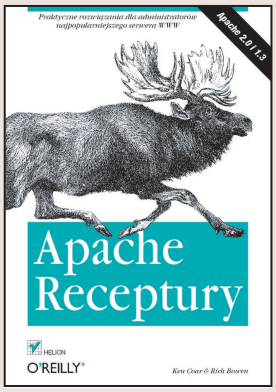
# Apache. Receptury




Autorzy: Ken Coar, Rich Bowen
Tłumaczenie: Witold Zioło
ISBN: 83-7361-416-8
Tytuł oryginału: Apache Cookbook
Format: B5, stron: 280

Apache jest najpopularniejszym obecnie serwerem WWW na świecie, co potwierdzają badania i statystyki. Większość witryn WWW działa właśnie na bazie tego serwera. Apache jest kolejnym, po Linuksie, potwierdzeniem fenomenu ruchu open source.

Dokumentacja Apache dostępna w sieci opisuje proces instalacji i konfiguracji serwera, co nie zawsze jest wystarczające, ponieważ administrowanie serwerem WWW działającym pod kontrolą Apache wymaga często rozwiązywania problemów pojawiających się w trakcie pracy. Dzięki ogromnej społeczności użytkowników, żadne wołanie o pomoc w sieci nie zostanie zignorowane, często jednak odpowiedź jest potrzebna natychmiast.

Książka „Apache. Receptury" jest zbiorem gotowych rozwiązań najczęściej pojawiających się problemów i wątpliwości, przeznaczonym dla administratorów, programistów i innych użytkowników Apache. Opisane w książce problemy są rzeczywiste – przydarzyły się autorom lub osobom zwracających się do nich o pomoc. Każde zagadnienie omówione jest w ten sam sposób – problem, analiza i rozwiązanie. Zaletą takiego przedstawienia informacji jest to, że Czytelnik dowiaduje się nie tylko, co powinien zrobić, ale również dlaczego powinien postąpić tak a nie inaczej. Wyjaśnienia zamieszczonych w książce kodów pomogą również dostosować je do innych przypadków.

Książka zawiera rozwiązania problemów występujących przy różnych czynnościach – od instalacji serwera, aż do obsługi szyfrowania SSL. Znajdujące się w niej informacje pomogą Czytelnikowi przy następujących przedsięwzięciach:

- Instalacja Apache w systemach Linux i Windows
- Instalowanie dodatkowych modułów
- Rejestracja zdarzeń i analizowanie dziennika zdarzeń serwera
- Konfigurowanie i obsługa serwerów wirtualnych
- Tworzenie aliasów, przekierowań i przypisań
- Zwiększenie poziomu bezpieczeństwa serwera
- Praca z szyfrowaniem SSL
- Konfigurowanie obsługi skryptów CGI i PHP
- Obsługa błędów
- Praca z serwerem proxy
- Optymalizacja i poprawa wydajności działania serwera

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

# *Spis treści*

# 3

# *Rejestracja zdarzeń*

Serwer WWW Apache może rejestrować i zwykle rejestruje wszystkie przetwarzane żądania. Kontrolowanie sposobu rejestracji oraz wydobywanie z dzienników zdarzeń przydatnych informacji jest równie ważne jak samo ich gromadzenie.

Dzienniki zdarzeń gromadzą dwa rodzaje danych — informacje o samym żądaniu oraz jeden lub więcej komunikatów informujących o nieprawidłowościach stwierdzonych w czasie przetwarzania żądań (na przykład spowodowanych brakiem odpowiednich uprawnień do pliku). Administrator nie ma zbyt wielu możliwości kontrolowania sposobu rejestracji informacji o błędach, ale może w znaczącym stopniu kontrolować format oraz ilość rejestrowanych informacji dotyczących przetwarzania żądań (*rejestracji aktywności*). Serwer może rejestrować informacje o żądaniach w kilku formatach i w kilku dziennikach zdarzeń, ale komunikaty o błędach może rejestrować tylko w jednym dzienniku.

Należy zdawać sobie sprawę z tego, że zapis w dzienniku zdarzeń pojawia się dopiero *po* tym, gdy żądanie zostanie całkowicie obsłużone. Czas, jaki upływa pomiędzy zgłoszeniem żądania a zakończeniem jego obsługi, może w pewnych przypadkach mieć znacznie.

Gdy, na przykład, w czasie pobierania szczególnie dużego pliku nastąpiła zmiana pliku dziennika zdarzeń, zapis w dzienniku odnotowujący żądanie pobrania pliku znajdzie się w nowym dzienniku w momencie, gdy zakończy się obsługa żądania, a nie w starym dzienniku, który obowiązywał, gdy żądanie nadeszło. W przeciwieństwie do tego, komunikaty o błędach rejestrowane są w dziennikach od razu po ich wystąpieniu.

Serwer WWW rejestruje zdarzenia przez cały czas pracy. Z tego powodu pliki dzienników zdarzeń mogą w przypadku bardzo obciążonych witryn WWW rozrastać się do niebotycznych rozmiarów, a w przypadku mniej obciążonych witryn ich rozmiar wciąż może być kłopotliwy. Żeby pliki dzienników zdarzeń nie rozrastały się zanadto, w większości witryn WWW zmienia się je co jakiś czas. Polega to na tym, że serwer zaprzestaje rejestracji zdarzeń w aktualnym pliku dziennika i rozpoczyna ją w nowym. Ponieważ serwer Apache stara się ze wszystkich sił nie zgubić żadnego wpisu,

zmuszenie go, by zmieniał plik dziennika na podstawie określonego harmonogramu, wymaga pewnego wysiłku. W kilku recepturach (na przykład 3.8 i 3.9) pokazano, jak zrobić to skutecznie i pewnie.

Dyrektywy rejestracji `CustomLog` oraz `ErrorLog` mogą pojawiać się wewnątrz kontenerów `<VirtualHost>`, poza nimi (w czymś co jest nazywane *serwerem głównym* lub *globalnym* lub czasem *zasięgiem globalnym*) lub w obu tych miejscach jednocześnie. Informacje są jednak rejestrowane tylko w jednym z zestawów plików — jeżeli żądanie lub błąd dotyczą kontenera `<VirtualHost>`, w którym znajduje się odpowiednia dyrektywa rejestracji, komunikat zostanie zapisany tylko w zestawie tego kontenera i nie pojawi się w żadnym z plików zadeklarowanych globalnie. Z drugiej strony, gdy w kontenerze `<VirtualHost>` nie ma żadnych dyrektyw rejestracji, serwer prowadzi rejestrację na podstawie dyrektyw globalnych.

Jednak, niezależnie od tego, który zasięg zostanie wykorzystany do określania używanych dyrektyw rejestracji, wszystkie dyrektywy `CustomLog` tego zasięgu będą przetwarzane i traktowane niezależnie. W takim przypadku, gdy dyrektywa `CustomLog` znajduje się w zasięgu globalnym i równocześnie dwie takie dyrektywy znajdują się wewnątrz kontenera `<VirtualHost>` — zastosowane zostaną *obydwie*. Podobnie, jeżeli dyrektywa `CustomLog` wykorzystuje opcję `env=`, nie ma wpływu na to, jakiego rodzaju żądania będą rejestrowane przez inne dyrektywy `CustomLog` w tym samym zasięgu.

Aktywność serwerów WWW rejestruje się od początku ich istnienia. Bardzo szybko pierwsi administratorzy serwerów WWW ustalili, jakiego rodzaju informacje mają być rejestrowane. Tak powstał *powszechny format rejestracji* CLF (ang. *Common Log Format*). W zapisie używanym przez serwer Apache ma on postać:

```
"%h %l %u %t \"%r\" %>s %b"
```

W zapisie tego typu rejestrowane są: nazwa komputera klienta lub jego adres IP, nazwa użytkownika klienta (w sposób zdefiniowany w RFC 1413, jeżeli serwerowi Apache nakazano jej śledzenie za pomocą dyrektywy `IdentityCheck On`), nazwa użytkownika, za pomocą której klient się uwierzytelnił (jeżeli serwer stosuje kontrolę dostępu), czas otrzymania żądania, treść żądania HTTP, końcowy stan przetwarzania żądania przez serwer oraz liczba bajtów zawartości wysłanej w odpowiedzi na żądanie.

