helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

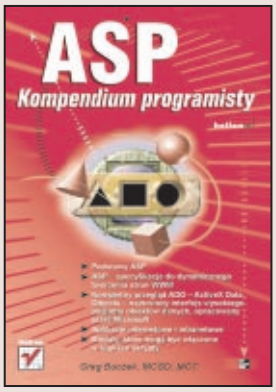
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

ASP. Kompendium programisty



Autor: Greg Buczek
Tłumaczenie: Jacek Banaszkiewicz, Paweł Koronkiewicz
ISBN: 83-7197-574-0
Tytuł oryginału: ASP Developer's Guide
Format: B5, stron: 700
Przykłady na ftp: 94 kB

ASP (Active Server Pages) dostarcza twórcom stron WWW środki do uruchamiania witryn z dynamiczną zawartością, sterowaną bazą danych. Kod, który tworzy tę bogatą zawartość, znajduje się w całości po stronie serwera, to znaczy, że jest na nim uruchamiany. Korzyścią płynącą z tworzenia skryptów po stronie serwera jest fakt, że wynikiem końcowym jest czysty HTML. Nie musisz martwić się o to, czy sprzęt odwiedzającego Twoją stronę będzie zgodny z językiem lub narzędziami, których używasz. Kolejną zaletą tworzenia skryptów po stronie serwera jest możliwość korzystania ze składników serwera bez konieczności instalacji poszczególnych programów po stronie użytkownika.
Książka ta szczegółowo prezentuje wszystkie składniki potrzebne do pracy z ASP. Po pierwsze, zdobędziemy informacje o modelu obiektu ASP. Następnie przyjrzymy się licznym składnikom potrzebnym podczas pracy z aplikacjami ASP. Później nauczymy się pracy z bazą danych oraz tworzenia składników. W końcu, w trzech ostatnich rozdziałach, użyjemy wszystkich narzędzi do stworzenia witryny WWW.

Książka została napisana w taki sposób, aby zaspokoić potrzeby zarówno tych Czytelników, którzy poznają ASP, tych, którzy są już z tą techniką zaznajomieni, jak również dla osób programujących w Visual Basicu.

Tym, którzy już są obeznani z ASP, książka ta zapewni przegląd wszystkich używanych składników, dzięki którym upewnią się, czy pamiętają wszystkie podstawowe zasady.

Dla twórców WWW książka będzie narzędziem pomagającym zrozumieć sposoby uruchamiania witryny. Zdobędą szeroką wiedzę na temat dodawania elementów dynamicznych do witryny WWW.

Osoby programujące w Visual Basicu, które chcą poszerzyć swoje horyzonty, nie zawiodą się na ASP. Ich umiejętności związane z Visual Basicem przydadzą się ponadto przy poznawaniu nowego środowiska programistycznego.

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

Spis treści

O Autorze 9

Przedmowa 11

Rozdział 1. Tworzenie dynamicznych aplikacji internetowych 13
Czym była sieć WWW 13
Zawartość statyczna a zawartość dynamiczna 16
Skrypty wykonywane po stronie klienta i po stronie serwera 18
Elementy dynamicznych rozwiązań internetowych 25
Nie tylko serwer IIS i system Windows NT 39

Rozdział 2. Serwer IIS z perspektywy twórcy stron www 41
Czym jest IIS? 41
Otrzymywanie kopii IIS 42
Konsola zarządzania 42
Właściwości usług WWW 44
Witryny WWW na serwerze IIS 52
Dodawanie witryny WWW 54
Właściwości witryny WWW 57
Eksploracja witryny 60
Aplikacje ASP 63
Witryny FTP 69

Rozdział 3. Narzędzia pracy 73
Przegląd aplikacji do tworzenia stron WWW 73
Notatnik 73
FrontPage 2000 75
NetObjects Fusion 88
NetObjects ScriptBuilder 95
Microsoft Visual InterDev 6.0 110
Ostatnie słowo o narzędziach 112

Rozdział 4. Podstawy ASP ... 113
Konstrukcja kodu ASP ... 113
Skrypt w skrypcie ... 118
Zastosowanie kodu ASP ... 125

Rozdział 5. Obiekt Request ... 165
Hierarchie i model obiektowy ASP ... 165
Odbieranie informacji od odwiedzającego ... 169
Właściwość obiektu Request ... 180
Metoda obiektu Request ... 181
Obiekt Request w działaniu ... 182

Rozdział 6. Obiekt Response ... 199
Wysyłanie informacji do gości ... 199
Zbiór obiektu Response ... 200
Właściwości obiektu Response ... 203
Metody obiektu Response ... 212
Obiekt Response w działaniu ... 219

Rozdział 7. Obiekt Server ... 233
Wejście na szczyt ... 233
Właściwość obiektu Server ... 233
Metody obiektu Server ... 236
Obiekt Server w działaniu ... 244

Rozdział 8. Obiekt Session, obiekt Application oraz plik global.asa ... 253
Aplikacje ASP ... 253
Tworzenie aplikacji ASP ... 254
Obiekt Session ... 256
Obiekt Application ... 269
Plik global.asa ... 276
Aplikacje ASP w użyciu ... 286

Rozdział 9. Obiekty CDO dla Windows NT Server ... 289
Uzupełnienie strony ASP o funkcję wysyłania wiadomości e-mail ... 289
Obiekt NewMail ... 290
Zastosowanie obiektu NewMail ... 308

Rozdział 10. Składniki ASP ... 311
Obiekty zwiększające możliwości stron ASP ... 311
Składnik Browser Capabilities ... 312
Składnik Ad Rotator ... 319
Składnik Page Counter ... 327
Składnik Counters ... 335
Składnik Content Linking ... 339
Składnik Content Rotator ... 347
Składnik MyInfo ... 352

Rozdział 11. Obiekty Skryptowe ... 355
Dodatkowe składniki VBScriptu ... 355
Związek pomiędzy obiektami systemu plików ... 355
Obiekt FileSystemObject ... 357

Zbiór Drives ... 378
Obiekt Drive ... 379
Zbiór Folders ... 383
Obiekt Folder ... 385
Zbiór Files ... 389
Obiekt File ... 390
Obiekt TextStream ... 395
Obiekt Dictionary ... 398

Rozdział 12. Obsługa błędów i Script Debugger ... 403
Obsługa błędów ... 403
Script Debugger ... 416

Rozdział 13. SQL, SQL Server i Access ... 423
Zaplecze aplikacji ASP ... 423
SQL ... 423
Praca z serwerem SQL w aplikacjach ASP ... 436
Praca z programem Microsoft Access w aplikacjach ASP ... 445

Rozdział 14. ADO ... 459
Obiekty danych ActiveX ... 459

Rozdział 15. Tworzenie aktywnych składników stron ... 507
Tworzenie własnych bibliotek w języku Visual Basic ... 507

Rozdział 16. ASP w praktyce — katalog produktów ... 541
ASP w praktyce ... 541
Witryna katalogu produktów ... 542
Baza danych katalogu produktów ... 553
Aplikacja ASP ... 561

Rozdział 17. ASP w praktyce — koszyk ... 589
Koszyk ... 589
Baza danych koszyka ... 597
Aplikacja ASP ... 602

Rozdział 18. ASP w praktyce — klienci ... 619
Praca z klientami ... 619
Baza danych modułu obsługi klienta ... 627
Aplikacja ASP ... 630

Dodatek A VBScript — leksykon podręczny ... 651

Skorowidz ... 675

10.

