Wykład 12

Serwery HTTP

wykład prowadzi: Maciej Zakrzewicz

Plan wykładu

- Wprowadzenie do architektury serwera HTTP Apache
- Podstawowe parametry konfiguracyjne
- Dziennik serwera
- Ścieżki logiczne i fizyczne
- Dyrektywy blokowe
- Ochrona dostępu

Celem wykładu jest przegląd własności funkcjonalnych serwerów HTTP na przykładzie serwera Apache. W ramach wykładu omówimy wewnętrzną architekturę serwera Apache, jego podstawowe parametry konfiguracyjne, strukturę i zastosowania dziennika serwera, mechanizmy odwzorowania logicznych ścieżek dostępu w ścieżki fizyczne, dyrektywy blokowe oraz funkcje ochrony dostępu do dokumentów znajdujących się po stronie serwera Apache.

Serwer Apache

- Najpopularniejszy serwer HTTP
- Projektu open-source
- Tekstowy plik konfiguracyjny httpd.conf
- Uruchamianie, zatrzymywanie:
- `apachectl start`
- `apachectl restart`
- `apachectl stop`

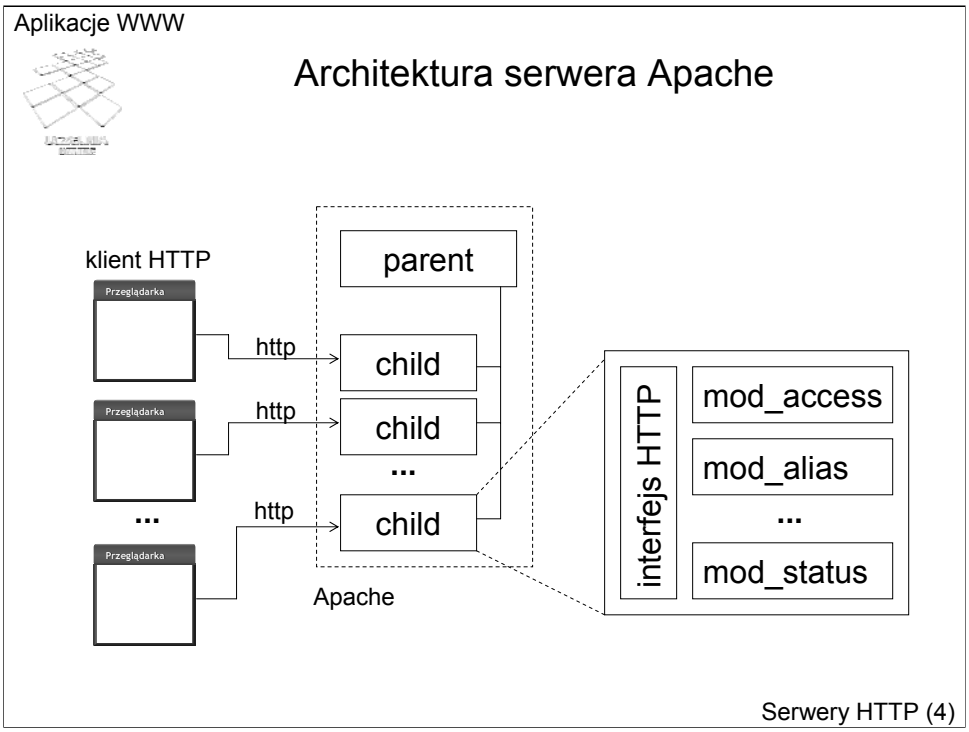
Apache (www.apache.org) to najpopularniejszy dziś serwer HTTP, rozwijany w ramach projektu open-source. Wywodzi się on z pierwszego serwera HTTP autorstwa Tima Bernersa-Lee. Dziś jest uznawany za doskonały przykład i wzorzec dla implementacji innych serwerów HTTP. Swoją popularność zawdzięcza w dużej mierze obecności w dystrybucyjnej wersji systemu Linux, jednak dostępne są też bezpłatne wydania serwera Apache dla rozmaitych platform operacyjnych.

Apache posługuje się pojedynczym plikiem konfiguracyjnym o nazwie "httpd.conf". Do jego uruchamiania i zatrzymywania służy skrypt o nazwie "apachectl". Na slajdzie przedstawiono przykładowe komendy skryptu "apachectl":

•apachectl start - powoduje uruchomienie serwera HTTP Apache

•apachectl restart - powoduje zatrzymanie i ponowne uruchomienie serwera HTTP Apache

•apachectl stop - powoduje zatrzymanie serwera HTTP Apache



Serwer Apache posiada architekturę wieloprocesową i wielomodułową. Na poziomie systemu operacyjnego serwer Apache jest reprezentowany przez jeden proces nadrzędny, tzw. parent process, i wiele procesów podrzędnych, tzw. child processes. Zadaniem procesu nadrzędnego jest prowadzenie nieprzerwanego nasłuchu sieciowego w oczekiwaniu na żądania nadchodzące od klientów HTTP. Gdy proces nadrzędny otrzymuje żądanie HTTP, wtedy uruchamia proces potomny i zleca mu obsługę tego żądania, a sam powraca do nasłuchu. Proces potomny przetwarza żądanie HTTP i wysyła odpowiedź HTTP do klienta HTTP. Po przesłaniu odpowiedzi HTTP proces potomny kończy pracę. Liczba równocześnie pracujących procesów potomnych wynika z liczby równocześnie obsługiwanych żądań HTTP. Przy braku żądań liczba procesów potomnych może wynosić zero.

