-dowiązania nazwa-pliku-docelowego`.

Jednakże rozwiązanie to nie jest pozbawione wad, naraża bowiem bezpieczeństwo plików różnych użytkowników tego samego systemu. W każdym stadzie znajdzie się czarna owca; serwer WWW nie jest tu wyjątkiem i łatwo sobie wyobrazić, że jednym z jego użytkowników jest podstępny Bolek dysponujący własną przestrzenią WWW, a w niej plikiem *…/bolek/public.html*. Załóżmy, że administrator serwera WWW utworzył skrypt CGI o nazwie *fido* przechowywany w *…/cgi-bin* i należący do użytkownika *webuser*. Jeśli administrator troszczy się o bezpieczeństwo serwera, powinien tak ustawić uprawnienia dostępu do skryptu, aby nie mógł go czytać ani uruchamiać żaden z użytkowników

systemu z wyjątkiem użytkownika *webuser*. Oczywiście klienci korzystający z serwera mogą korzystać z pliku, gdyż odwołują się do niego za pośrednictwem procesu roboczego serwera, również należącego do użytkownika *webuser*. Wydaje się, że Bolek nie może odczytać zawartości z pliku. Taki stan wydaje się więc zgodny z polityką bezpieczeństwa serwera, według której należy ograniczać krąg użytkowników mających dostęp do skryptów CGI. Zmniejsza to ryzyko wykorzystania przez złośliwych użytkowników wiedzy o lukach w tych skryptach.

Niemniej jednak Bolek może podstępnie wykonać dowiązanie symboliczne do skryptu *fido* z własnej przestrzeni WWW. Co prawda z poziomu powłoki systemowej nadal nie będzie mógł odczytać zawartości pliku, ale może się do niego dobrać za pośrednictwem serwera WWW i własnej przestrzeni WWW.

Dyrektywa `Options` bez specyfikatorów `All` i `FollowSymLinks` uniemożliwia ten proceder. Administrator WWW może jednak dopuścić korzystanie z dowiązań symbolicznych (co pozwala zaoszczędzić przestrzeń dyskową), pod warunkiem, że właścicielem pliku docelowego i dowiązania jest ten sam użytkownik, za pośrednictwem opcji `SymLinksIfOwnerMatch`.

Restart serwera

Niekiedy zachodzi potrzeba zatrzymania serwera Apache i ponownego jego uruchomienia z uwzględnieniem nowej zawartości pliku konfiguracji, zwykle po dodaniu do niego definicji nowego serwera wirtualnego lub też po takiego serwera usunięciu. Oczywiście restart można wykonać metodą siłową, unicestwiając proces główny serwera poleceniem `kill <PID>` i ponownie uruchamiając serwer. Tyle że wszelkie realizowane w tym czasie transakcje zostaną zerwane, co może nieco poirytować zalogowanych i korzystających z witryn użytkowników. Ostatnie wersje Apache pozwalają na bardziej eleganckie przeładowanie serwera głównego — bez gwałtownego wyrzucenia z systemu procesów potomnych zajętych obsługą żądań.

UNIX W systemie Unix serwer Apache może zostać przeładowany na trzy sposoby (patrz też rozdział 2.):

- Unicestwienie procesu głównego serwera i uruchomienie go z uwzględnieniem nowej wersji pliku konfiguracyjnego:

```
% kill PID
% httpd [opcje]
```

- Przesłanie do procesu serwera głównego sygnału `-HUP`:

```
% kill -HUP PID
```

- Łagodny restart przez przesłanie do procesu serwera sygnału `-USR1`. Reakcją serwera na otrzymanie takiego sygnału jest ponowny odczyt pliku konfiguracyjnego przy umożliwieniu procesom potomnym dokończenia realizowanych aktualnie żądań i transakcji. Po zakończeniu transakcji proces potomny zastępowany jest nowym

egzemplarzem procesu roboczego serwera, uwzględniającym ową konfigurację. Procedura ta jest odpowiednia w większości przypadków, ponieważ jest mało odczuwalna dla klientów serwera (o ile nowy plik konfiguracyjny nie zawiera błędów):

```
% kill -USR1 PID
```

Skrypt realizujący „elegancką" procedurę przeładowania, przeznaczony do wywoływania z katalogu głównego serwera, miałby następującą postać:

```
#!/bin/sh
kill -USR1 'cat logs/httpd.pid'
```

WIN32 W systemach z rodziny Windows „łagodny" restart można wykonać, otwierając okno wiersza poleceń DOS i wpisując:

```
> apache -k restart
```

Więcej informacji na temat uruchamiania i zatrzymywania procesów serwera znajdziesz w rozdziale 2.

Pliki .htaccess

Metodą alternatywną w stosunku do modyfikowania pliku konfiguracyjnego serwera i jego przeładowywania jest mechanizm wykorzystujący pliki *.htaccess*, omówiony szerzej w rozdziale 5. Polega on na przechowywaniu części konfiguracji serwera (tej, która ulega najczęstszym zmianom) w osobnym pliku, znajdującym się w katalogu *…/htdocs*. W przeciwieństwie bowiem do głównego pliku konfiguracyjnego, plik *.htaccess* jest odczytywany przez serwer nie podczas uruchamiania, ale każdorazowo w ramach obsługi żądania do danej witryny. Zaletą tego podejścia jest elastyczność konfiguracji — administrator może modyfikować działanie serwera w dowolnym momencie, bez przerywania jego pracy. Wadą jest znaczny spadek wydajności serwera — przetwarzanie pliku konfiguracji dla każdego żądania musi spowodować wydłużenie jego obsługi. Dlatego administrator może ograniczyć zmiany wprowadzane przez właścicieli witryn do ich plików *.htaccess*, deklarując w głównym pliku konfiguracyjnym dyrektywę `AllowOverride`.