Składniki ASP

Obiekty zwiększające możliwości stron ASP

Serwer IIS posiada rozmaite składniki, które wspierają proces tworzenia stron ASP. Jeden z nich, obiekt CDO, poznaliśmy w poprzednim rozdziale. IIS posiada również zbiór mniejszych składników, z których możesz skorzystać w kodzie podczas realizacji różnego rodzaju zadań. W rozdziale tym skoncentrujemy się na tych pomniejszych składnikach. Tabela 10.1 prezentuje je oraz pokazuje, gdzie można je wykorzystać.

Tabela 10.1. *Składniki ASP*

Składnik	Przeznaczenie
`Browser Capabilities`	Dostarcza informacje o używanej przez gościa przeglądarce oraz o jej możliwościach.
`Ad Rotator`	Dostarcza metody pozwalające na łatwe zarządzanie banerami reklamowymi w witrynie.
`Page Counter`	Pozwala na wyświetlanie i śledzenie liczby gości strony.
`Counters`	Dostarcza mechanizmy pozwalające na obsługę liczby, która ma być inkrementowana poprzez aplikację ASP.
`Content Linking`	Pozwala na stworzenie serii stron zorganizowanych na kształt książki i zapewnia możliwość poruszania się po tych stronach.
`Content Rotator`	Pozwala na włączenie do strony lub do jej elementów zawartości losowej.
`MyInfo`	Dostarcza obiekt wraz z licznymi właściwościami, których możesz użyć w celu zapamiętania informacji o Tobie i Twojej firmie.

Składnik Browser Capabilities

Składnik Browser Capabilities dostarcza informacje o przeglądarce gościa oraz o systemie, którym dysponuje. Poniżej zaprezentowano proste wywołanie tego składnika:

```
set BC = server.createobject("MSWC.BrowserType")
Browser = BC.Browser & " " & BC.Version
FramesSupport = BC.Frames
Platform = BC. Platform
```

Instancja składnika `Browser Capabilities` jest tworzona przy użyciu metody `CreateObject`:

```
set BC = server.createobject("MSWC.BrowserType")
```

Wywołanie tworzy instancję klasy `BrowserType` serwera `MSWC` i wpisuje ją do zmiennej obiektowej `BC`. Po wywołaniu, w zmiennej `Browser` ustawiana jest nazwa i wersja przeglądarki używanej przez gościa:

```
Browser = BC.Browser & " " & BC.Version
```

W zmiennej `FramesSupport` ustawiana jest wartość informująca o tym, czy przeglądarka gościa obsługuje ramki:

```
FramesSupport = BC.Frames
```

Dodatkowo w zmiennej `Platform` zapisywana jest nazwa systemu operacyjnego gościa. Wykorzystywana jest właściwość `Platform` obiektu `Browser Capabilities`:

```
Platform = BC.Platform
```

Składnik `Browser Capabilities` działa w bardzo prosty sposób. Wyszukuje zmienną nagłówka HTPP o nazwie `User Agent`, która zawiera nieco skomplikowany wiersz z informacjami o gościu. Wyszukana wartość jest następnie sprawdzana w pliku *.ini* o nazwie *browscap.ini*, który jest plikiem tekstowym, zawierającym informacje o konfiguracji wielu systemów operacyjnych i przeglądarek. Plik *browscap.ini* musi znajdować się w tym samym katalogu, co plik biblioteki składnika *browscap.dll*.

Dla Internet Explorera w systemie Windows NT zawartość nagłówka `User Agent` może wyglądać tak:

```
Mozilla/4.0 (compatible; MSIE 4.01; Windows NT)
```

Składnik przegląda plik *browscap.ini* w poszukiwaniu zapisu:

```
[Mozilla/4.0 (compatible; MSIE 4.01; Windows NT)*]
```

Poniżej znajduje się informacja o konfiguracji systemowej:

```
parent=IE 4.0
minorver=01
platform=WinNT
```

Zwróć uwagę na pierwszy wiersz określający *przodka* (*parent*). Informuje on o obecności w pliku *browscap.ini* pozycji o nazwie IE 4.0. Biblioteka użyje zapisanych tutaj informacji, a w przypadku braku którejś z nich, użyje wartości zapisanych w przodku tego typu.

W katalogu nadrzędnym zapisana zostanie konfiguracja dla tego systemu:

```
browser=IE
Version=4.0
majorver=#4
minorver=#0
frames=TRUE
tables=TRUE
cookies=TRUE
backgroundsounds=TRUE
vbscript=TRUE
javascript=TRUE
javaapplets=TRUE
ActiveXControls=TRUE
Win16=False
beta=False
AK=False
SK=False
AOL=False
crawler=False
CDF=True
```

W przypadku przeglądarki Netscape Navigator w systemie Windows 95 nagłówek zwróci wartość:

```
Mozilla/4.0 * (Win95; U)
```

Składnik `Browser Capabilities` będzie szukał tej wartości w pliku *browscap.ini* i znajdzie zapis:

```
parent=Netscape 4.0
platform=Win95
```

Zauważ, że w pozycji przodka znajduje się teraz nazwa Netscape 4.0. Składnik poszuka tej wartości dla innych właściwości i znajdzie następujące pozycje:

```
browser=Netscape
version=4.0
majorver=#4
minorver=#0
frames=TRUE
tables=TRUE
cookies=TRUE
backgroundsounds=FALSE
vbscript=FALSE
javascript=TRUE
javaapplets=TRUE
ActiveXControls=FALSE
beta=False
```

Przy pisaniu kodu można użyć składnika `Browser Capabilities`, który określa możliwości systemu gościa. Oto przykładowy kod:

```
<%
Option Explicit
Dim BC
set BC = server.createobject("MSWC.BrowserType")
Response.Write "Przeglądarka: " & BC.Browser & " " & BC.Version & "<BR>"
Response.Write "Platforma: " & BC.Platform & "<BR>"
Response.Write "DHTML: " & BC.dhtml & "<BR>"
Response.Write "Ramki: " & BC.frames & "<P>"
```

```
Response.Write "Tabele: " & BC.tables & "<BR>"
Response.Write "Cookies: " & BC.cookies & "<BR>"
Response.Write "Dźwięki tła: " & BC.backgroundsounds & "<BR>"
Response.Write "VBScript: " & BC.vbscript & "<P>"
Response.Write "JavaScript: " & BC.javascript & "<BR>"
Response.Write "Aplety Java: " & BC.javaapplets & "<BR>"
Response.Write "Obiekty sterujące ActiveX: " & BC.activexcontrols & "<BR>"
Response.Write "AOL: " & BC.AOL & "<P>"
Response.Write "Beta: " & BC.Beta & "<BR>"
Response.Write "CDF: " & BC.cdf
%>
```

Na wstępie zaznaczamy, że deklarowane będą zmienne:

```
Option Explicit
```

Następnie tworzymy zmienną o nazwie `BC`:

```
Dim BC
```

W zmiennej `BC` tworzona jest kopia składnika `Browser Capabilities`:

```
set BC = server.createobject("MSWC.BrowserType")
```

Do przeglądarki wysyłana jest nazwa i wersja przeglądarki gościa:

```
Response.Write "Przeglądarka: " & BC.Browser & " " & BC.Version & "<BR>"
```

Dalej do przeglądarki przesyłana jest nazwa systemu operacyjnego gościa:

```
Response.Write "Platforma: " & BC.Platform & "<BR>"
```