Często administratorzy korygują taki domyślny model zarządzania procesami serwera Apache i za pomocą stosownych parametrów konfiguracyjnych sprawiają, że w pamięci operacyjnej zawsze znajduje się niewielka pula bezczynnych procesów potomnych, gotowych przyjąć nowe żądania. Pula ta jest tworzona w chwili uruchomienia serwera Apache, a gdy wolne procesy potomne są konsumowane przez nowe żądania, dochodzi do automatycznego uruchamiania dalszych zapasowych procesów. Gdy proces potomny kończy obsługę żądania HTTP, wtedy może nie kończyć pracy lecz powrócić do puli procesów bezczynnych.

Każdy proces Apache ma budowę modułową. Składa się z modułu interfejsu HTTP, odpowiadającego za obsługę komunikacji sieciowej oraz z wielu modułów rozszerzających. Każdy moduł rozszerzający odpowiada za realizację fragmentu funkcjonalności serwera. Moduły posiadają niepowtarzalne nazwy, rozpoczynające się od słowa "mod_". Przykładowo, moduł "mod_access" odpowiada za zapisy historii żądań HTTP w pliku dziennika serwera Apache.

Architektura serwera Apache może być rozszerzana przez programistów. W oparciu o opublikowaną specyfikację interfejsów istnieje możliwość implementacji własnych modułów rozszerzających i ich dołączania do serwera. Administrator serwera Apache ma też możliwość usuwania modułów, z których funkcji nie chce korzystać.

Plik konfiguracyjny httpd.conf

```
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
ServerRoot "C:\oracle\ora90\Apache\Apache"
# PidFile: The file in which the server should record its process
# identification number when it starts.
PidFile logs/httpd.pid
# Timeout: The number of seconds before receives and sends time out.
Timeout 300
...
```


Parametry pracy serwera Apache są definiowane w jego tekstowym pliku konfiguracyjnym o nazwie "httpd.conf". Z logicznego punktu widzenia jest to jeden plik, lecz fizycznie może być podzielony na wiele fragmentów zapisanych w odrębnych plikach. Składnia zapisu parametrów w pliku konfiguracyjnym jest bardzo prosta: podajemy nazwę parametru, po niej spację, a następnie wartość. Wiersze rozpoczynające się od znaku "#" są traktowane jako komentarze. Na slajdzie przedstawiono fragment oryginalnego pliku konfiguracyjnego "httpd.conf". Widoczne są trzy parametry: ServerRoot, PidFile i Timeout. Wartością parametru Timeout jest "300". Znaczenie wybranych parametrów zostanie omówione w dalszej części wykładu.

Plik konfiguracyjny "httpd.conf" jest odczytywany tylko w chwili uruchamiania serwera. Z tego powodu ewentualne zmiany wartości parametrów wymagają jego zatrzymania i ponownego uruchomienia.

Aplikacje WWW

Podstawowe parametry konfiguracyjne (1)

- ServerRoot
- PidFile
- DocumentRoot
- ErrorLog
- StartServers
- MaxClients
- MaxSpareServers
- MinSpareServers

Serwery HTTP (6)

Na slajdzie przedstawiono wybrane podstawowe parametry konfiguracyjne serwera Apache. Ich znaczenie jest następujące:

• ServerRoot - to nazwa katalogu domowego serwera Apache; może być stosowany jako ścieżka odniesienia dla innych parametrów określających lokalizację plików,

• PidFile - to lokalizacja pliku, w którym nadrzędny proces serwera zapisuje swój własny identyfikator procesu w systemie operacyjnym aby umożliwić procesom potomnym odnalezienie procesu nadrzędnego,

• DocumentRoot - to nazwa katalogu, w którym znajdują się dokumenty HTML udostępniane przez serwer klientom HTTP,

• ErrorLog - to lokalizacja pliku, w którym zapisywane są komunikaty o błędach obsługi żądań i błędach wewnętrznych serwera Apache,

• StartServers - liczba procesów potomnych serwera, automatycznie uruchamianych podczas jego startu,

• MaxClients - to maksymalna liczba żądań, jakie mogą być obsługiwane współbieżnie - jest to więc maksymalna liczba procesów potomnych, jakie mogą pracować równocześnie,

• MaxSpareServers - to maksymalna liczba potomnych procesów serwera, jakie mogą pozostawać bezczynne; po przekroczeniu tej liczby nadmiarowe procesy potomne są zatrzymywane,

• MinSpareServers - to minimalna liczba potomnych procesów serwera, jakie muszą pozostawać bezczynne w ramach gotowości do przyjmowania nowych żądań.