Niedługo po tym, gdy protokół HTTP już dojrzał, zaistniała potrzeba stworzenia powszechnego formatu rejestrowania informacji, w wyniku czego powstał wzbogacony format o nazwie *złożony format rejestracji*:

```
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""
```

Pojawiły się w nim dwa nowe człony — `Referer` (napisany w specyfikacji z błędem) oraz `User-agent`. Są to odpowiednio: adres URL strony zawierającej łącze do żądanego dokumentu oraz nazwa i wersja przeglądarki lub innego klienta występującego z żądaniem.

Oba te formaty są obecnie szeroko stosowane, a wiele narzędzi do analizy dzienników zdarzeń zakłada, że zapisy dzienników mają jeden z dwóch powyższych formatów.

Standardowy moduł rejestracji aktywności serwera Apache nazywający się (niespodzianka!) *mod_log_config*, daje się bardzo dobrze konfigurować i umożliwia stosowanie własnych formatów dzienników zdarzeń. Serwer Apache 2.0 wyposażono w dodatkowy moduł *mod_logio*, który rozszerza funkcje modułu *mod_log_config* o możliwość rejestrowania liczby wysłanych i odebranych siecią bajtów. Jeżeli jednak oba te moduły nie spełniają oczekiwań, można użyć jednego z wielu dostępnych modułów innych producentów wymienionych w rejestrze modułów na stronie *http://modules.apache.org/*.

Kod stanu przetwarzania żądania występujący w obu formatach wymaga szerszego omówienia, gdyż jego znaczenie nie jest od razu oczywiste. Kody stanu zdefiniowane zostały w specyfikacji protokołu HTTP (aktualny dokument RFC 2616 znajduje się po adresem *ftp://ftp.isi.edu/in-notes/rfc2616.txt*). W tabeli 3.1 zebrano kody zdefiniowane do czasu powstania książki.

*Tabela 3.1. Kody stanu protokołu HTTP*

| **Kod** | **Znaczenie** |
|---|---|
| *1xx — kody informacyjne* | |
| 100 | Kontynuacja |
| 101 | Przełączenie protokołów |
| *2xx — kody powodzenia* | |
| 200 | OK |
| 201 | Utworzony |
| 202 | Zaakceptowany |
| 203 | Informacja nieautorytatywna |
| 204 | Brak zawartości |
| 205 | Zerowanie zawartości |
| 206 | Zawartość częściowa |
| *3xx — kody przekierowania* | |
| 300 | Wiele możliwości |
| 301 | Przeniesiony na stałe |
| 302 | Znaleziony |
| 303 | Spróbuj inny |
| 304 | Niezmodyfikowany |
| 305 | Użyj proxy |
| 306 | (Nieużywany) |
| 307 | Przekierowanie tymczasowe |

*Tabela 3.1. Kody stanu protokołu HTTP — ciąg dalszy*

| Kod | Znaczenie |
|---|---|
| *4xx — kody błędów klienta* | |
| 400 | Złe żądanie |
| 401 | Dostęp nieautoryzowany |
| 402 | Wymagana opłata |
| 403 | Dostęp zabroniony |
| 404 | Nieznaleziony |
| 405 | Metoda niedozwolona |
| 406 | Nie do zaakceptowania |
| 407 | Wymagane uwierzytelnienie proxy |
| 408 | Upłynął limit czasu żądania |
| 409 | Konflikt |
| 410 | Nieobecny |
| 411 | Potrzebna długość |
| 412 | Niespełnione warunki wstępne |
| 413 | Żądanie zbyt długie |
| 414 | Żądanie URI zbyt długie |
| 415 | Nieobsługiwany rodzaj medium |
| 416 | Zakres żądania niezadowalający |
| 417 | Oczekiwanie niespełnione |
| *5xx — kody błędów serwera* | |
| 500 | Wewnętrzny błąd serwera |
| 501 | Niezaimplementowany |
| 502 | Zła brama |
| 503 | Usługa niedostępna |
| 504 | Upłyną limit czasu bramy |
| 505 | Nieobsługiwana wersja protokołu HTTP |

Takie jednowierszowe opisy kodów stanu, jak te przedstawione w tabeli 3.1, mogą być mylące, ale przynajmniej dają pewne pojęcie o tym, co według serwera miało miejsce. Pierwsza cyfra kodu służy do wyróżnienia klas lub inaczej — kategorii kodów. Na przykład wszystkie kody zaczynające się cyfrą 5 oznaczają, że wystąpił problem obsługi żądania i że serwer „sądzi", że przyczyna tego leży po jego stronie, a nie po stronie klienta.

Pełny opis wszystkich kodów stanu można znaleźć w dokumentach poświęconych protokołowi HTTP lub w samym RFC.

# 3.1. Zwiększenie szczegółowości zapisów dziennika zdarzeń

## Problem

Zapisy dziennika zdarzeń dostępu powinny zawierać trochę więcej szczegółów.

## Rozwiązanie

Zamiast powszechnego (`common`) formatu rejestracji należy użyć złożonego formatu rejestracji (`combined`):

```
CustomLog logs/access_log combined
```

## Analiza

Domyślnie informacje rejestrowane przez serwer Apache zapisywane są w formacie powszechnym (`common`), jednak za pomocą dyrektywy `LogFormat` można zmienić go na format złożony (`combined`).

Złożony format rejestracji zawiera dwie dodatkowe informacje niedostępne w formacie powszechnym. Są to `Referer` — strona, z której nadeszło żądanie, oraz `User-agent` — używana przez klienta przeglądarka.

Każdy poważny program analizujący dzienniki zdarzeń potrafi analizować dzienniki utworzone w obu formatach, a wiele z tych programów tworzy na podstawie dodatkowych pól dodatkowe statystyki. Używając formatu złożonego, nie traci się nic, a można zyskać dodatkowe informacje.

## Zobacz również

- *http://httpd.apache.org/docs/mod/mod_log_config.html*.
- *http://httpd.apache.org/docs-2.0/mod/mod_log_config.html*.

# 3.2. Zwiększenie liczby komunikatów o błędach

## Problem

Aby można było łatwiej znaleźć przyczynę problemu, w dzienniku błędów powinno znajdować się więcej informacji.

## Rozwiązanie

W tym celu w pliku *httpd.conf* należy zmienić lub dodać do niego wiersz `LogLevel`. Dyrektywa ta ma kilka argumentów omówionych dalej.

Na przykład:

```
LogLevel Debug
```

## Analiza

Istnieje kilka zhierarchizowanych poziomów rejestracji błędów, każdy oznaczony innym słowem kluczowym. Domyślnym argumentem dyrektywy `LogLevel` jest `warn`. Innymi możliwymi argumentami są w kolejności od najważniejszej:

`emerg`
: Zdarzenia krytyczne, gdy serwer WWW nie nadaje się do użycia.

`alert`
: Zdarzenia wymagające natychmiastowego podjęcia działania.

`crit`
: Zdarzenia krytyczne.

`error`
: Błędy.

`warn`
: Ostrzeżenia.

`notice`
: Zdarzenia normalne, ale istotne.

`info`
: Informacje.

`debug`
: Komunikaty poziomu rozwiązywania problemów.

Najmniej informacji będzie rejestrowanych w przypadku użycia argumentu `emerg`, a w przypadku użycia argumentu `debug` — najwięcej. W tej drugiej sytuacji rejestrowanych będzie prawdopodobnie wiele informacji niezwiązanych z rozwiązywanym problemem, więc po rozwiązaniu problemu należy powrócić do poprzedniego poziomu rejestracji.

Mimo że poziomy rejestracji są z natury zhierarchizowane, komunikaty poziomu `notice` są rejestrowane *zawsze*, bez względu na ustawienia dyrektywy `LogLevel`.

Poziomy rejestracji są dość luźno zdefiniowane i jeszcze luźniej używane. Mówiąc inaczej — poziom, przy którym zostanie odnotowany dany błąd, zależy wyłącznie od poglądu osoby, która napisała kod programu. Opinia administratora na ten temat może być inna.