Teraz do przeglądarki wysyłąna jest informacja o tym, czy przeglądarka obsługuje dynamiczny HTML. Możliwe dla tej właściwości wartości to: prawda (`True`), fałsz (`False`) oraz nieznany (`Unknown`):

```
Response.Write "DHTML: " & BC.dhtml & "<BR>"
```

Następny fragment odpowiada na pytanie, czy przeglądarka obsługuje ramki. Możliwe wartości dla tej właściwości to: `True`, `False` i `Unknown`:

```
Response.Write "Ramki: " & BC.frames & "<P>"
```

Składnik `Browser Capabilities` informuje o tym, czy przeglądarka wyświetla tabele. Możliwe wartości to: `True`, `False` i `Unknown`:

```
Response.Write "Tabele: " & BC.tables & "<BR>"
```

Czy przeglądarka gościa obsługuje znaczniki cookie? Możliwe wartości dla tej właściwości to: `True`, `False` i `Unknown`. Przeglądarka może obsługiwać cookies, ale gość może dysponować narzędziem blokującym te znaczniki:

```
Response.Write "Cookies: " & BC.cookies & "<BR>"
```

Czy przeglądarka potrafi odtwarzać dźwięki tła? Możliwe wartości dla tej właściwości to: `True`, `False` i `Unknown`:

```
Response.Write "Dźwięki tła: " & BC.backgroundsounds & "<BR>"
```

Czy przeglądarka potrafi przetwarzać VBScript po stronie klienta? Zwróć uwagę na to, że mówimy tutaj o stronie klienta, co nie ma nic wspólnego ze stronami ASP, które są przetwarzane przez Twój serwer. Możliwe wartości dla tej właściwości to: `True`, `False` i `Unknown`:

```
Response.Write "VBScript: " & BC.vbscript & "<P>"
```

To samo pytanie dotyczy JavaScriptu. Ponownie odnosimy się do możliwości po stronie klienta. Możliwe wartości dla tej właściwości to `True`, `False` i `Unknown`:

```
Response.Write "JavaScript: " & BC.javascript & "<BR>"
```

Czy przeglądarka obsługuje aplety Javy? Odpowiedź na to pytanie zawarta jest we właściwości `JavaApplets` składnika `Browser Capabilities`:

```
Response.Write "Aplety Javy: " & BC.javaapplets & "<BR>"
```

Czy przeglądarka może współpracować ze składnikami ActiveX po stronie klienta? Możliwe wartości dla tej właściwości to `True`, `False` i `Unknown`:

```
Response.Write "Obiekty sterujące ActiveX: " & BC.activexcontrols & "<BR>"
```

Czy gość używa AOL? Możliwe wartości dla tej właściwości to `True`, `False` i `Unknown`:

```
Response.Write "AOL: " & BC.AOL & "<P>"
```

Czy przeglądarka jest w wersji beta? Możliwe wartości dla tej właściwości to `True`, `False` i `Unknown`:

```
Response.Write "Beta: " & BC.Beta & "<BR>"
```

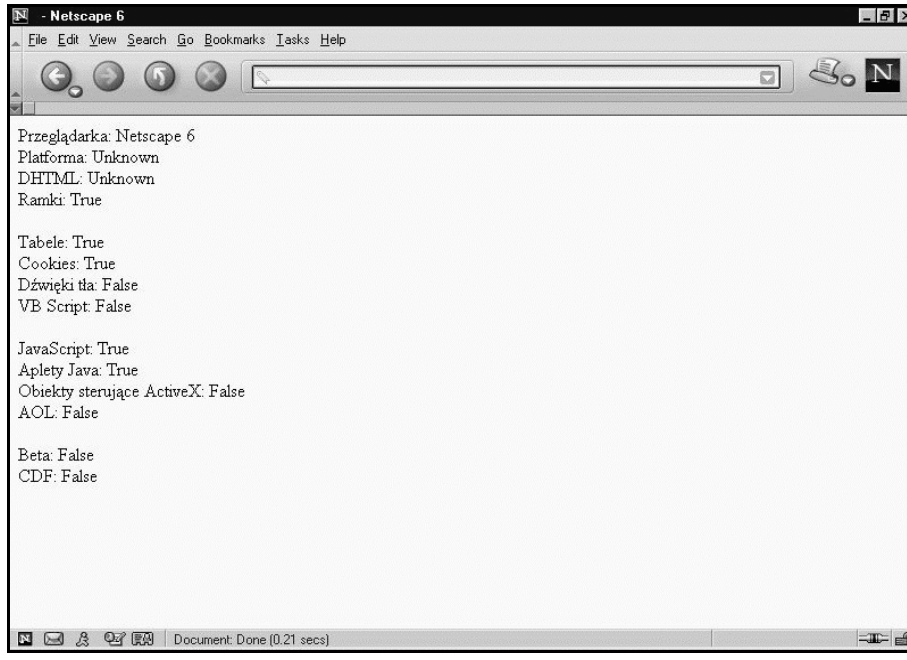
I wreszcie, czy przeglądarka obsługuje format CDF? Możliwe wartości dla tej właściwości to `True`, `False` i `Unknown`:

```
Response.Write "CDF: " & BC.cdf
```

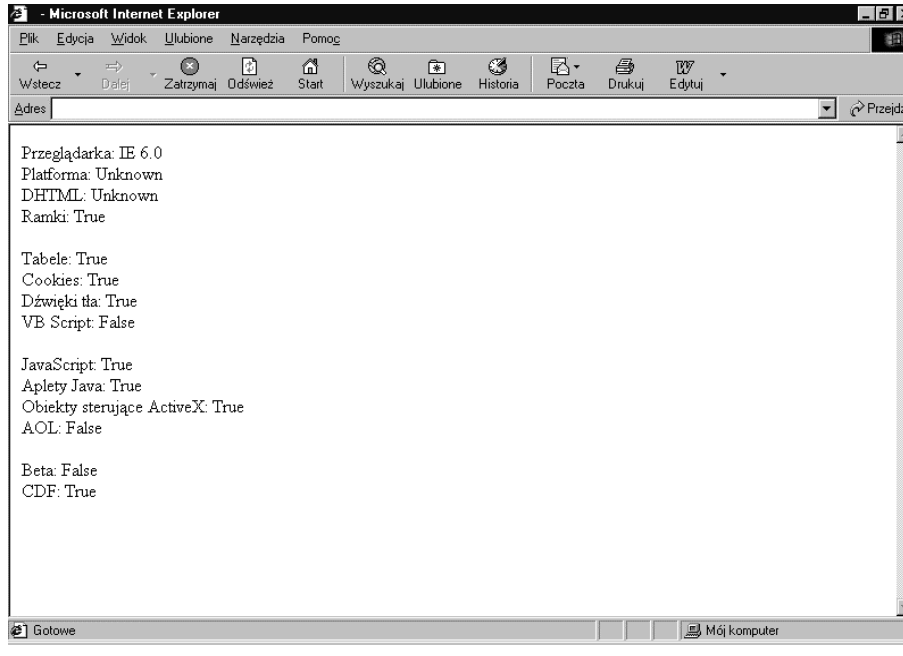
Rysunek 10.1. przedstawia wygląd tej strony w systemie Windows 98 używającym przeglądarki Netscape Navigator 4.0.