Aplikacje WWW

Podstawowe parametry konfiguracyjne (2)

- KeepAlive
- KeepAliveTimeout
- MaxKeepAliveRequests
- Port
- Listen
- CustomLog
- LogFormat
- DirectoryIndex

Serwery HTTP (7)

Na slajdzie przedstawiono kolejne parametry konfiguracyjne serwera Apache. Ich znaczenie jest następujące:

• KeepAlive - uaktywnia obsługę mechanizmu Persistent Connections protokołu HTTP 1.1,

• KeepAliveTimeout - określa maksymalny czas oczekiwania procesu potomnego serwera podtrzymującego połączenie HTTP 1.1 Persistent Connection, po upływie którego następuje zamknięcie połączenia HTTP; jest to mechanizm ochrony przed atakami typu Denial of Service (DoS),

• MaxKeepAliveRequests - określa maksymalną liczbę żądań, jakie mogą zostać obsłużone w ramach jednego połączenia HTTP 1.1 Persistent Connection; to również jest mechanizm ochrony przed atakami typu Denial of Service (DoS),

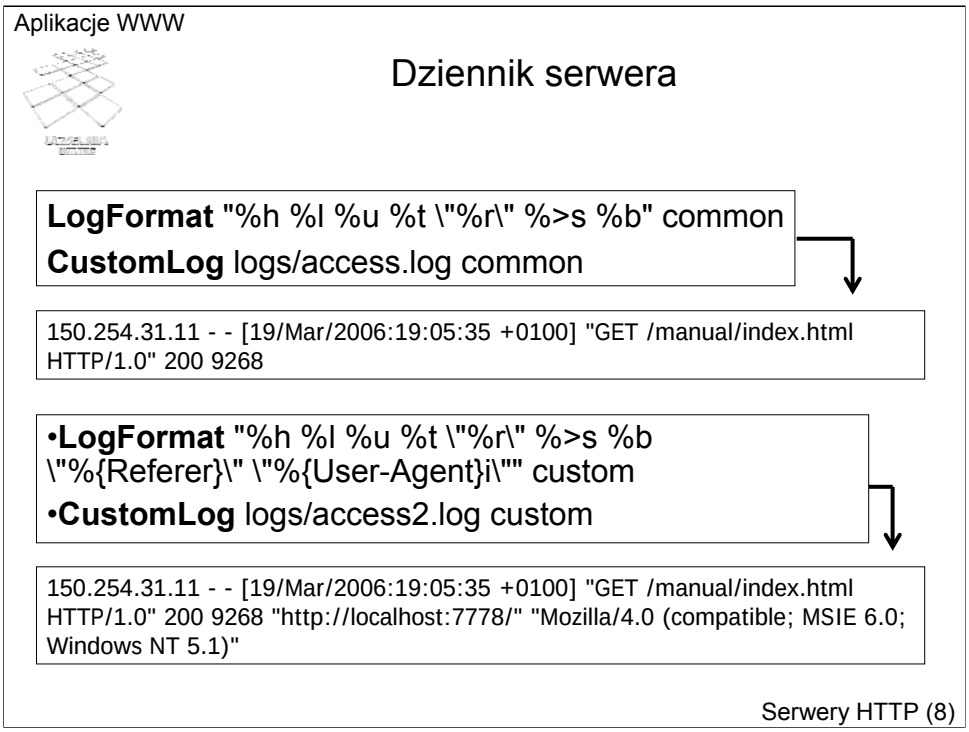
• Port - to numer głównego portu TCP, na którym serwer nasłuchuje połączeń HTTP; domyślnie jest to port 80,

• Listen - to alternatywne adresy IP i numery portów, na których serwer Apache nasłuchuje żądań HTTP,

• CustomLog - to lokalizacja pliku dziennika, w którym rejestrowane są wszystkie żądania HTTP otrzymane przez serwer Apache,

• LogFormat - określa format rekordów zapisywanych do pliku dziennika serwera Apache,

• DirectoryIndex - to nazwa pliku, który zostanie przesłany w odpowiedzi na żądanie HTTP zawierające niepełny adres URL, np. http://www.poznan.com/dir/ (brak nazwy pliku).