Administrator może również uniemożliwić klientom podgląd ich własnych plików *.htacces*. Można to osiągnąć, wprowadzając do pliku konfiguracyjnego serwera następujące wiersze:

```
<Files .htaccess>
order allow,deny
deny from all
</Files>
```

Metapliki w standardzie CERN

Metaplik (ang. *metafile*) to plik zawierający dodatkowe nagłówki przesyłane wraz z plikiem transmitowanym przez serwer w odpowiedzi HTTP. Może on zawierać, na przykład, nagłówek `Refresh:`. Jako że omówienie tego mechanizmu trudno wpasować

w tematykę pozostałych rozdziałów książki, zdecydowaliśmy się zamieścić je tu, w rozdziale omawiającym podstawowe aspekty konfiguracji.

MetaFiles

```
MetaFiles on|off
Wartość domyślna: off
Zastosowanie: katalog
```

Włącza (lub wyłącza) przetwarzanie metaplików dla danego katalogu.

MetaDir

```
MetaDir nazwa-katalogu
Wartość domyślna: .web
Zastosowanie: katalog
```

Wartość tej dyrektywy określa katalog, w którym serwer Apache będzie szukał metaplików. Jest to zwykle katalog ukryty (jego nazwa zaczyna się znakiem kropki), będący podkatalogiem katalogu, w którym przechowywany jest udostępniany dokument. Ustawienie dyrektywy `MetaDir` na pojedynczą kropkę (`.`) spowoduje poszukiwanie metapliku w katalogu dokumentu.

MetaSuffix

```
MetaSuffix rozszerzenie
Wartość domyślna: .meta
Zastosowanie: katalog
```

Wartość tej dyrektywy definiuje rozszerzenie, jakie powinien nosić plik zawierający metainformacje.

Jeżeli pozostawiona zostanie wartość domyślna dyrektywy `MetaSuffix`, żądanie odwołujące się do pliku *…/mój-katalog/bolek.html* będzie przetworzone z wykorzystaniem metainformacji przechowywanych w pliku *…/mój-katalog/bolek.html.meta*.

Określanie terminu ważności dokumentu

Począwszy od wersji 1.2 Apache, do głównej dystrybucji serwera dołączany jest moduł *mod_expires*. Moduł ten pozwala właścicielowi witryny na wprowadzenie do odpowiedzi serwera nagłówków przekazujących informacje o terminie ważności danej strony; przeglądarka może na podstawie wartości takich nagłówków zdecydować o powtórnym załadowaniu strony w określonym czasie lub, jeśli termin ważności jest długi, o umieszczeniu strony w buforze lokalnym. Moduł wykorzystuje *mod_expires* wykorzystuje trzy dyrektywy:

ExpiresActive

```
ExpiresActive on|off
Zastosowanie: konfiguracja główna, serwery wirtualne, katalogi, pliki .htaccess
```

Dyrektywa ta służy do uaktywniania i blokowania mechanizmu przedawnień.

ExpiresByType

```
ExpiresByType typ-mime czas
Zastosowanie: konfiguracja główna, serwery wirtualne, katalogi, pliki .htaccess
```

Dyrektywa `ExpiresByType` przyjmuje dwa parametry. Parametr *`typ-mime`* określa typ MIME pliku. Parametr *`czas`* służy natomiast do określenia okresu, w przeciągu którego pliki danego typu MIME pozostają aktualne. Drugi parametr zapisywany jest przy użyciu jednej z dwóch składni. Pierwsza:

```
kod liczba-sekund
```

Pomiędzy kodem a liczbą sekund nie ma spacji. Kod może przyjąć jedną z dwóch wartości:

`A` — oznacza czas dostępu (czyli moment, w którym obsługiwane jest żądanie),

`M` — oznacza czas ostatniej modyfikacji pliku.

`liczba-sekund` jest zwykłą liczbą całkowitą. Przykładowo, aby określić termin ważności danego typu MIME jako 565 656 sekund od momentu odwołania się do takiego pliku, parametr *`czas`* dyrektywy należy określić następująco:

```
A565656
```

Druga, bardziej czytelna składnia parametru *`czas`* to:

```
baza [plus] liczba typ [liczba typ] ...
```

Baza może przyjąć jedną z trzech wartości:

`access`
: Moment dostępu.

`now`
: Synonim dla `access`.

`modification`
: Moment ostatniej modyfikacji pliku.

Słowo kluczowe `plus` jest opcjonalne. Jako typ należy podstawić jeden z poniższych specyfikatorów:

- `year[s]` (lata)
- `month[s]` (miesiące)
- `week[s]` (tygodnie)
- `day[s]` (dni)
- `hour[s]` (godziny)
- `minute[s]` (minuty)
- `second[s]` (sekundy)

Przykład parametru *czas* dyrektywy `ExpiresByType`, określającego termin ważności na 1 dzień i 4 godziny:

```
now plus 1 day 4 hours
```

ExpiresDefault

```
ExpiresDefault czas
Zastosowanie: konfiguracja główna, serwery wirtualne, katalogi, pliki .htaccess
```

Dyrektywa ta ustanawia domyślny termin ważności plików stosowany dla dokumentów, których terminu ważności nie określono za pomocą dyrektywy `ExpireByType`.