Rysunek 10.2 prezentuje rezultat działania kodu w systemie Windows NT z uruchomionym Internet Explorerem 5.0. Najważniejszą możliwością wykorzystania tego kodu jest kierowanie gości do właściwej wersji Twojej witryny WWW. Może ona zawierać witrynę optymalizowaną dla Internet Explorera, drugą dla Netscape Navigatora, trzecią dla WebTV i kolejną dla pozostałych typów przeglądarek. Możesz to zrobić, wykorzystując kod, taki jak ten:

```
<%
Option Explicit
Dim BC
set BC = server.createobject("MSWC.BrowserType")
If BC.Browser = "IE" then
  Response.redirect "./IE/index.asp"
ElseIf BC.Browser = "Netscape" then
  Response.redirect "./Netscape/index.asp"
ElseIf BC.Browser = "WebTV" then
  Response.redirect "./WebTV/index.asp"
Else
  Response.redirect "./other/index.asp"
End If
%>
```



Rysunek 10.1. *Wyjście kodu wykorzystującego składnik Browser Capabilities*



Rysunek 10.2. *Wyjście kodu wykorzystującego składnik Browser Capabilities*

Na wstępie wymagana jest deklaracja zmiennych:

```
Option Explicit
```

Tworzona jest zmienna o nazwie `BC`:

```
Dim BC
```

Zmienna wykorzystana jest do stworzenia instancji obiektu `Browser Capabilities`:

```
set BC = server.createobject("MSWC.BrowserType")
```

Następnie pytamy właściwość `Browser` obiektu `Browser Capabilities`, czy używana przeglądarka to Internet Explorer:

```
If BC.Browser = "IE" then
```

Jeśli tak, to gość zostaje przekierowany do podkatalogu zawierającego strony dla tej przeglądarki:

```
Response.redirect "./IE/index.asp"
```

W innym wypadku kod sprawdza, czy używana przeglądarka to Netscape Navigator:

```
ElseIf BC.Browser = "Netscape" then
```

Jeśli to prawda, gość odsyłany jest do katalogu *Netscape*:

```
Response.redirect "./Netscape/index.asp"
```

Jeśli szerokie grono Twych gości korzysta z WebTV, musisz dysponować witryną w wersji odpowiadającej ich potrzebom:

```
ElseIf BC.Browser = "WebTV" then
```

Jeśli jest to ich typ przeglądarki, są przekierowani do podkatalogu `WebTV`:

```
Response.redirect "./WebTV/index.asp"
```

w przeciwnym razie gości odsyła się do domyślnego widoku Twojej witryny:

```
Response.redirect "./other/index.asp"
```

Kolejnym powodem do przekierowywania Twoich gości do określonej wersji witryny są różnice w technologiach. Jeśli martwisz się o to, czy starsze przeglądarki będą mogły korzystać z Twojej witryny, to metodą na zróżnicowanie wyglądu witryny jest sprawdzenie obsługi tabel i ramek. Zwróć uwagę na ten kod:

```
<%
Option Explicit
Dim BC
set BC = server.createobject("MSWC.BrowserType")
If BC.Tables = "True" and BC.Frames = "True" then
  Response.Redirect "./full/index.asp"
ElseIf BC.Tables = "True" then
  Response.Redirect "./noframes/index.asp"
Else
  Response.Redirect "./none/index.asp"
End If
%>
```

Jak zwykle wymagana jest deklaracja zmiennych:

```
Option Explicit
```

Tworzona jest zmienna `BC`:

```
Dim BC
```

i ustawiany jest w niej obiekt `Browser Capabilities`:

```
set BC = server.createobject("MSWC.BrowserType")
```

Zaczynamy od pytania podstawowego. Czy przeglądarka obsługuje tabele i ramki?

```
If BC.Tables = "True" and BC.Frames = "True" then
```

Jeśli przeglądarka obsługuje zarówno jedno, jak i drugie, gość odsyłany jest tutaj:

```
Response.Redirect "./full/index.asp"
```

Następnie cofamy się o jeden poziom w technologii. Ramki pojawiły się później niż tabele, usuwamy więc z równania pytanie o ramki i sprawdzamy, czy przeglądarka obsługuje przynajmniej tabele:

```
ElseIf BC.Tables = "True" then
```

Jeśli tak jest, wykorzystujemy metodę `Redirect` obiektu `Response` w celu odesłania gości do odpowiedniego miejsca:

```
Response.Redirect "./noframes/index.asp"
```

W innym wypadku przeglądarka gościa nie obsługuje żadnej z tych technologii i wtedy odsyłamy go do podstawowej wersji naszej witryny:

```
Else
   Response.Redirect "./none/index.asp"
```

Pamiętaj, że składnik szuka pozycji `User Agent` w pliku *.ini* i dopasowuje ją do wartości zmiennej `User Agent` zwróconej przez nagłówek HTPP. Plik *.ini* jest jedynie plikiem tekstowym, zawierającym zbiór różnego rodzaju wydruków, co oznacza, że jeśli nie będziesz się nim zajmował — plik szybko stanie się przestarzały. Faktycznie, jeśli instalujesz nowy IIS, prawdopodobnie będzie on dysponował przestarzałą wersją pliku *.ini*. Jeśli więc korzystasz z tego narzędzia, musisz dokładać starań, aby plik był stale aktualny.

Jednym z miejsc, w których możesz otrzymać uaktualnienie, jest cyScape, Inc. (*http://www.cyscape.com/browscap*). Firma często udostępnia nową wersję pliku *browscap.ini* wraz z dodatkami i poprawkami. Możesz zgłosić chęć otrzymywania uaktualnienia przez e-mail, wtedy firma będzie powiadamiała Cię o pojawieniu się nowych wersji pliku. cyScape dysponuje również własną wersją składnika `Browser Capabilities`, która zawiera dodatkowe właściwości, inne niż te omówione w bieżącym podrozdziale.

Modyfikacja pliku browser.ini

Możesz również sam uaktualnić plik. Pamiętaj, że jest to tylko plik tekstowy, który musi mieć określony format. Jeśli chcesz go zmodyfikować, najpierw stwórz jego kopię zapasową. Wtedy możesz wykorzystać poniższy opis w celu modyfikacji poszczególnych sekcji pliku.

Komentarzem w pliku jest każdy wiersz rozpoczynający się średnikiem:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Definicja mojej przeglądarki ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

Każda część pliku zawiera albo nagłówek `User Agent`, albo definicję przodka. Mógłbyś na przykład zdefiniować przeglądarkę, której nie było na liście, a dla której zwracany byłby następujący ciąg w `User Agent`:

```
[My Browser/1.2.2beta (Windows 95)]
```

Zwróć uwagę na to, że definicja sekcji jest umieszczona w nawiasie kwadratowym. W następnej kolejności pojawiają się właściwości sekcji, aż do kolejnego nagłówka sekcji. Możesz zdefiniować właściwości Twojej sekcji w następujący sposób:

```
parent=MyBrowser 1.0
platform=Win95
beta=True
```

Zwróć uwagę na szczególną pozycję o nazwie `parent` (przodek), która informuje o tym, że gdzieś w pliku *.ini* znajduje się pozycja `MyBrowser 1.0`. Przejdź do tej pozycji i użyj wszystkich właściwości, o ile nie określiłem jakiejś w tej sekcji. W następnej kolejności określona została platforma oraz typ przeglądarki, beta.

Gdzieś w pliku *.ini* musimy dodać sekcję określającą przodka. Pasuje ona dokładnie do nazwy użytej tutaj, a wygląda następująco:

```
[MyBrowser 1.0]
browser=MyBrowser
version=1.0
majorver=#1
minorver=#0
frames=FALSE
tables=TRUE
cookies=TRUE
backgroundsounds=FALSE
vbscript=TRUE
javascript=FALSE
javaapplets= FALSE
Platform=Windows95
beta=False
```

Zauważ, że niektóre pozycje są powtórzone tutaj, w przodku. Te, których użyto w rzeczywistej definicji, unieważniają wartości przodka, który po prostu przechowuje domyślne wartości nie zdefiniowane ponownie przez pozycję potomka.