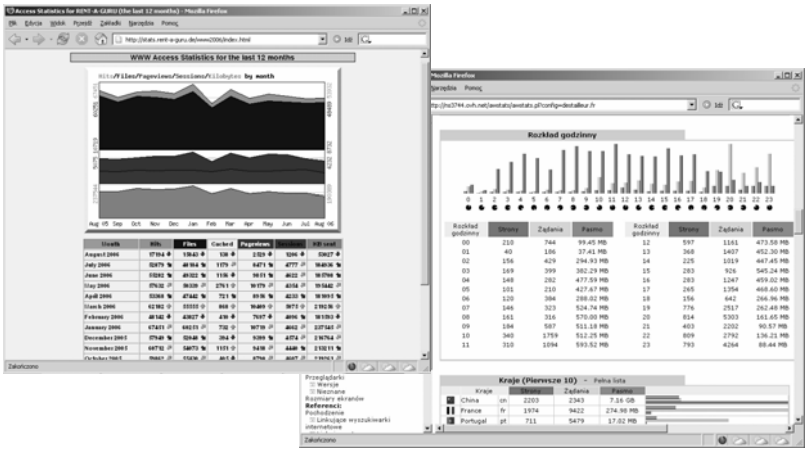
Ważnym elementem serwera Apache jest mechanizm dziennika (ang. web log). Każde otrzymane żądanie HTTP jest odnotowywane w pliku dziennika wraz z m.in. adresem IP klienta HTTP, znacznikiem czasowym i statusem odpowiedzi serwera HTTP. Informacje te mogą być zapisywane w formacie konwencjonalnym, nazywanym Common Log Format (CLF) lub w formacie zdefiniowanym przez administratora serwera. Do definiowania własnego formatu wierszy służy parametr LogFormat, natomiast lokalizację i nazwę pliku dziennika określa parametr CustomLog. Znaczenie symboli używanych podczas definiowania formatu pliku dziennika jest następujące:

- %h: adres IP klienta HTTP
- %l: nazwa użytkownika użyta podczas ew. uwierzytelniania
- %u: nazwa użytkownika w systemie operacyjnym klienta
- %t: czas otrzymania żądania
- %r: pierwszy wiersz nagłówka żądania HTTP
- %>s: status obsługi żądania HTTP (200=OK)
- %b: rozmiar odpowiedzi HTTP w bajtach
- %{Referer}: adres dokumentu, z którego pochodzi łącznik powodujący żądanie HTTP
- %{User-Agent}: nazwa programu klienta HTTP

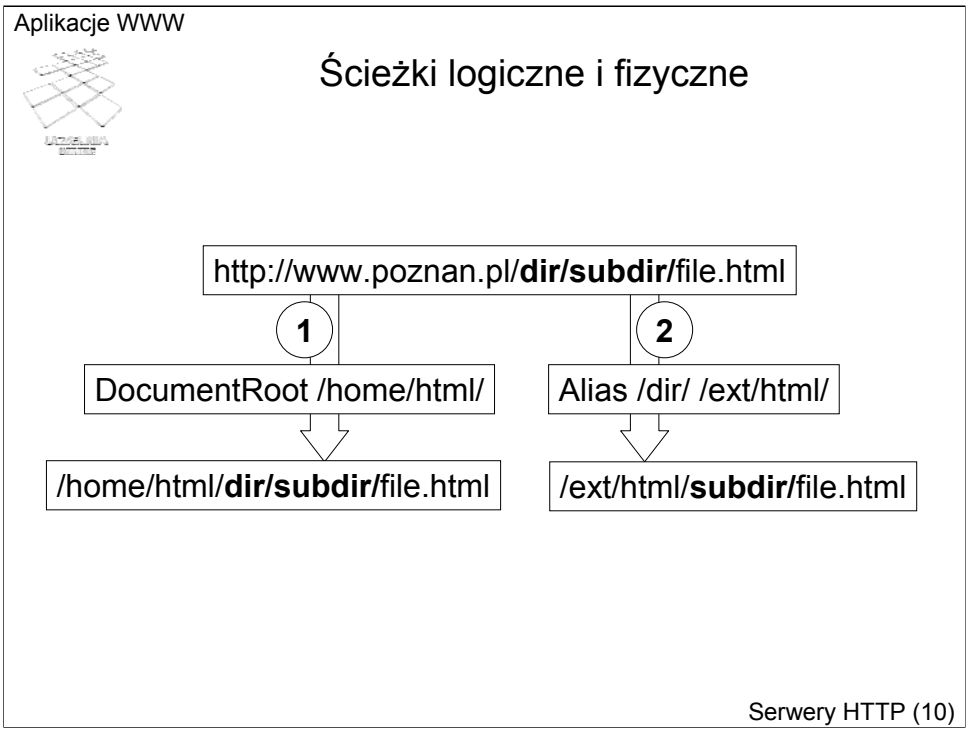
Na slajdzie przedstawiono dwie przykładowe konfiguracje plików dziennika. Pierwsza powoduje rejestrowanie żądań HTTP w pliku access.log, w formacie zgodnym z Common Log Format. Każde żądanie jest odnotowywane w postaci wiersza (rekordu) zawierającego: adres IP klienta HTTP, nazwy użytkownika użyte podczas uwierzytelniania (w przedstawianym przykładzie nie dochodziło do uwierzytelniania, stąd znaki minus), znacznik czasowy otrzymania żądania HTTP, treść pierwszego wiersza nagłówka żądania HTTP, kod zwrotny (OK) i rozmiar udzielonej odpowiedzi HTTP w bajtach.