Oto kilka przykładów komunikatów o różnych poziomach:

```
[Thu Apr 18 01:37:40 2002] [alert] [client 64.152.75.26]
/home/smith/public_html/test/.htaccess: Invalid command 'Test', perhaps
mis-spelled or defined by a module not included in the server configuration
[Thu Apr 25 22:21:58 2002] [error] PHP Fatal error: Call to undefined
function: decode_url( ) in /usr/apache/htdocs/foo.php on line 8
[Mon Apr 15 09:31:37 2002] [warn] pid file /usr/apache/logs/httpd.pid
overwritten --Unclean shutdown of previous Apache run?
[Mon Apr 15 09:31:38 2002] [info] Server built: Apr 12 2002 09:14:06
[Mon Apr 15 09:31:38 2002] [notice] Accept mutex: sysvsem (Default: sysvsem)
```

Są to typowe komunikaty, z którymi można zetknąć się w przypadku czynnego serwera WWW. Po zmianie poziomu na `debug` może pojawić się dużo więcej tajemniczych komunikatów, takich jak:

```
[Thu Mar 28 10:29:50 2002] [debug] proxy_cache.c(992): No CacheRoot, so no
caching. Declining.
[Thu Mar 28 10:29:50 2002] [debug] proxy_http.c(540): Content-Type: text/html
```

Do tego właśnie służą te komunikaty — dzięki nim programista serwera Apache może dowiedzieć się, co dokładnie dzieje się w module proxy.

## *Zobacz również*

W czasie, gdy powstawała ta książka, prowadzone były wysiłki mające na celu stworzenie słownika komunikatów o błędach serwera Apache, dzięki któremu będzie wiadomo, co one dokładnie oznaczają i co w przypadku ich wystąpienia należy zrobić. Jednak na razie nie można jeszcze zaprezentować żadnych wyników tych prac. Gdy prace się zakończą, na pewno pojawi się odpowiednia informacja na stronie twórców serwera Apache:

*http://httpd.apache.org/dev/*

W takim przypadku informacja ta znajdzie się również na stronie WWW książki:

*http://Apache-Cookbook.Com/*

Szczegółowy opis działania dyrektywy `LogLevel` można znaleźć na stronie serwera Apache:

*http://httpd.apache.org/docs/mod/core.html#loglevel*

# 3.3. Rejestracja zawartości POST

## Problem

Należy rejestrować dane dostarczone za pomocą metody POST, na przykład w postaci formularzy WWW.

## Rozwiązanie

W przypadku serwera Apache 1.3 nie jest to w zasadzie możliwe, chyba że robi to moduł obsługujący metodę POST. W serwerze Apache 2.0 można to zrobić za pomocą filtrów, ale w czasie, gdy powstawała ta książka nie było jeszcze filtrów tego rodzaju.

## Analiza

W wersji 1.3 serwera Apache możliwość przetwarzania treści komunikatu żądania zawierającego zmienną POST ma tylko jeden moduł i jedynie on może je rejestrować. Informacja o żądaniu jest czytana z sieci tylko raz — przez moduł wybrany przez serwer do utworzenia odpowiedzi. Z tego powodu informacja ta nie jest dostępna w czasie fazy rejestracji, która następuje po fazie obsługi zawartości.

Gdy używany jest, na przykład, moduł *mod_perl*, informacje tego rodzaju można rejestrować, jeżeli kod obsługujący zawartość i tworzący odpowiedzi jest skryptem języka Perl obsługiwanym przez moduł *mod_perl*.

## Zobacz również

- Jeżeli pojawi się odpowiedni filtr wejściowy, informacja o nim znajdzie się na stronie WWW książki:

  *http://Apache-Cookbook.Com/*

- Więcej informacji na ten temat można znaleźć w różnych artykułach omawiających filtry serwera Apache 2.0, na przykład znajdujących się na stronach:

  *http://OnLAMP.Com/apache/*

  *http://ApacheToday.Com/*

# 3.4. Rejestracja adresu IP klienta łączącego się poprzez serwer proxy

## Problem

Należy rejestrować adres IP klienta otwierającego strony WWW, nawet gdy jego żądania przychodzą za pośrednictwem serwera proxy.

## Rozwiązanie

Brak.

## Analiza

Niestety, uniemożliwia to sam protokół HTTP. Patrząc od strony klienta — serwery proxy są zupełnie przeźroczyste, a patrząc od strony serwera WWW, w którym znajduje się żądana zawartość — są one całkowicie nieprzeźroczyste, co powoduje, że tożsamość klienta zgłaszającego żądanie jest ukrywana.

Pozostaje tylko rejestrowanie adresów IP, z których nadchodzą żądania. Jeżeli przychodzą one bezpośrednio z przeglądarki — adresem tym będzie adres klienta, a jeżeli przychodzą one z jednego lub więcej serwerów proxy, będzie to adres tego serwera proxy, który bezpośrednio łączy się z serwerem WWW.

Oba formaty rejestracji — złożony oraz powszechny — zawierają dyrektywę `%h` reprezentującą tożsamość (zdalnego) klienta. W zależności od ustawień dyrektywy `HostNameLookups` może być nią nazwa komputera lub jego adres IP. Jeżeli w dzienniku zdarzeń mają pojawiać się wyłącznie adresy IP, należy użyć dyrektywy `%a`.

## Zobacz również

- Specyfikacja protokołu HTTP dostępna pod adresem *ftp://ftp.isi.edu /in-notes/rfc2616.txt*.

# 3.5. Rejestracja adresu MAC klienta

## Problem

Należy rejestrować adres MAC (adres sprzętowy) klienta łączącego się z serwerem WWW.

## Rozwiązanie

W większości przypadków nie można tego zrobić wcale i nie może tego zrobić również serwer Apache.

## Analiza

Adres MAC nie ma żadnego znaczenia, za wyjątkiem przypadków, gdy żądanie nadchodzi z sieci lokalnej (LAN). Na dodatek adres MAC nie występuje w transakcjach przeprowadzanych poprzez sieci rozległe (WAN). Gdy pakiet sieciowy przechodzi przez router, na przykład, gdy opuszcza sieć LAN, w polu adresu MAC pakietu, router zazwyczaj umieszcza swój adres sprzętowy.

## Zobacz również

- Specyfikacja protokołu TCP/IP (należy przejść na stronę *http://www.rfc-editor.org/cgi-bin/rfcsearch.pl* i w polu wyszukiwania wpisać „TCP").

# 3.6. Rejestracja Cookies

## Problem

Należy rejestrować wszystkie cookies wysyłane do serwera przez klienty oraz wszystkie cookies, o które serwer prosi, by klienci umieścili je w swych bazach. Może się to przydać do rozwiązywania problemów z aplikacjami WWW posługującymi się cookies.

## Rozwiązanie

Aby rejestrować cookies otrzymywane od klientów, należy użyć:

```
CustomLog logs/cookies_in.log "%{UNIQUE_ID}e %{Cookie}i"
CustomLog logs/cookies2_in.log "%{UNIQUE_ID}e %{Cookie2}i"
```

natomiast, aby rejestrować wartości cookies, które ustala i wysyła do klienta serwer:

```
CustomLog logs/cookies_out.log "%{UNIQUE_ID}e %{Set-Cookie}o"
CustomLog logs/cookies2_out.log "%{UNIQUE_ID}e %{Set-Cookie2}o"
```

Użycie dyrektywy formatu `%{Set-Cookie}o` do rozwiązywania problemu nie jest zalecane w sytuacji, gdy dotyczy on (lub może dotyczyć) wielu cookies — w dzienniku zdarzeń zostanie zarejestrowane tylko pierwsze. W punkcie zawierającym analizę tej receptury zostanie podany odpowiedni przykład.

W czasie, gdy powstawała ta książka, serwer Apache nie potrafił rejestrować wartości wszystkich cookies, ale jeden z autorów książki pracuje nad rozwiązaniem tego problemu. Gdy problem zostanie już rozwiązany, odpowiednia informacja pojawi się na stronie WWW książki (*http://Apache-Cookbook.Com/*).

# *Analiza*

Pola cookies bywają bardzo długie i złożone, zatem podane instrukcje rejestrują je w oddzielnych plikach. Zapisy dziennika cookies można wiązać z dziennikiem żądań dostępu odbieranych od klientów za pomocą zmiennej środowiskowej `server-set UNIQUE_ID` (zakładając, że moduł *mod_unique_id* jest aktywny, i że w formacie rejestracji zdarzeń tę zmienną środowiska uwzględniono za pomocą dyrektywy formatu `%{UNIQUE_ID}e`).