Składnik Ad Rotator

Składnik Ad Rotator udostępnia narzędzia do zarządzania paskami reklamowymi (banerami), pojawiającymi się na stronie WWW. Składnik będzie losowo wyświetlał baner na stronie ASP za każdym razem, kiedy ktoś uzyskuje do niej dostęp. Składnik używa oddzielnego pliku harmonogramu,

który stworzony jest w celu określenia częstości wyświetlania jednych banerów względem drugich. Wywołanie składnika `Ad Rotator` wygląda następująco:

```
set objAdRot = server.createobject("MSWC.AdRotator")
Response.Write objAdRot.GetAdvertisement("./html/AdFiles/AdFile.txt)
```

Pierwszy wiersz tworzy kopię instancję `AdRotator`, będącej klasą `MSWC`, umieszczając ją w zmiennej obiektowej o nazwie `objAdRot`:

```
set objAdRot = server.createobject("MSWC.AdRotator")
```

Następnie wywoływana jest metoda `GetAdvertisement` w celu wyszukania informacji o pasku reklamowym, który będzie wyświetlany. Zwracana wartość wysłana zostaje do przeglądarki przy użyciu metody `Write` obiektu `Response`:

```
Response.Write objAdRot.GetAdvertisement("./html/AdFiles/AdFile.txt)
```

A oto ta wpisywana do przeglądarki wartość:

```
<A HREF="http://www.netstats2000.com/nmaha/html/AdFiles/AdRedirect.
asp?
url= http://www.netstats2000.com/nmaha/html/AdFiles/HA.html
&image= http://www.netstats2000.com/nmaha/html/AdFiles/ha.gif" >
<IMG SRC=" http://www.netstats2000.com/nmaha/html/AdFiles/ha.gif"
ALT="Get your organization online" WIDTH=468 HEIGHT=60 BORDER=0>
```

Metodzie `GetAdvertisement` przekazywany jest jeden parametr, ścieżka do pliku *Schedule*, zawierającego listę wyświetlanych banerów oraz informacje o częstości wyświetlania każdego z nich. Plik *Schedule* zawiera również działania, jakie należy podjąć w chwili, gdy gość kliknie baner, a także rozmiar prezentowanego obrazka. Metoda `GetAdvertisement` sprawdza wszystkie wartości pliku *Schedule* i tworzy pokazany powyżej kod HTML.

Zawartość pliku *Schedule* w tym autentycznym przykładzie zaprezentowano tutaj:

```
Redirect http://www.netstats2000.com/nmaha/html/AdFiles/AdRedirect.asp
width 468
height 60
border 0
*
http://www.netstats2000.com/nmaha/html/AdFiles/ha.gif
http://www.netstats2000.com/nmaha/html/AdFiles/HA.html
Get your organization online
10
http://www.netstats2000.com/nmaha/html/AdFiles/sgianim.gif
http://www.silkgraph.com
Silkscreen Graphics Inc.
10
http://www.netstats2000.com/nmaha/html/AdFiles/forrent.gif
mailto:mcecchi@ibm.net
Space For Rent
10
```

Pierwsza sekcja pliku *Schedule* zawiera informacje konfiguracyjne. Pierwszy wiersz określa ścieżkę do strony, do której goście są odsyłani po kliknięciu paska reklamowego:

```
Redirect http://www.netstats2000.com/nmaha/html/AdFiles/AdRedirect.asp
```

Za pomocą zbioru `QueryString` stronie przekazywana jest nazwa kliknięty obrazka oraz adres, do którego należy po tej akcji przejść.

Drugi i trzeci wiersz pierwszej sekcji określają szerokość i wysokość paska reklamowego:

```
width 468
height 60
```

Czwarty wiersz określa jego obwódkę:

```
border 0
```

Spójrz na poprzedni HTML, stworzony przez metodę `GetAdvertisement`, a zobaczysz w nim informacje konfiguracyjne. Pojedyncza gwiazdka kończy sekcję konfiguracyjną:

```
*
```

Dalej w pliku znajduje się lista banerów z ich harmonogramem. Pierwszy wiersz każdej pozycji banera jest ścieżką dostępu do niego:

```
http://www.netstats2000.com/nmaha/html/AdFiles/ha.gif
```

Potem pojawia się lokalizacja WWW, do której ostatecznie goście będą przekierowani po kliknięciu banera:

```
http://www.netstats2000.com/nmaha/html/AdFiles/HA.html
```

W trzecim wierszu znajduje się tekst etykietki:

```
Get your organization online
```

Czwarty wiersz zawiera liczbę, określającą częstość wyświetlania jednego banera proporcjonalnie w stosunku do innych:

```
10
```

Rezultat działania s tego wpisu w pliku konfiguracyjnym pokazany jest na rysunku 10.3. Zwróć uwagę na baner znajdujący się na szczycie tego rysunku. Jest on rezultatem pierwszej pozycji w pliku, tak więc tekst etykiety (atrybut ALT) zgadza się z tym, co pokazano na rysunku, a na pasku statusu widać adres URL, z którym łączy się baner.

Jeśli spojrzysz wstecz na całość pliku *Schedule*, zauważysz, że opisana wyżej struktura się powtarza. Następny punkt wyświetli tę grafikę:

```
http://www.netstats2000.com/nmaha/html/AdFiles/sgianim.gif
```

Jeśli gość kliknie tę grafikę, zostaje odesłany do następującego adresu URL:

```
http://www.silkgraph.com
```

Tekst etykietki dla grafiki:

```
Silkscreen Graphics Inc.
```

Relatywna liczba wyświetleń tej grafiki w porównaniu z innymi jest taka:

```
10
```



Rysunek 10.3. *Baner po pierwszym wyświetleniu.*

Ta pozycja wyświetli widok strony pokazany na rysunku 10.4. Zauważ, że grafika odpowiada treści pliku konfiguracyjnego komponentu. Poza tym zwróć uwagę, że jest to ta sama strona, na której prezentowany był poprzedni baner. Za każdym razem, gdy strona jest przeglądana, pojawia się na niej inna grafika, mianowicie ta, która dawno nie była wyświetlana.