Druga przykładowa konfiguracja powoduje rejestrowanie żądań HTTP w pliku access2.log, w rozszerzonym formacie, obejmującym dodatkowo adres dokumentu, z którego nastąpił skok do dokumentu będącego przedmiotem żądania HTTP oraz nazwę programu klienta HTTP, używanego przez użytkownika końcowego.

Analiza plików dziennika



Pliki dziennika serwerów HTTP stanowią interesujące źródło informacji o charakterystyce obciążenia, głównych odbiorcach, najpopularniejszych dokumentach, typowych ścieżkach nawigacyjnych, itp. Niestety, manualna analiza zawartości plików dziennika zapisanych np. w formacie CLF jest niezwykle trudna za względu na ogromną liczbę wierszy opisujących żądania HTTP użytkowników. W praktyce analizę plików dziennika wykonuje się z użyciem automatycznych narzędzi, które prezentują wyniki w postaci tabel, rankingów i wykresów graficznych. Popularnymi narzędziami do analizy plików dziennika (ang. web analyzers) są: AWStats (awstats.sourceforge.net), HTTP-Analyze (http-analyze.org), Webalizer (www.webalizer.com).



Adresy URL zamieszczane w żądaniach HTTP specyfikują nazwę i ścieżkę dostępu do dokumentu. Ścieżka dostępu ma charakter ścieżki logicznej, tzn. nie opisuje rzeczywistego położenia dokumentu w systemie plików serwera HTTP. Zadaniem serwera HTTP jest odwzorowanie ścieżki logicznej, użytej w adresie URL, na ścieżkę fizyczną w systemie plików. Odwzorowanie to odbywa się za pomocą jednej z dwóch metod:

1. Ścieżka fizyczna powstaje poprzez połączenie wartości parametru DocumentRoot i ścieżki logicznej,

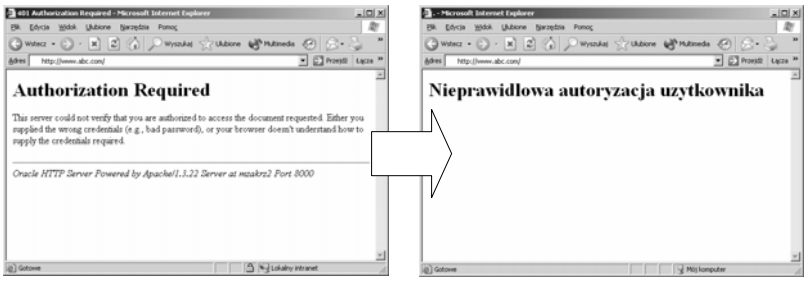
2. Ścieżka fizyczna jest zdefiniowana za pomocą parametru Alias jako odwzorowanie ścieżki logicznej.

Przykład przedstawiony na slajdzie obrazuje oba sposoby generowania fizycznej ścieżki dostępu do żądanego pliku. W pierwszym przypadku, katalogiem głównym jest "/home/html/", a użyta w adresie URL ścieżka "/dir/subdir/" jest traktowana jako względna. Stąd wynikowa ścieżka fizyczna to "/home/html/dir/subdir/". W drugim przypadku, administrator serwera Apache zdefiniował powiązanie ścieżki logicznej "/dir/" z katalogiem fizycznym "/ext/html". Użyta w adresie URL ścieżka "/dir/subdir/" zmienia się więc na fizyczny podkatalog "/subdir/" wewnątrz katalogu "/ext/html/".

Aplikacje WWW

Parametr ErrorDocument

```
ErrorDocument 404 "Przepraszamy, nie posiadamy dokumentu o takim URL
ErrorDocument 401 /messages/accessdenied.html
```



Serwery HTTP (11)

Jeżeli obsługa żądania kończy się niepowodzeniem, serwer HTTP umieszcza w nagłówku odpowiedzi HTTP kod zwrotny błędu. Kod ten jest wykorzystywany przez program klienta HTTP do wyświetlenia stosownego komunikatu dla użytkownika końcowego. Serwer Apache daje jednak możliwość podmiany kodu błędu na kompletny poprawny dokument opisujący problem, który wystąpił. Wówczas program klienta HTTP potraktuje swoje żądanie jako obsłużone właściwie i wyświetli otrzymany dokument.

Do zastępowania standardowych kodów zwrotnych dokumentami opisowymi służy parametr ErrorDocument. Parametr ten jest dwuwartościowy. Pierwsza wartość to podmieniany kod zwrotny, a druga to nazwa pliku dokumentu lub wręcz treść dokumentu opisowego. W przykładzie na slajdzie dokonano dwóch podmian standardowych kodów zwrotnych:

•jeżeli obsługa żądania HTTP zakończy się kodem 404, wtedy klient HTTP otrzyma dokument tekstowy o treści "Przepraszamy, nie posiadamy dokumentu o takim URL",

•jeżeli obsługa żądania HTTP zakończy się kodem 401, wtedy klient HTTP otrzyma dokument HTML o nazwie "accessdenied.html" znajdujący się w katalogu "/messages".