W czasie, gdy powstawała ta książka, najczęściej stosowane były pola nagłówka `Cookie` oraz `Set-Cookie`. Pola `Cookie2` oraz `Set-Cookie2` są nowsze i zostały stworzone w celu poprawienia niektórych niedoskonałości pierwotnej specyfikacji, ale nie zyskały sobie jeszcze większej popularności.

Z powodu sposobu, w jaki zmieniała się z czasem składnia pól nagłówka cookie, instrukcje rejestracji mogą, ale nie muszą, rejestrować wszystkich szczegółów cookies.

Należy pamiętać, że podane dyrektywy rejestracji będą rejestrować wszystkie cookies, a nie tylko te, które z jakichś powodów są interesujące. Na przykład, niżej pokazano zapis dziennika przedstawiający żądanie klienta zawierające dwa cookies — jedno nazywające się `RFC2109-1`, a drugie — `RFC2109-2`:

```
PNCSUsCoF2UAACI3CZs RFC2109-1="This is an old-style cookie, with space
characters embedded"; RFC2109-2=This_is_a_normal_old-style_cookie
```

Mimo że jest to jeden zapis, zawiera on informacje o dwóch cookies.

Po stronie ustalającej zawartość cookie, pole nagłówka `Set-Cookie` wysyłane przez serwer w nagłówku swojej odpowiedzi wygląda tak:

```
Set-Cookie: RFC2109-1="This is an old-style cookie, with space characters
embedded"; Version=1; Path=/; Max-Age=60; Comment="RFC2109 demonstration
cookie"
Set-Cookie: RFC2109-2=This_is_a_normal_old-style_cookie; Version=1; Path=/;
Max-Age=60; Comment="RFC2109 demonstration cookie"
```

W dzienniku pojawia się zapis odpowiadający tej odpowiedzi:

```
PNCSUsCoF2UAACI3CZs RFC2109-1="This is an old-style cookie, with space
characters embedded"; Version=1; Path=/; Max-Age=60; Comment="RFC2109
demonstration cookie"
```

Jak widać, rejestracja pola `Cookie` w nagłówku żądania została wykonana poprawnie, ale zarejestrowano tylko jedno z pól nagłówka `Set-Cookie` odpowiedzi.

## Zobacz również

- RFC 2109 — *„HTTP State Management Mechanizm"* — definicje IETF pól nagłówka `Cookie` oraz `Set-Cookie` dostępne pod adresem *ftp://ftp.isi.edu/in-notes/rfc2109.txt*.
- RFC 2965 — *„HTTP State Management Mechanism"* — definicje IETF pól nagłówka `Cookie2` oraz `Set-Cookie2` dostępne pod adresem *ftp://ftp.isi.edu /in-notes/rfc2965.txt*.
- Oryginalna propozycja specyfikacji cookies firmy Netscape znajdująca się pod adresem *http://home.netscape.com/newsref/std/cookie_spec.html*.

# 3.7. Zaniechanie rejestracji żądań pobierania obrazów pochodzących ze stron lokalnych

## Problem

Należy rejestrować żądania pobierania obrazów znajdujących się w witrynie WWW za wyjątkiem żądań pochodzących z własnych stron witryny. Przydaje się w przypadku konieczności ograniczenia rozmiaru pliku dziennika zdarzeń lub po to, by wyśledzić strony WWW, które traktują te obrazy jako własne.

## Rozwiązanie

Aby ograniczyć rejestrację żądań tylko do pochodzących spoza danego ośrodka WWW, można wykorzystać dyrektywę `SetEnvIfNoCase`:

```
<FilesMatch \.(jpg|gif|png)$>
    SetEnvIfNoCase Referer "^http://www.przykladowa.pl/" local_referrer=1
</FilesMatch>
CustomLog logs/access_log combined env=!local_referrer
```

## Analiza

W wielu przypadkach strony umieszczone na serwerze WWW zawierają odwołania do obrazów przechowywanych również na tym samym serwerze. Z punktu widzenia analizy dzienników zdarzeń jedynymi interesującymi zapisami są tylko te, które informują o stronach, z których nastąpiło odwołanie. Jak zapobiec rejestrowaniu żądań pobierania obrazów pochodzących ze strony lokalnej serwera?

W przypadku, gdy strona, na której znajdują się łącza do obrazu, umieszczona jest w serwerze *www.przykladowa.pl* (oczywiście nazwę tę należy zamienić na rzeczywistą nazwę serwera) i żądane są obrazy umieszczone w plikach GIF, PNG lub JPEG, za pomocą dyrektywy `SetEnvIfNoCase` nadana zostanie wartość odpowiedniej zmiennej środowiska.

Dyrektywa `SetEnvIfNoCase` jest podobna do dyrektywy `SetEnvIf`, różni się od niej tylko tym, że porównuje zmienne bez zwracania uwagi na wielkość liter.

Dyrektywa `CustomLog` zarejestruje wszystkie żądania, którym nie towarzyszy określona wartość zmiennej środowiska, czyli wszystkie żądania pobrania obrazów pochodzące spoza przykładowej strony.

Przedstawione tu rozwiązanie działa tylko w przypadku klientów WWW, które informują o stronie, z której pochodzi żądanie. Niektórzy uważają, że adresy URL stron, z których następuje odwołanie, nie powinny nikogo interesować. W przypadku niektórych klientów można wybrać, czy informacja ta ma być przekazywana czy też nie. W internecie działają też serwery zapewniające anonimowość, które działają jako serwery proxy i skutecznie ukrywają tego typu informacje.

## Zobacz również

- Receptura 6.5.

# 3.8. Zmiana pliku dziennika zdarzeń o określonej porze dnia

## Problem

O określonej godzinie należy zmienić plik dziennika zdarzeń serwera Apache w sposób niewymagający zatrzymania i ponownego uruchamiania serwera.

## Rozwiązanie

Należy posłużyć się dyrektywą `CustomLog` oraz programem *rotatelogs*:

```
CustomLog "| /ścieżka/do/rotatelogs /ścieżka/do/logs/access_log.%Y-%m-%d
86400" combined
```

## Analiza

Skrypt *rotatelogs* wykorzystuje funkcję serwera Apache o nazwie *rejestrowanie potokowe* (ang. *piped logging*), której działanie polega na kierowaniu komunikatów zdarzeń nie do pliku, a na wejście innego programu. Umieszczając skrypt *rotatelogs* pomiędzy serwerem WWW, a plikiem na dysku, można uniknąć konieczności zatrzymywania i ponownego uruchamiania serwera WWW w celu utworzenia nowego pliku dziennika zdarzeń. Skrypt automatycznie otworzy nowy plik i o określonej godzinie rozpocznie zapisywanie do niego.

Pierwszym argumentem skryptu *rotatelogs* jest stały człon nazwy pliku dziennika zdarzeń. Jeżeli argument ten będzie zawierać jeden lub więcej znaków `%`, zostanie uznany za ciąg w formacie `strftime(3)`. W przeciwnym razie do stałego członu nazwy dodany zostanie czas zmiany pliku (wyrażony w liczbie sekund, które upłynęły od 1 stycznia 1970) w postaci dziesięciocyfrowej liczby. Na przykład, gdy stałym członem nazwy będzie wyraz `foo`, nazwy plików dzienników będą miały postać *foo.1020297600*, natomiast w przypadku stałego członu nazwy w postaci `foo.%Y-%m-%d`, nazwy plików dzienników będą miały postać *foo.2002-04-29*.

Drugim argumentem jest wyrażony w sekundach interwał pomiędzy zmianami plików. Zmiana pliku nastąpi wówczas, gdy wartość czasu systemowego będzie wielokrotnością interwału. Doba składa się z 86400 sekund, zatem użycie interwału wynoszącego 86400 sekund spowoduje, że nowy plik dziennika będzie tworzony o północy każdego dnia — gdy czas systemowy oparty na dacie 1 stycznia 1970 będzie wielokrotnością 24 godzin.

Ponieważ interwał jest liczbą sekund, która upływa pomiędzy zmianą pliku, zmiana czasu z zimowego na letni nie ma wpływu na czas pomiędzy kolejnymi zmianami plików.