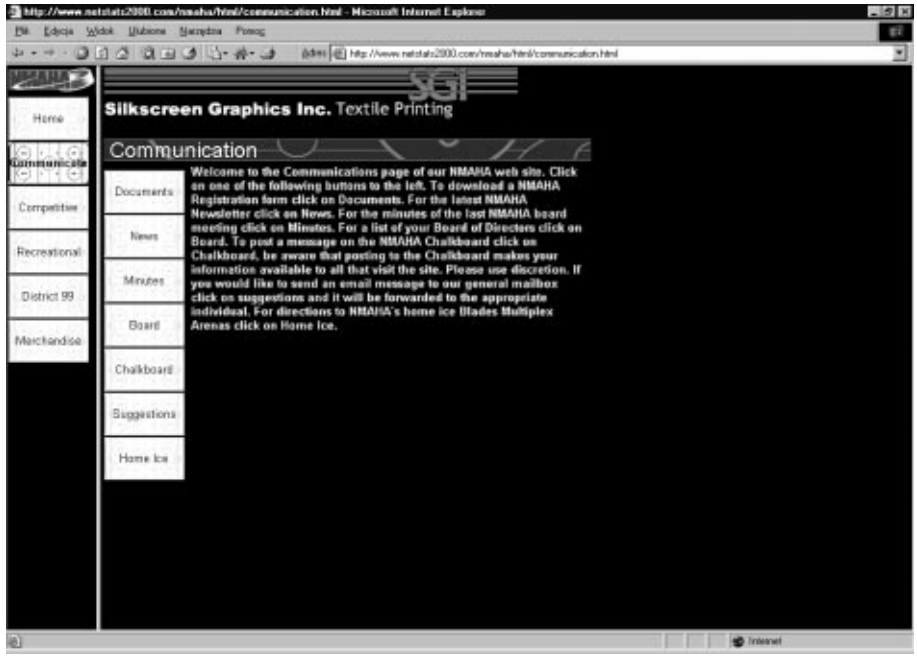
Ponadto każda strona miała zaplanowaną tą samą proporcję wyświetleń, 10, ale tak być nie musi. Możesz dla jednej strony ustawić wartość 20, a dla pozostałych dwóch 10. Oznaczałoby to, że pierwsza strona ukazywałaby się dwa razy częściej niż dwie pozostałe.

Pamiętaj, że pierwszy wiersz pliku *Schedule* określa miejsce, do którego odsyłani są goście, gdy klikną baner:

```
Redirect http://www.netstats2000.com/nmaha/html/AdFiles/AdRedirect.asp
width 468
```

Ta strona ASP normalnie nie wysyla żadnego kodu HTML do przeglądarki. Daje Ci ona możliwość zapisywania kliknięć banera (*click-through*) w bazie danych w celu dalszej analizy. Poniżej przedstawiono przykład takiego kodu.

```
<%
set conn = server.createobject ("adodb.connection")
conn.open "EmpDir", "sa", "twoje_haslo"
conn.execute "insert into Referrals (ReferredTo) values ('" _
   & Request.QueryString("url") & "')"
response.redirect(Request.QueryString("url"))
%>
```



Rysunek 10.4. *Drugie ogłoszenie z pliku obrotu*

Na początku kod łączy się z bazą danych:

```
set conn = server.createobject ("adodb.connection")
conn.open "EmpDir", "sa", "twoje_haslo"
```

Następnie kod dodaje rekord do tabeli o nazwie `Referrals`, zawierający adres docelowy na banerze kliknięty przez gościa:

```
conn.execute "insert into Referrals (ReferredTo) values ('" _
  & Request.QueryString("url") & "')"
```

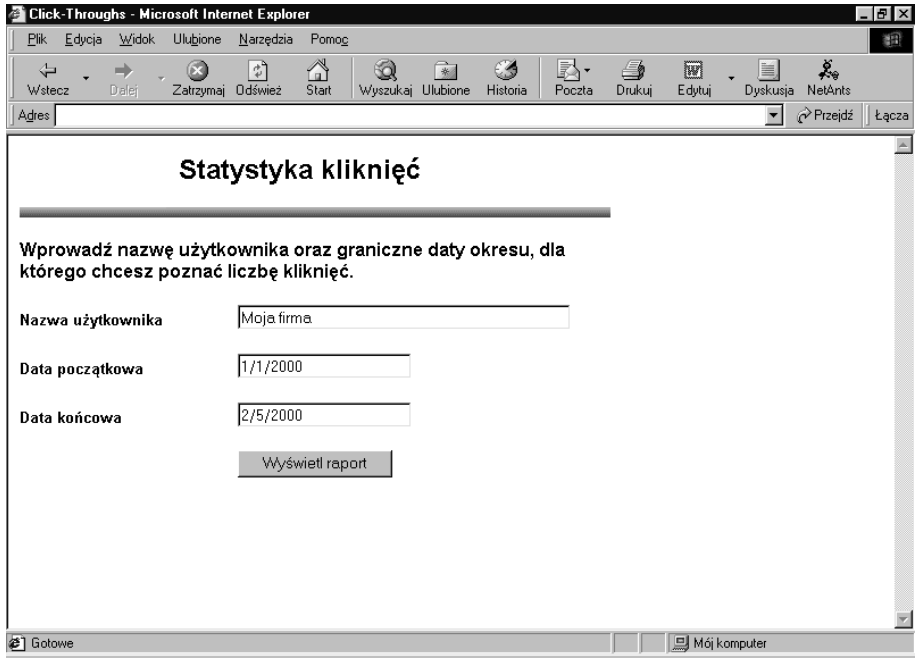
W końcu goście są przekierowani do miejsca docelowego wskazanego przez kliknięty baner w witrynie WWW:

```
response.redirect(Request.QueryString("url"))
```

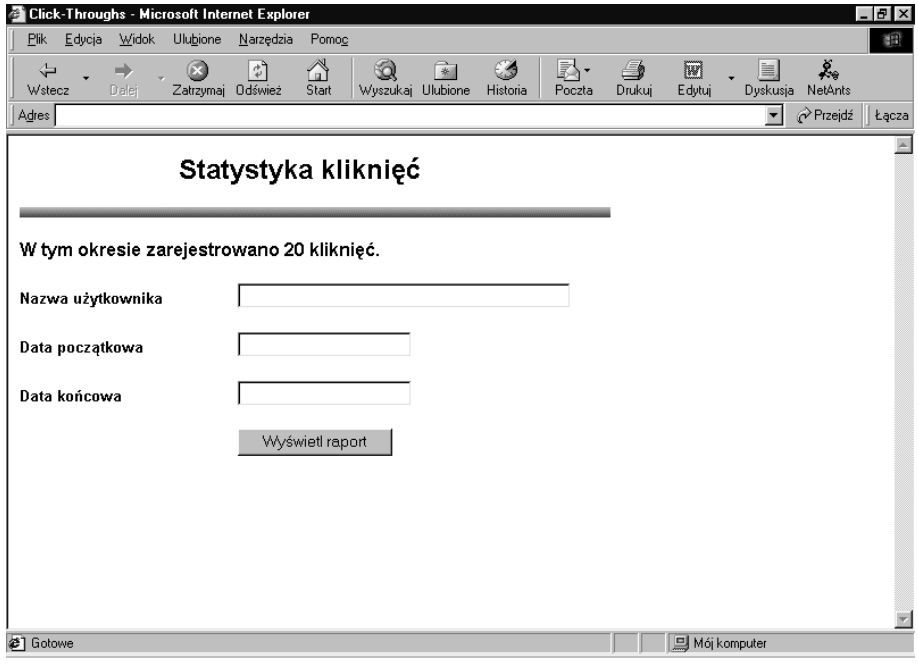
Pomyśl o innych polach, które chciałbyś zawrzeć w tej pozycji. Na przykład prawdopodobnie będziesz chciał zapisywać datę i godzinę kliknięcia banera przez gościa. Możesz również wykorzystać składnik `Browser Capabilities`, omówiony we wcześniejszym podrozdziale, w celu zapisywania obszernych informacji na temat narzędzi gościa, który kliknął baner.

Z drugiej strony może chciałbyś dysponować stroną, na której reklamujący się mogliby sprawdzić, ile razy ich baner był kliknięty. Taką stronę prezentuje rysunek 10.5.

Domyślnie strona ładuje się, wyświetlając datę w skali rocznej. Kiedy goście klikną przycisk *Wyświetl raport*, zobaczą wynik ich zapytania pokazany na rysunku 10.6.