Dyrektywy blokowe

- <Directory>
- <DirectoryMatch>
- <Files>
- <FilesMatch>
- <Location>
- <LocationMatch>
- <VirtualHost>

Domyślnie, parametry konfiguracyjne serwera Apache są jednowersyjne i mają zasięg globalny. Istnieje jednak możliwość zawężenia ich zasięgu i wprowadzenia wielowersyjności. Do tego celu służą tzw. dyrektywy blokowe. Dyrektywy blokowe mają postać znaczników, przypominających znaczniki HTML. Kiedy znaczniki dyrektywy blokowej otaczają grupę parametrów, to parametry te są aktywne wyłącznie wtedy, gdy spełniony jest warunek opisujący dyrektywę blokową. Na slajdzie przedstawiono najpopularniejsze dyrektywy blokowe:

• <Directory>: ogranicza zasięg parametrów do żądań dotyczących nazwanego katalogu fizycznego i jego wszystkich podkatalogów,

• <DirectoryMatch>: jak wyżej, lecz zamiast nazwy katalogu podawane jest wyrażenie regularne,

• <Files>: ogranicza zasięg parametrów do żądań dotyczących plików spełniających podany wzorzec nazwy,

• <FilesMatch>: jak wyżej, lecz zamiast wzorca nazwy pliku podawane jest wyrażenie regularne,

• <Location>: ogranicza zasięg parametrów do żądań dotyczących nazwanego katalogu wirtualnego,

• <LocationMatch>: jak wyżej, lecz zamiast nazwy katalogu wirtualnego podawane jest wyrażenie regularne,

• <VirtualHost>: ogranicza zasięg parametrów do żądań dotyczących podanego serwera wirtualnego.

Przykłady użycia dyrektyw blokowych zostaną przedstawione na kolejnych slajdach.

Na slajdzie przedstawiono przykład użycia dyrektyw blokowych do definicji wielu tzw. wirtualnych serwerów HTTP. Idea serwerów wirtualnych opiera się na emulowaniu obecności wielu komputerów posiadających różne adresy sieciowe, wyposażonych w odrębne serwery HTTP, posiadające odmienne ustawienia konfiguracyjne. W celu uzyskania takiego efektu należy:

1. Przypisać posiadanemu komputerowi wiele adresów DNS, skojarzonych z tym samym adresem IP.

2. Przygotować wiele alternatywnych zestawów ustawień parametrów konfiguracyjnych.

3. Otoczyć alternatywne zestawy ustawień parametrów konfiguracyjnych znacznikami dyrektyw blokowych VirtualHost.

4. Wewnątrz zestawów parametrów umieścić parametry ServerName, skojarzone z adresami DNS, dla których zestawy te mają być aktywowane.

Serwer Apache skonfigurowany zgodnie z przykładem przedstawionym na slajdzie będzie się zachowywać następująco. Gdy użytkownik wystosuje żądanie pobrania pliku "index.html" z serwera "www.abc.com", wtedy żądanie to zostanie obsłużone zgodnie z następującymi ustawieniami parametrów konfiguracyjnych:

•DocumentRoot "C:\serv1\htdocs"

•CustomLog "C:\serv1\logs\access.log" common

•ErrorLog "C:\serv1\logs\error.log"

W związku z tym, użytkownik końcowy otrzyma w odpowiedzi plik "index.html" z fizycznego katalogu "C:\serv1\htdocs", a żądanie to zostanie odnotowane w pliku dziennika "access.log" umieszczonego w katalogu "C:\serv1\logs".

Jeżeli natomiast użytkownik wystosuje żądanie pobrania pliku "index.html" z serwera "www.xyz.com", wtedy żądanie to zostanie obsłużone zgodnie z następującymi ustawieniami parametrów konfiguracyjnych:

•DocumentRoot "C:\serv2\htdocs"

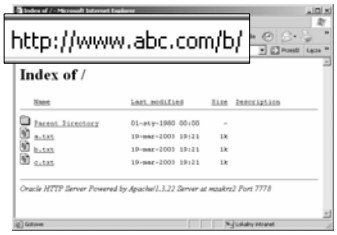
•CustomLog "C:\serv2\logs\access.log" common

•ErrorLog "C:\serv2\logs\error.log"

W związku z tym, użytkownik końcowy otrzyma w odpowiedzi plik "index.html" z fizycznego katalogu

Dyrektywy blokowe - przykład (2)

```
DirectoryIndex index.html
DocumentRoot C:\serv1\htdocs
<Directory C:\serv1\htdocs\a>
   Options -Indexes
</Directory>
<Directory C:\serv1\htdocs\b>
   Options +Indexes
</Directory>
```



Kolejny przykład użycia dyrektyw blokowych został przedstawiony na slajdzie. Przykład dotyczy obsługi żądań opisanych niepełnym adresem URL, tj. adresem URL, w którym brakuje nazwy żądanego dokumentu. Domyślne zachowanie się serwera Apache polega w takim przypadku na przekazaniu klientowi HTTP tzw. dokumentu domyślnego, czyli dokumentu o nazwie zdefiniowanej za pomocą parametru DirectoryIndex. Dokument ten jest pobierany z katalogu opisanego w adresie URL.