## Zobacz również

- Strona podręcznika man programu *rotatelogs*:

  ```
  % man -M /ścieżka/do/ServerRoot/man/rotatelogs.8
  ```

  gdzie w miejsce */ścieżka/do/ServerRoot* należy wpisać ścieżkę użytą w dyrektywie `ServerRoot` umieszczonej w pliku *httpd.conf*.

# 3.9. Zmiana pliku dziennika zdarzeń pierwszego dnia miesiąca

## Problem

W pierwszym dniu nowego miesiąca powinno zakończyć się rejestrowanie zdarzeń w dotychczasowym pliku i rozpocząć w nowym.

## Rozwiązanie

W dystrybucji serwera Apache nie ma żadnego skryptu, który realizowałby tę funkcję, ale skrypt taki można znaleźć na stronie WWW książki pod adresem *http://Apache-Cookbook.Com/sources/Chapter04/rotate-log-monthly.pl*[1].

[1] W grudniu 2003 roku skrypt ten nie był dostępny pod wskazanym adresem, gdyż, jak twierdzą autorzy książki, „są z nim pewne problemy" — *przyp. tłum.*

Skryptu używa się, kierując przez niego potok komunikatów o zdarzeniach, na przykład:

```
CustomLog "| rotate-log-monthly.pl logs/access_log-%Y-%M" CLF
```

Argumentem skryptu jest nazwa pliku dziennika zdarzeń. Ciąg znaków rozpoczynający się od znaku `%` przesyłany jest funkcji `strftime(3)` w celu utworzenia członu nazwy nowego pliku dziennika uzależnionego od daty utworzenia tego pliku.

## *Analiza*

Podobnie jak w przypadku innych przedstawionych w tym rozdziale rozwiązań rejestracji zdarzeń skrypt ten pełni tylko jedną funkcję. Jeżeli trzeba wykonać kilka funkcji — na przykład rozdzielić dzienniki zdarzeń pomiędzy serwery wirtualne i zmieniać pliki dzienników każdego pierwszego dnia miesiąca — należy użyć innych skryptów.

Rozwiązanie zastosowane w skrypcie *rotate-log-monthly.pl* należy raczej do rozwiązań typu „ślepej siły" i może nie działać zbyt dobrze w przypadku bardzo obciążonych serwerów. Przyczyną tego jest odczytywanie czasu systemowego podczas zapisu każdej pozycji dziennika zdarzeń. Mimo to skrypt stanowi ilustrację przyjętej w nim metody.

## *Zobacz również*

- *http://httpd.apache.org/docs/logs.html#piped*.

# *3.10. Rejestracja nazw komputerów zamiast ich adresów IP*

## *Problem*

W dziennikach zdarzeń powinny być umieszczane nazwy komputerów, a nie ich adresy IP.

## *Rozwiązanie*

Serwer WWW może w czasie przetwarzania żądania klienta odwzorowywać adres IP klienta na odpowiadającą mu nazwę komputera. W tym celu należy wydać polecenie:

```
HostnameLookups On
```

Można też pozostać w czasie przetwarzania żądania przy adresie IP, a odwzorowanie adresu IP przeprowadzić w fazie zapisywania pozycji dziennika, wykorzystując do tego celu potok:

```
HostnameLookups Off
CustomLog "| /ścieżka/do/logresolve -c >>
/ścieżka/do/logs/access_log.resolved" combined
```

Można również zapisywać w dzienniku adresy IP klientów i odwzorować je na nazwy komputerów później — podczas analizowania dziennika zdarzeń. W tym celu do pliku *httpd.conf* należy dodać wiersz:

```
CustomLog /ścieżka/do/logs/access_log.raw combined
```

I w razie potrzeby przetworzyć plik dziennika za pomocą polecenia:

```
% /ścieżka/do/logresolve -c < access_log.raw > access_log.resolved
```

# *Analiza*

Mechanizm rejestracji aktywności serwera Apache może rejestrować adresy IP klientów lub nazwy ich komputerów (albo jedno i drugie). Rejestrowanie nazwy komputera w czasie przetwarzania żądania powoduje, że serwer musi poświęcić trochę czasu na wykonanie operacji wyszukiwania w DNS mającej na celu zamianę (już posiadanego) adresu IP na odpowiadającą mu nazwę komputera. Tego rodzaju działanie może mieć poważny wpływ na wydajność serwera, gdyż aby zamienić adres komputera na jego nazwę, proces potomny serwera (lub jego wątek) musi za każdym razem zwracać się do odpowiedniej usługi odwzorowującej, a w czasie, gdy oczekuje on odpowiedzi — nie może obsługiwać innych żądań klienta. Alternatywą dla takiego postępowania jest rejestrowanie adresów IP klientów i odwzorowanie ich przez serwer na nazwy komputerów w czasie przetwarzania pliku dziennika. Ostatnią możliwością jest przeprowadzenie tej czynności za pomocą oddzielnego procesu, który nie angażuje serwera WWW.

Teoretycznie do dyspozycji jest zatem szeroki wybór środków, jednak nie są one wolne od pułapek. Winny jest temu sam, dołączony do serwera Apache, program *logresolve* (zwykle instalowany w podkatalogu *bin/* katalogu `ServerRoot`), który odwzorowuje tylko adresy IP występujące na samym początku zapisów dziennika zdarzeń, a zatem nie nadaje się do użycia w przypadku, gdy stosowane są niestandardowe formaty dzienników. Wadą ostatniego rozwiązania jest również to, że pomiędzy zarejestrowaniem adresów IP, a ich odwzorowaniem na nazwy komputerów może minąć zbyt dużo czasu, w którym informacje DNS mogą się istotnie zmienić, co może doprowadzić do mylących lub nawet niepoprawnych wyników. Dotyczy to szczególnie adresów IP nadawanych dynamicznie, co ma miejsce na przykład w przypadku adresów IP nadawanych przez dostawców usług internetowych (ISP).

Inna wada tych rozwiązań ujawnia się wówczas, gdy komunikaty o zdarzeniach kierowane są do programu *logresolve* za pośrednictwem potoku — do wersji 1.3.24 serwera Apache program *logresolve* nie opróżnia buforów wyjściowych natychmiast, istnieje więc niebezpieczeństwo utraty przechowywanych w nich informacji o zdarzeniach spowodowanej, na przykład, przez załamanie się procesu rejestracji lub działania systemu.

## Zobacz również

- Strona podręcznika man programu *logresolve*:

```
% man -M /ścieżka/do/ServerRoot/man/logresolve.8
```

# 3.11. Oddzielne pliki dzienników zdarzeń serwerów wirtualnych

## Problem

Każdy z serwerów wirtualnych ma mieć oddzielny plik dziennika zdarzeń, ale nie powinny być one stale otwarte, jak to ma miejsce w przypadku wielokrotnego użycia dyrektywy `CustomLog`.

## Rozwiązanie

W tym celu należy posłużyć się programem *split-logfile* znajdującym się w pakiecie serwera Apache. Aby podzielić plik dziennika zdarzeń po zakończeniu w nim rejestracji, należy wydać następujące polecenia (*`/ścieżka/do/ServerRoot`* należy zamienić na prawidłową ścieżkę):

```
# cd /ścieżka/do/ServerRoot
# mv logs/access_log logs/access_log.old
# bin/apachectl graceful
[należy poczekać na zamknięcie dotychczasowego pliku dziennika]
# cd logs
# ../bin/split-logfile < access_log.old
```

Aby rozdzielać zapisy do odpowiednich plików w czasie, gdy one powstają, należy do pliku *httpd.conf* dodać wiersz:

```
CustomLog "| /ścieżka/do/split-logfile /usr/local/Apache/logs" combined
```

## Analiza

Aby program *split-logfile* działał prawidłowo, format zapisów w dzienniku zdarzeń musi rozpoczynać się od dyrektywy „`%v `” (po literze `v` musi znajdować się spacja). W ten sposób na samym początku każdego zapisu dziennika zdarzeń znajdzie się nazwa serwera wirtualnego i na jej podstawie program *split-logfile* będzie mógł zdecydować, do którego pliku powinna trafić dana pozycja dziennika. Przed przeniesieniem zapisu do odpowiedniego dziennika zdarzeń nazwa serwera wirtualnego zostanie z zapisu usunięta.