Rysunek 10.5. *Przykładowa strona raportu kliknięć*



Rysunek 10.6. *Wynik zapytania o liczbę kliknięć*

Kod strony wyśle zapytanie do bazy danych na podstawie informacji przesłanych w formularzu. Kod strony również musi wyświetlać informacje w skali rocznej. Główny blok kodowy jest następujący:

```
<%
Option Explicit
Dim conn
Dim RSHits
Dim TheMessage
Dim StartDate
Dim EndDate
If Not IsEmpty(Request.Form("DisplayReport")) Then
  set conn = server.createobject ("adodb.connection")
  conn.open "EmpDir", "sa", "twoje_haslo"
  set RSHits = conn.Execute("select Count(ReferralID) as TheCount " _
    & "from Referrals " _
    & "where UserName = '" & Request.Form("UserName") & "' " _
    & "and HitDate >= '" & Request.Form("StartDate") & "' " _
    & "and HitDate <= '" & Request.Form("EndDate") & "' ")
  TheMessage = " W tym okresie zarejestrowano" _
    & RSHits("TheCount") & " kliknięć."
else
  TheMessage ="Wprowadź nazwę użytkownika oraz graniczne daty, dla" _
    & "dla którego chcesz poznać liczbę kliknięć."
End If
StartDate = "1/1/" & Year(Date)
EndDate = Date
%>
```

Na wstępie wykorzystujemy dyrektywę `Option Explicit`:

```
Option Explicit
```

Następne deklarujemy zmienną zapamiętującą połączenie z bazą danych:

```
Dim conn
```

Dalej deklarujemy zmienną przechowującą informacje z bazy danych:

```
Dim RSHits
```

Zmienna wiadomości będzie wyświetlała w przeglądarce instrukcje lub wynik zapytania:

```
Dim TheMessage
```

Kolejne dwie zmienne będą przechowywały domyślną datę początkową:

```
Dim StartDate
```

i datę końcową:

```
Dim EndDate
```

Strona ma dwa stany. W *stanie początkowym* strona właśnie została wyświetlona i goście muszą ujrzeć instrukcje postępowania. W *stanie przesyłania* goście wysyłają formularz, a my musimy go przetworzyć. Stan strony określamy, sprawdzając w zbiorze `Form`, czy przycisk wyświetlający raport (`DisplayReport`) został naciśnięty.

```
If Not IsEmpty(Request.Form("DisplayReport")) Then
```

Jeśli element zbioru formularza nie jest pusty, formularz został przesłany. W tym wypadku musimy połączyć się z bazą danych, aby ustalić liczbę kliknięć. Tworzony jest obiekt `Connection`:

```
set conn = server.createobject ("adodb.connection")
```

który łączy się z żądaną bazą danych:

```
conn.open "EmpDir", "sa", "twoje_haslo"
```

Uruchamiane jest zapytanie, które zwraca liczbę rekordów dla użytkownika pomiędzy wybranymi przez niego datami:

```
set RSHits = conn.Execute("select Count(ReferralID) as TheCount " _
   & "from Referrals " _
   & "where UserName = '" & Request.Form("UserName") & "' " _
   & "and HitDate >= '" & Request.Form("StartDate") & "' " _
& "and HitDate <= '" & Request.Form("EndDate") & "' ")
```

Następnie do zmiennej wpisywany jest wyświetlany gościom tekst, który zawiera wyszukane w bazie danych informacje o kliknięciach:

```
TheMessage = " W tym okresie zarejestrowano" _
   & RSHits("TheCount") & " kliknięć."
```

Jeśli formularz nie został przesłany, w zmiennej wiadomości ustawiana jest instrukcja postępowania:

```
else
  TheMessage ="Wprowadź nazwę użytkownika oraz graniczne daty, dla" _
    & "dla którego chcesz poznać liczbę kliknięć."
```

W każdym z przypadków musimy utworzyć zmienne `StartDate` i `EndDate`. Jako datę początkową ustawiamy pierwszy dzień bieżącego roku:

```
StartDate = "1/1/" & Year(Date)
```

a jako datę końcową bieżącą datę systemu:

```
EndDate = Date
```

Następnie używamy wbudowanej w HTML metody `Write` obiektu `Response` w celu wysłania zmiennej wiadomości do przeglądarki:

```
<P><B><% Response.Write TheMessage %></B>
```

Wartość daty początkowej wpisywana jest w pole tekstowe *StartDate*:

```
<INPUT TYPE=TEXT NAME="StartDate" VALUE="<% Response.Write
StartDate %>"
SIZE=40 MAXLENGHT=50>
```

Następnie ustalamy domyślną datę końcową w parametrze wartości:

```
<INPUT TYPE=TEXT NAME="EndDate" VALUE="<% Response.Write EndDate %>"
SIZE=40 MAXLENGHT=50>
```

Składnik Page Counter

Składnik Page Counter pozwala w prosty sposób śledzić i wyświetlać liczbę trafnych wywołań strony WWW. Ten składnik nie jest instalowany domyślnie wraz z serwerem IIS. Jest on częścią zestawu IIS Resource Kit, który możesz znaleźć w witrynie Microsoftu lub, jeśli subskrybujesz Microsoft Technet, znajduje się on na CD-ROM-ie. Składnik możesz zainstalować, kopiując plik z płyty CD-ROM położony w *\IIS Resource Kit\Component\Page Counter\DLL\i386\PageCnt.dll*, a następnie umieszczając go w Twoim katalogu *WinNT\System32*. Wtedy możesz zarejestrować składnik przy użyciu polecenia *Uruchom* (*Run*) z menu *Start*, wprowadzając tekst `Regsvr32 PageCnt.dll`.

Powinieneś ujrzeć wiadomość o pomyślnym zarejestrowaniu składnika, którą zaprezentowano na rysunku 10.7. Po zarejestrowaniu biblioteki możesz zacząć korzystać ze składnika w kodzie. Składnik tworzy się podobnie jak inne składniki, to znaczy przy użyciu metody `CreateObject` obiektu `Server`.



Rysunek 10.7. *Pomyślna rejestracja składnika Page Counter*

Dokumentacja Microsoftu prezentuje dwa sposoby tworzenia obiektu. Odkryłem, że jeden z nich nie działa, ale przyjrzyjmy się obydwu sposobom na wypadek, gdyby przyszłe wersje używały tej niefunkcjonalnej składni. Poniższa deklaracja aktualnie nie działa:

```
Set MyPageCounter = Server.CreateObject("MSWC.PageCounter")
```

Zmienna `MyPageCounter` zawiera instancję składnika `Page Counter`. Druga składnia do tworzenia kopii tego składnika używa innego ciągu aplikacji i klasy:

```
Set MyPageCounter = Server.CreateObject("IISSample.PageCounter")
```

Tu również zmienna `MyPageCounter` jest obiektem klasy `Page Counter`.

Ta biblioteka *.dll* tworzy plik tekstowy i zapamiętuje, ile razy strona była przeglądana. Składnik okresowo umieszcza te informacje wpliku tekstowym, tak więc dane zostaną zachowane, nawet gdy serwer zostanie wyłączony. Kiedy składnik jest inicjowany, ładuje wartości trafnych wywołań różnych stron.