Jeżeli we wspomnianym katalogu nie znajduje się dokument domyślny, możliwe są dwa warianty dalszej obsługi:

1. Żądanie kończy się błędem - klient HTTP otrzymuje kod zwrotny Forbidden.

2. Klient HTTP otrzymuje automatycznie wygenerowany dokument zawierający listę plików znajdujących się w katalogu.

Do wyboru powyższych wariantów służy parametr Options. Wartość "-Indexes" oznacza zakaz prezentacji zawartości katalogu, a wartość "+Indexes" oznacza, że lista plików zostanie przekazana klientowi HTTP.

Przykład na slajdzie wykorzystuje parametr Options wraz z dyrektywami blokowymi w celu uzyskania następującego efektu:

1. Jeżeli żądanie HTTP dotyczy katalogu fizycznego "C:\serv1\htdocs\a", to w przypadku braku dokumentu domyślnego, odpowiedzią na żądanie będzie kod zwrotny Forbidden.

2. Jeżeli żądanie HTTP dotyczy katalogu fizycznego "C:\serv1\htdocs\b", to w przypadku braku dokumentu domyślnego, odpowiedzią na żądanie będzie automatycznie wygenerowany dokument zawierający listę plików znajdujących się w katalogu.

Aplikacje WWW

Ochrona dostępu wg adresów

- allow from
- deny from
- order

```
<Directory C:\serv1\htdocs\private>
  order deny,allow
  deny from all
  allow from 150.254.31.10 192.168.1.12
</Directory>
```

Serwery HTTP (15)

W wielu przypadkach zależy nam na ograniczeniu dostępu użytkowników do dokumentów znajdujących się po stronie serwera Apache. Jedną z możliwości rozwiązania tego problemu jest blokada dostępu do plików według adresów IP klientów HTTP. Za pomocą parametrów "allow from" i "deny from" administrator definiuje listy adresowe komputerów uprawnionych lub nieuprawnionych do dostępu do zasobów serwera Apache. Ponieważ listy te mogą się przecinać, to istotna jest kolejność ich analizowania przez serwer. O tej kolejności decyduje kolejny parametr o nazwie "order". Wszystkie te parametry umieszcza się zwykle wewnątrz dyrektyw blokowych Directory lub Location.

Przykład przedstawiony na slajdzie ma następujące znaczenie. Dla plików znajdujących się w fizycznym katalogu "C:\serv1\htdocs\private" ustala się ograniczenia dostępu wg adresów IP klientów HTTP. Definiuje się dwie listy adresów. Lista adresów, którym odmawia się dostępu obejmuje "all", czyli wszystkie możliwe adresy. Lista adresów, którym zezwala się na dostęp obejmuje dwa adresy IP: 150.254.31.10 i 192.168.1.2. Serwer Apache przeanalizuje te listy w następującej kolejności: "deny,allow" - najpierw listę zakazów, potem listę zezwoleń. W rezultacie zakaz dostępu do plików będzie obejmować wszystkie możliwe adresy IP z wyłączeniem adresów 150.254.31.10 i 192.168.1.2.

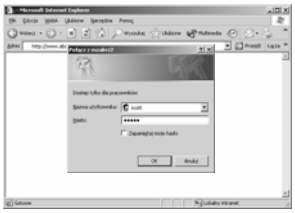
Aplikacje WWW

Ochrona dostępu poprzez uwierzytelnianie użytkowników

Utworzenie pliku haseł

```
htpasswd -c c:\pwd\users.pwd scott
Automatically using MD5 format.
New password: *****
Re-type new password: *****
Adding password for user scott

htpasswd c:\pwd\users.pwd guest
Automatically using MD5 format.
New password: ******
Re-type new password: ******
Adding password for user guest
...
```



Ochrona dostępu do katalogu

```
<Directory C:\serv1\htdocs>
  AuthName "Dostęp tylko dla pracowników"
  AuthType Basic
  AuthUserFile "C:\pwd\users.pwd"
  Require user scott
</Directory>
```

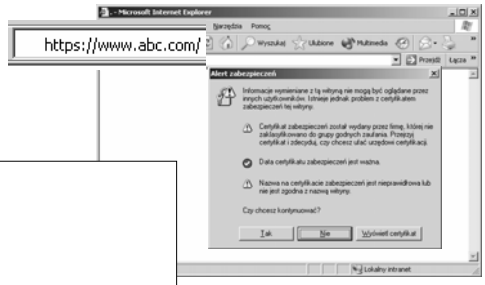
Serwery HTTP (16)