Są dwa sposoby podziału dziennika zdarzeń — po jego zapisaniu, zamknięciu i rozpoczęciu zapisywania do nowego pliku lub na bieżąco, w miarę powstawania kolejnych zapisów.

Aby rozdzielić zamknięty plik, należy skierować go na wejście skryptu *split-logfile*. Aby rozdzielać zapisy w czasie ich powstawiania, należy tak zmodyfikować konfigurację, by komunikaty o zdarzeniach trafiały do skryptu za pomocą potoku.

Każda z tych metod ma swoje wady i zalety. W przypadku pierwszej metody potrzeba dwukrotnie więcej miejsca na dysku (na przechowywanie pliku nierozdzielonego oraz plików rozdzielonych). Należy też dopilnować, aby plik dziennika zdarzeń został całkowicie zamknięty. (Niestety, nie ma prostej możliwości zrobienia tego bez całkowitego zatrzymania pracy serwera lub ponownego uruchomienia serwera z opcją `graceful`, a i tak możliwe jest, że jakieś powolne połączenie nie dopuści do zamknięcia dotychczasowego pliku dziennika przez dłuży czas po ponownym uruchomieniu serwera z opcją `graceful`). Rozdzielanie zapisów w miarę ich powstawania jest bardzo wrażliwe na załamanie się procesu rejestracji zdarzeń — mimo to, że serwer Apache uruchomi go ponownie, komunikaty o zdarzeniach oczekujące przetworzenia mogą się spiętrzyć i utworzyć w serwerze zator.

## Zobacz również

- Receptura 3.10.

# 3.12. Rejestracja żądań proxy

## Problem

Żądania przechodzące przez własny serwer proxy mają być odnotowywane w innym pliku niż żądania przychodzące bezpośrednio do serwera.

## Rozwiązanie

Żądania przechodzące przez własny serwer proxy można oznaczyć za pomocą dyrektywy `SetEnv`, dzięki czemu będzie możliwa ich rejestracja warunkowa:

```
<Directory proxy:*>
    SetEnv is_proxied 1
</Directory>
CustomLog logs/proxy_log combined env=is_proxied
```

## Analiza

Serwer Apache 1.3 używa specjalnej składni dyrektywy `<Directory>` dotyczącej w szczególności żądań przechodzących przez moduł proxy. Mimo że znak gwiazdki (`*`) sugeruje, że do dopasowywania dokumentów można używać symboli wieloznacznych — nie jest to wcale symbol wieloznaczny. Można dopasowywać albo całe ścieżki, takie jak na przykład *proxy:http://example.com/foo.html*, lub za pomocą znaku `*` dopasowywać *wszystko*. Nie można użyć dopasowania takiego jak: *proxy:http://example.com/*.html*.

Gdy do różnych ścieżek proxy trzeba stosować różne dyrektywy — należy skorzystać z innego modułu. Ponieważ żądania przechodzące przez serwer proxy, a nie są obsługiwane bezpośrednio (czyli dany serwer jest serwerem *proxy*, a nie jest serwerem *docelowym*), w celu zastosowania dyrektyw do konkretnych dokumentów dostępnych poprzez proxy nie może użyć kontenera `<Files>` ani `<FilesMatch>`. Nie można też użyć sekcji `<Location>` ani `<LocationMatch>`, gdyż nie mogą się one pojawiać w kontenerze `<Directory>`. Można jednak, w celu podjęcia decyzji na podstawie ścieżki żądanego dokumentu, skorzystać z modułu *mod_rewrite*. Na przykład żądania dotyczące obrazów przechodzące przez proxy można rejestrować w oddzielnym pliku za pomocą dyrektyw:

```
<Directory proxy:*>
    RewriteEngine On
    RewriteRule "\.(gif|png|jpg)$" "-" [ENV=proxied_image:1]
    RewriteCond "%{ENV:proxied_image}" "!1"
    RewriteRule "^" "-" [ENV=proxied_other:1]
</Directory>
CustomLog logs/proxy_image_log combined env=proxied_image
CustomLog logs/proxy_other_log combined env=proxied_other
```

Dyrektywy umieszczone wewnątrz kontenera `<Directory proxy:*>` będą dotyczyć jedynie żądań przechodzących przez serwer. Pierwsza dyrektywa `RewriteRule` definiuje zmienną środowiska w przypadku, gdy nazwa żądanego dokumentu kończy się rozszerzeniem *.gif*, *.png* lub *.jpg*. Dyrektywa `RewriteCond` sprawdza, czy zmienna nie jest zdefiniowana, i w takim przypadku następująca po niej dyrektywa `RewriteRule` ustala wartość innej zmiennej środowiska. Obie dyrektywy `CustomLog` w zależności od wartości zmiennych środowiska wysyłają do różnych plików informacje o różnych rodzajach żądań.

## Zobacz również

- Dokumentacja modułów *mod_rewrite* oraz *mod_log_config*.

# 3.13. Rejestracja komunikatów o błędach różnych serwerów wirtualnych w różnych plikach

## Problem

W przeciwieństwie do informacji o zdarzeniach związanych z dostępem do serwera WWW informacje o błędach serwer Apache rejestruje tylko w jednym pliku. Należy wobec tego użyć środków, które spowodują, że komunikaty o błędach dotyczące serwerów wirtualnych będą trafiały do ich indywidualnych dzienników zdarzeń oraz jednocześnie do dziennika globalnego.

## Rozwiązanie

Istnieją co najmniej dwa rozwiązania tego problemu:

1. Można za pomocą potoku skierować komunikaty o błędach do odpowiedniego skryptu, który skopiuje je i skieruje do odpowiednich plików.
2. Można za pomocą potoku kierować komunikaty o błędach do kilku dzienników zdarzeń jednocześnie:

```
ErrorLog "| tee logfile1 | tee logfile2 > logfile3"
```

## Analiza

W przeciwieństwie do komunikatów o zdarzeniach związanych z dostępem, komunikaty o błędach serwer Apache rejestruje tylko w jednym pliku. Jeżeli komunikat o błędzie dotyczy konkretnego serwera wirtualnego, a w jego kontenerze `<VirtualHost>` znajduje się dyrektywa `ErrorLog`, błąd zostanie odnotowany tylko w tym pliku i nie pojawi się już w żadnym dzienniku globalnym. Gdy w kontenerze `<VirtualHost>` nie ma dyrektywy `ErrorLog`, komunikat o błędzie trafi jedynie do dziennika globalnego. (Dziennik globalny definiuje ostatnia napotkana dyrektywa `ErrorLog` nieznajdująca się w żadnym kontenerze `<VirtualHost>`).

Obecnie jedynym rozwiązaniem tego problemu jest powielanie zapisów za pomocą oddzielnego procesu (przez, na przykład, wysyłanie komunikatów o błędach do tego procesu za pomocą potoku). Spośród przedstawionych wyżej dwóch rozwiązań pierwsze wymaga użycia własnoręcznie napisanego skryptu i jest bardziej elastyczne. Natomiast jeżeli wystarczy proste powielanie wszystkich zapisów dziennika do wielu plików jednocześnie, lepsze będzie drugie rozwiązanie, ale by ono działało, potrzebny jest program `tee`, którego nie ma na przykład w systemie Windows. To rozwiązanie może również prowadzić do opóźnień zapisywania komunikatów, gdy używany program `tee` nie opróżnia natychmiast swoich buforów wyjściowych po otrzymaniu każdego rekordu. W takim przypadku, przerwanie potoku lub załamanie systemu może doprowadzić do utraty komunikatów.

## Zobacz również

- *http://httpd.apache.org/docs/logs.html#piped*.

# 3.14. Rejestracja adresu IP serwera

## Problem

Należy rejestrować adres IP serwera, który obsłużył żądanie. Przydaje się to w przypadku, gdy każdy z serwerów wirtualnych dysponuje wieloma adresami IP.

## Rozwiązanie

W dyrektywie `LogFormat` lub `CustomLog` należy posłużyć się dyrektywą formatu `%A`:

```
CustomLog logs/served-by.log "%{UNIQUE_ID}e %A"
```

## Analiza

Dyrektywa `%A` informuje system rejestracji aktywności serwera, by w odpowiednim miejscu dziennika zdarzeń umieszczał również lokalny adres IP — czyli adres IP serwera WWW. Może się to przydać, gdy serwer używa kilku adresów IP. W takim przypadku w konfiguracji serwera mogą, na przykład, znaleźć się następujące dyrektywy:

```
Listen 10.0.0.42
Listen 192.168.19.243
Listen 263.41.0.80
<VirtualHost 192.168.19.243>
    ServerName Private.Example.Com
</VirtualHost>
<VirtualHost 10.0.0.42 263.41.0.80>
    ServerName Foo.Example.Com
    ServerAlias Bar.Example.Com
</VirtualHost>
```

Taka konfiguracja bywa przydatna, gdy użytkownicy sieci wewnętrznej mają móc korzystać z serwera *Foo.Example.Com* poprzez adres prywatny `10.0.0.42`, a nie adres publiczny (na przykład w celu oddzielenia na kartach sieciowych ruchu wewnętrznego od ruchu zewnętrznego). Drugi wirtualny serwer będzie otrzymywał żądania kierowane pod dwa adresy, mimo że za pomocą dyrektywy `ServerName` zdefiniowano tylko jedną nazwę serwera. Używając w takim przypadku w formacie dziennika zdarzeń dyrektywy `%A`, możliwe będzie stwierdzenie, ile żądań kierowanych do tego serwera wirtualnego przychodzi każdym interfejsem sieciowym.

## Zobacz również

- Dokumentacja modułu *mod_log_config*.

# 3.15. Rejestracja stron, z których nadchodzą żądania

## Problem

Należy rejestrować adresy URL stron, z których następują odwołania do stron własnego serwera, na przykład w celu stwierdzenia, kim są odwiedzający jego strony.

## Rozwiązanie

Do formatu dziennika zdarzeń aktywności serwera WWW należy dodać dyrektywę:

```
%{Referer}i
```

## Analiza

Jednym z pól, które może zawierać nagłówek żądania jest pole `Referer`. Jest to adres URL strony WWW, z której pochodzi aktualne żądanie. Gdy na przykład plik *a.html* zawiera łącze:

```
<a href="b.html">inna strona</a>
```

to po wybraniu tego łącza nagłówek żądania otwarcia strony *b.html* będzie zawierał pole `Referer`, w którym znajdować się będzie adres URL strony *a.html*.

Pole `Referer` nie jest polem wymaganym, a na dodatek nie jest polem wiarygodnym — niektórzy użytkownicy używają programów i narzędzi ukrywających adresy stron, które odwiedzają. Jednak tego rodzaju praktyki spotyka się dość rzadko i w przypadku większości stron można nie zwracać na to uwagi.

## Zobacz również

- Receptura 3.17.
- Receptura 6.5.

# 3.16. Rejestracja nazw używanych przeglądarek

## Problem

Należy gromadzić informacje, jakiego rodzaju przeglądarek używają odwiedzający witrynę WWW, na przykład w celu optymalizacji wyglądu jego stron pod kątem możliwości przeglądarek używanych przez większość odwiedzających.

## Rozwiązanie

Do formatu dziennika zdarzeń aktywności serwera WWW należy dodać dyrektywę:

```
%{User-agent}i
```

## Analiza

Nagłówki żądań często zawierają pole `User-agent`. Pole to przechowuje nazwę oraz wersję oprogramowania klienta używanego do zgłoszenia żądania. Pole `User-agent` może, na przykład, wyglądnąć następująco:

```
User-Agent: Mozilla/4.77 [en] (X11; U; Linux 2.4.4-4GB i686)
```

W tym przypadku klient twierdzi, że jest programem Netscape Navigator 4.77 działającym w systemie Linux i używającym interfejsu GUI w postaci X-window.

Pole `User-agent` nie jest polem wymaganym, a na dodatek nie jest polem wiarygodnym — wielu użytkowników używa programów i narzędzi ukrywających informacje o używanych przez nich przeglądarkach. Niektóre programy, informując o swojej tożsamości, kłamią po to, by na przykład móc korzystać ze stron WWW przeznaczonych tylko dla wybranych rodzajów przeglądarek. Niektórzy dość dziwnie sądzą, że administratora strony WWW nie powinno interesować, jakiego rodzaju przeglądarki używają. Jest to jeden z powodów, dla którego strony WWW powinny być uniezależnione od rodzaju przeglądarek WWW na tyle, na ile to jest tylko możliwe. Jeżeli na podstawie tego pola mają być podejmowane jakieś decyzje, należy założyć, że jego zawartość odpowiada prawdzie, gdyż nie ma żadnego sposobu stwierdzenia, że tak nie jest.

## Zobacz również

- Receptura 3.17.

# 3.17. Rejestracja dowolnych pól nagłówka żądania

## Problem

Należy rejestrować wartości dowolnych pól nagłówka umieszczanych przez klientów w żądaniach, na przykład w celu dostosowania rodzaju oferowanej zawartości w zależności od potrzeb odwiedzających.

## Rozwiązanie

W deklaracji formatu dziennika zdarzeń aktywności serwera WWW należy umieścić zmienną formatu `%{...}i`. Aby, na przykład, rejestrować zawartość pola `Host` nagłówka, należy użyć zmiennej:

```
%{Host}i
```

## Analiza

Żądanie HTTP wysłane przez przeglądarkę WWW może mieć bardzo złożoną postać, a gdy klient nie jest przeglądarką, lecz specjalizowana aplikacją, może umieszczać w nim dodatkowe metadane zrozumiałe dla serwera. Na przykład jednym z bardziej przydatnych pół nagłówka żądania jest pole `Accept` informujące serwer, jaki rodzaj zawartości klient może i chce otrzymać. W przypadku wiersza `CustomLog` takiego jak:

```
CustomLog logs/accept_log "%{UNIQUE_ID}e \"%{Accept}i\""
```

odpowiadający mu zapis w dzienniku zdarzeń mógłby wyglądać następująco:

```
PNb6VsCoF2UAAH1dAUo "text/html, image/png, image/jpeg, image/gif, image/x-
xbitmap, */*"
```

Oznacza on, że klient, który zgłosił żądanie, może obsługiwać zawartość HTML oraz pewnie rodzaje plików graficznych, ale w razie potrzeby przyjmie wszystko, co serwer mu wyśle (oznaczono to za pomocą zapisu `*/*`).

## Zobacz również

- Receptura 3.15.
- Receptura 3.17.

# 3.18. Rejestracja dowolnych pól nagłówka odpowiedzi

## Problem

Należy rejestrować wartości dowolnych pól nagłówka umieszczanych przez serwer w jego odpowiedziach, na przykład w przypadku rozwiązywania problemów związanych z działaniem skryptu lub aplikacji.

## Rozwiązanie

W deklaracji formatu dziennika zdarzeń aktywności serwera WWW należy umieścić zmienną formatu `%{...}o`. Aby, na przykład, rejestrować zawartość pola `Last-Modified` nagłówka, należy użyć zmiennej:

```
%{Last-Modified}o
```

## Analiza

Odpowiedź HTTP wysyłana przez serwer Apache może mieć bardzo złożoną postać — zależy to od konfiguracji serwera. Rozbudowane skrypty lub serwery aplikacji mogą dodawać do nagłówków odpowiedzi serwera niestandardowe pola, a znajomość umieszczanych w nich wartości może być bardzo przydatna podczas śledzenia problemów ze skryptem bądź aplikacją.

Poza faktem, że rejestrowane są zawartości pól, które serwer *wysyła*, a nie odbiera, receptura ta jest taka sama jak receptura 3.17 i tam należy szukać więcej szczegółów. Jedyna różnica w składni dyrektyw formatu rejestracji jest taka, że pola odpowiedzi są rejestrowane za pomocą dyrektywy `o`, a pola żądań — za pomocą dyrektywy `i`.

## Zobacz również

- Receptura 3.17.

# 3.19. Rejestracja aktywności serwera w bazie danych MySQL

## Problem

Zamiast zapisywać zdarzenia związane z dostępem do serwera WWW w płaskich plikach tekstowych, informacje te powinny trafiać wprost do bazy danych. Ułatwi to ich późniejszą analizę.