Inną formą ochrony dostępu do dokumentów znajdujących się po stronie serwera Apache jest uwierzytelnianie użytkowników końcowych. W celu skorzystania z tego mechanizmu administrator serwera musi przygotować tzw. plik haseł użytkowników, w którym będą znajdować się nazwy i hasła użytkowników końcowych. Do tworzenia i modyfikacji pliku haseł służy narzędzie "htpasswd", którego przykład użycia przedstawiono na slajdzie. Pierwsze wywołanie narzędzia "htpasswd" posługuje się parametrem "-c", wskazującym, że ma zostać utworzony nowy plik haseł. Kolejne parametry wywołania to nazwa pliku haseł i nazwa pierwszego użytkownika, którego hasło zostanie umieszczone w pliku haseł. Narzędzie "htpasswd" pobiera wtedy hasło użytkownika, transformuje je algorytmem MD5 i zapisuje w pliku. Kolejne wywołania narzędzia "htpasswd" są już pozbawione parametru "-c" i służą dopisywaniu następnych użytkowników i ich haseł.

Po utworzeniu pliku haseł możemy go wykorzystać do kontroli dostępu użytkowników do plików znajdujących się po stronie serwera Apache. W tym celu parametrowi konfiguracyjnemu AuthType nadajemy wartość "Basic", za pomocą parametru "AuthUserFile" wskazujemy lokalizację utworzonego wcześniej pliku haseł, a za pomocą parametru "Require user" określamy którzy użytkownicy z pliku haseł będą uprawnieni do pobierania plików, oczywiście pod warunkiem pomyślnego uwierzytelnienia. Parametr "AuthName" definiuje informacyjny komunikat tekstowy wyświetlany w oknie logowania.

Na slajdzie przedstawiono przykład konfiguracji takiej metody kontroli dostępu. Do plików znajdujących się w fizycznym katalogu "C:\serv1\htdocs" dostęp ma wyłącznie użytkownik o nazwie "scott", który musi w tym celu podać hasło zgodne z zapisem w pliku haseł o nazwie "C:\pwd\users.pwd". Okienko do logowania zwraca się do użytkownika z komunikatem "Dostęp tylko dla pracowników".

Połączenia HTTPS



Aktywacja modułu SSL

```
<VirtualHost 150.254.31.10>
    ServerName www.abc.com
    DocumentRoot "C:\serv1\htdocs"
    SSLEngine On
    SSLCertificateFile conf\ssl.crt\server.crt
    SSLCertificateKeyFile conf\ssl.key\server.key
</VirtualHost>
```


Serwer Apache obsługuje również połączenia HTTPS inicjowane przez klientów HTTP. Aby połączenie HTTPS mogło być nawiązane, należy skonfigurować i uaktywnić jego moduł kryptograficzny nazywany mod_ssl. Do aktywacji modułu mod_ssl służy parametr SSLEngine, któremu należy nadać wartość "On". Ponadto, serwer Apache musi zostać wyposażony w dwa dodatkowe pliki: plik klucza prywatnego i plik certyfikatu X.509. Lokalizację tych plików wskazujemy za pomocą parametrów SSLCertificateFile i SSLCertificateKeyFile.

Przykład przedstawiony na slajdzie obrazuje aktywację połączeń HTTPS dla jednego serwera wirtualnego. Połączenia z pozostałymi serwerami wirtualnymi będą nadal odbywać się z użyciem standardowego protokołu HTTP.

Podsumowanie

- Modułowa architektura
- Wielowariantowa konfiguracja
- Katalogi logiczne i fizyczne
- Mechanizmy ochrony dostępu

Serwer Apache jest najpopularniejszym serwerem HTTP dostępnym na rynku. Posiada modułową, rozszerzalną architekturę, która ułatwia dostosowywanie jego funkcjonalności do specyficznych wymagań systemowych. Parametry konfiguracyjne serwera Apache mogą być wielowersyjne, dobierane do charakteru żądania HTTP, a dzięki temu możliwa jest m.in. konstrukcja serwerów wirtualnych. Ścieżki dostępu umieszczane w adresach URL mają charakter ścieżek logicznych, odwzorowywanych w ścieżki fizyczne na podstawie parametrów konfiguracyjnych serwera Apache. Standardowe moduły serwera Apache oferują trzy mechanizmy ochrony dostępu: wg adresów IP, przez uwierzytelnianie użytkownika i przez zastosowanie protokołu HTTPS.

Materiały dodatkowe

- "Apache HTTP Server Documentation", http://httpd.apache.org/docs/
- "Log File Formats", http://www.summary.net/manual/log_formats.html

•"Apache HTTP Server Documentation", http://httpd.apache.org/docs/
•"Log File Formats", http://www.summary.net/manual/log_formats.html