## Rozwiązanie

Ze strony *http://www.grubbybaby.com/mod_log_sql/* należy pobrać oraz zainstalować (posługując się wskazówkami z receptury 2.1) najnowszą wersję modułu *mod_log_sql*. Następnie należy wydać polecenia:

```
# mysqladmin create apache_log
# mysql apache_log < access_log.sql
# mysql apache_log
mysql> grant insert,create on apache_log.* to webserver@localhost identified
by 'wwwpw';
```

Do pliku *httpd.conf* należy dodać następujące wiersze:

```
<IfModule mod_log_mysql.c>
    MySQLLoginInfo localhost webserver wwwpw
    MySQLDatabase apache_log
    MySQLTransferLogTable access_log
    MySQLTransferLogFormat huSUsbTvRA
</IfModule>
```

## Analiza

Warto zauważyć, że nazwy strony WWW, pliku archiwum tar oraz modułu różnią się. Plik modułu oraz jego katalog zawierają nazwę „mysql", ale katalog strony WWW oraz nazwa pliku archiwum tar zawierają już bardziej ogólną nazwę „sql".

## Zobacz również

- *http://www.grubbybaby.com/mod_log_sql/*.

# 3.20. Rejestracja zdarzeń w dzienniku systemowym

## Problem

Informacje o zdarzeniach mają trafiać do dziennika systemowego (syslog) systemu operacyjnego.

## Rozwiązanie

Żeby do dziennika systemowego trafiały komunikaty o błędach, należy użyć dyrektywy:

```
ErrorLog syslog:user
```

W niektórych środowiskach zamiast klasy rejestracji `user` bardziej właściwa może okazać się klasa `local7`.

Żeby do dziennika systemowego trafiały komunikaty związane z dostępem do serwera WWW, trzeba wykonać odrobinę więcej pracy. W tym celu w pliku konfiguracyjnym należy umieścić wiersz:

```
CustomLog |/usr/local/apache/bin/apache_syslog combined
```

w którym *apache_syslog* jest nazwą na przykład takiego programu:

```
#!/usr/bin/perl
use Sys::Syslog qw( :DEFAULT setlogsock );

setlogsock('unix');
openlog('apache', 'cons', 'pid', 'user');

while ($log = <STDIN>) {
        syslog('notice', $log);
}
closelog;
```

(Należy zauważyć, że skrypt ten jest tylko szablonem programu, który w wersji nadającej się do użycia powinien zawierać kontrolę błędów i tym podobne funkcje).

## *Analiza*

Istnieją dwa powody rejestrowania zdarzeń w dzienniku systemowym (syslog) systemu operacyjnego. Pierwszym z nich jest możliwość rejestracji zdarzeń wielu serwerów w centralnym systemie rejestracji zdarzeń. Drugi wynika z dużej liczby istniejących narzędzi służących do monitorowania dziennika systemowego i potrafiących wysyłać powiadomienia o pewnych zdarzeniach. Użycie tego rodzaju narzędzi przynosi całej instalacji serwera WWW wyłącznie korzyści.

Komunikaty o błędach serwer Apache domyślnie kieruje do dziennika systemowego. Jest to zdecydowanie najlepsze dla nich miejsce, gdyż przeważnie trafiają tam informacje o błędach, a rzadziej zwykłe komunikaty informacyjne.

Składnia dyrektywy `ErrorLog` umożliwia użycie jako jej parametru dziennika systemowego (`syslog`) oraz wyszczególnienie jego klasy. W poprzednim przykładzie posłużono się klasą `user`. W pliku */etc/syslog.conf* należy zdefiniować, gdzie w przypadku danej klasy powinny być kierowane zapisy dziennika — do pliku czy do zdalnego serwera przechowującego centralne dzienniki zdarzeń.

Ponieważ serwer Apache nie rejestruje domyślnie w dzienniku systemowym informacji związanych z dostępem do serwera, należy do tego celu użyć dyrektywy wykorzystującej potok. Zastosowany do tego celu prosty, napisany w języku Perl program korzysta z modułu *Sys::Syslog* będącego standardowym modułem instalacji języka Perl. Ponieważ używany w potoku program obsługi pliku dziennika zdarzeń uruchamiany jest wraz z serwerem i oczekuje danych jedynie na wejściu STDIN, użycie języka Perl nie ma wpływu na wydajność pracy serwera.

Mając kilka serwerów WWW i chcąc, by wszystkie one kierowały komunikaty o zdarzeniach do centralnego dziennika zdarzeń, należy skonfigurować je tak, by kierowały komunikaty o zdarzeniach do dziennika systemowego, a następnie skierować użytą w tym celu klasę do centralnego serwera. Należy pamiętać, że w takim przypadku pozycje w centralnym dzienniku zdarzeń mogą nie pojawiać się w poprawnej kolejności, co nie jest aż wcale takie ważne, ale początkowo wygląda dość dziwnie. Skutki tego efektu można zmniejszyć, synchronizując zegary serwerów za pomocą protokołu NTP.

Więcej szczegółów na temat konfiguracji sieciowego serwera syslog można znaleźć w dokumentacji programu *syslogd*.

## *Zobacz również*

- Strony dokumentacji man programu *syslogd* oraz pliku *syslog.conf*.

# 3.21. Rejestracja katalogów użytkowników

## Problem

Każda witryna WWW katalogu użytkownika (czyli witryna dostępna pod adresem *http://nazwa_serwera/~nazwa_użytkownika*) ma posiadać swój dziennik zdarzeń.

## Rozwiązanie

W pliku *httpd.conf* należy umieścić dyrektywę:

```
CustomLog "|/usr/local/apache/bin/userdir_log" combined
```

Następnie w pliku */usr/local/apache/bin/userdir_log* należy umieścić następujący kod:

```
#!/usr/bin/perl

my $L = '/usr/local/apache/logs'; # Katalog dzienników zdarzeń

my %is_open = ( ); # Zapamiętanie uchwytu pliku
$|=1;
open(F, ">>$L/access_log"); # Domyślny dziennik błędów

while (my $log = <STDIN>) {
    if ($log =~ m!\s/~(.*?)/!) {
        my $u = $1;
        unless ($is_open{$u}) {
            my $fh;
            open $fh, '>>' . $L . '/'. $u;
            $is_open{$u} = $fh;
        }
        select ($is_open{$u});
        $|=1;
        print $log;
    }
    else {
        select F;
        $|=1;
        print F $log;
    }
}

close F;
foreach my $h (keys %is_open) {
    close $h;
}
```

(Należy zauważyć, że skrypt ten jest tylko szablonem programu, który w wersji nadającej się do użycia powinien zawierać kontrolę błędów i tym podobne funkcje).

# *Analiza*

Zwykle żądania kierowane do witryn WWW użytkowników rejestrowane są w głównym dzienniku zdarzeń serwera WWW, bez rozróżniania poszczególnych witryn. Z tego powodu bardzo trudno jest użytkownikowi odnaleźć komunikaty dotyczące jego osobistej witryny WWW.

Za pomocą przedstawionego tu rozwiązania informacje o żądaniach kierowane są do dzienników zdarzeń poszczególnych użytkowników. Program można oczywiście zmodyfikować tak, by wszystkie komunikaty trafiały do indywidualnych dzienników oraz jednocześnie do dziennika głównego.

Uchwyty plików są zapamiętywane w celu zmniejszenia obciążenia dysków, które powstawałoby na wskutek otwierania i zamykania plików przy każdym żądaniu dostępu. Sprawia to jednak, że zawsze otwartych jest jednocześnie wiele plików, co w przypadku serwerów obsługujących bardzo dużo witryn WWW użytkowników może powodować problemy z dostępnością zasobów systemowych.

Ponieważ interpreter Perla domyślnie buforuje swoje wyjście, należało nakazać skryptowi wyłączyć to buforowanie, dzięki czemu informacje o zdarzeniach będą mogły trafiać do pliku natychmiast. W tym celu zmiennej odpowiedzialnej za automatyczne opróżnianie buforów (`$|`) należy nadać wartość `true`. Informuje to interpreter Perla, aby nie buforował wyjścia prowadzącego do ostatnio używanego pliku. Bez podjęcia tego środka zaradczego wyjście byłoby buforowane i mogłoby się wydawać, że w plikach dzienników zdarzeń nic nie jest zapisywane.

# *Zobacz również*

- *http://httpd.apache.org/docs/mod/mod_log_config.html*.
- *http://httpd.apache.org/docs-2.0/mod/mod_log_config.html*.