Aby zapisać wywołanie na stronie, musisz użyć metody `PageHit` składnika `Page Counter`:

```
<%
Option Explicit
Dim MyPageCounter
Set MyPageCounter = Server.CreateObject("IISSample.PageCounter")
MyPageCounter.PageHit
%>
```

Najpierw instrukcja `Option Explicit`:

```
Option Explicit
```

Następnie tworzona jest zmienna przechowująca obiekt `Page Counter`:

```
Dim MyPageCounter
```

W zmienną wpisywany jest składnik `Page Counter`:

```
Set MyPageCounter = Server.CreateObject("IISSample.PageCounter")
```

Metoda `PageHit` obiektu `Page Counter` używana jest do inkrementacji liczby dla określonej strony:

```
MyPageCounter.PageHit
```

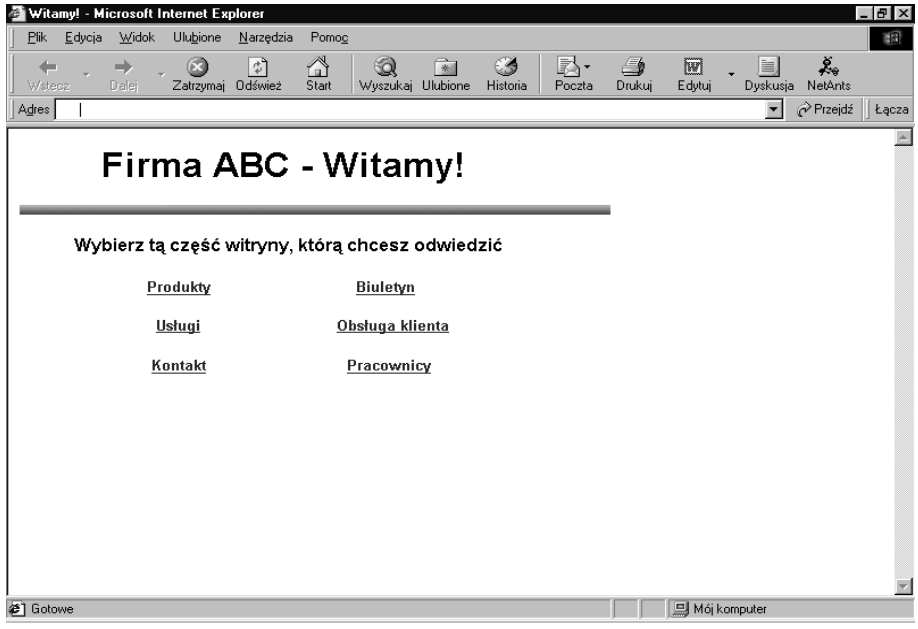
tak więc stan licznika dla tej strony wewnętrznie zwiększany jest o jeden. Składnik zapamiętuje nazwę strony i jej położenie wraz z wartością licznika, co zapewniaunikalność wpisu.

Licznik stron nie byłby wiele wart, jeśli nie mógłbyś wyświetlać jego zawartości. W tym celu składnik zawiera metodę `Hits`. Metoda ta ma następującą składnię:

```
TheCount = MyPageCounter.Hits(OptPageName)
```

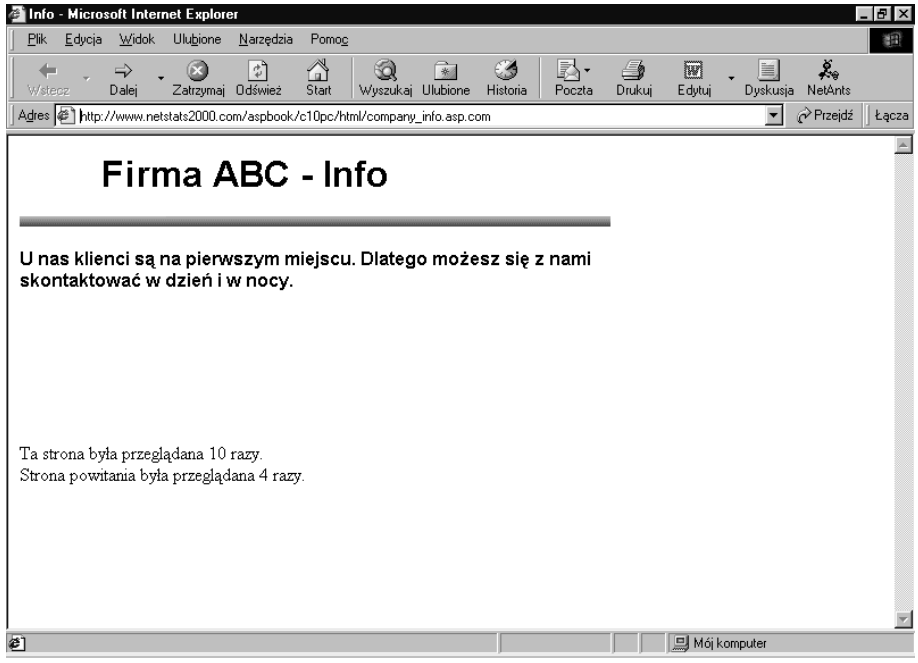
Zmienna `MyPageCounter` musi być poprawnym obiektem `Page Counter`. Zwracaną wartością polecenia jest liczba trafnych wywołań strony, która znajdzie się w zmiennej `TheCount`. Metoda pobiera jeden nieobowiązkowy parametr — nazwę i wirtualną ścieżkę do strony, której liczbę wywołań chcemy poznać. Jeśli nie podasz tego parametru, metoda zwróci liczbę wywołań bieżącej strony.

Spójrzmy na przykładową witrynę, która korzysta z tego składnika. Na początku strona powitania, którą pokazano na rysunku 10.8.



Rysunek 10.8. *Strona powitania przykładowej witryny*

Strona powitania zapamiętuje liczbę swoich wywołań, ale informacji tej nie wyświetla. Następna strona przykładowej witryny również zapamiętuje liczbę swoich wywołań, a ponadto wyświetla informacje o wywołaniach obydwu stron witryny. Ta strona pokazana jest na rysunku 10.9. Zauważ, że wyświetlane informacje faktycznie dotyczą obu stron.



Rysunek 10.9. *Strona wyświetlająca informacje o liczbie wywołań*

Kod strony powitania inkrementuje licznik wywołań dla tej strony. Blok kodowy jest taki:

```
<%
Option Explicit
Dim MyPageCounter
Set MyPageCounter = Server.CreateObject("IISSample.PageCounter")
MyPageCounter.PageHit
%>
```

Po pierwsze, informujemy kompilator o tym, że deklarowane będą zmienne:

```
Option Explicit
```

Następnie deklarujemy zmienną licznika stron:

```
Dim MyPageCounter
```

i tworzymy jego instancję:

```
Set MyPageCounter = Server.CreateObject("IISSample.PageCounter")
```

Licznik strony powitania jest zwiększany o jeden (inkrementowany):

```
MyPageCounter.PageHit
```

Kod strony informacyjnej inkrementuje licznik tej strony, a następnie wyświetla informacje o trafnych wywołaniach obu stron. Główny blok kodowy wygląda następująco:

```
<%
Option Explicit
Dim MyPageCounter
Dim CompInfoCount
Dim WelcomeCount
Set MyPageCounter = Server.CreateObject("IISSample.PageCounter")
MyPageCounter.PageHit
If MyPageCounter.Hits = 1 Then
  CompInfoCount = "1 raz"
else
  CompInfoCount = MyPageCounter.Hits & " razy"
End If
If MyPageCounter.Hits("/aspbook/c10/pc/html/welcome.asp") = 1 Then
  WelcomeCount = "1 raz"
else
  WelcomeCount = MyPageCounter.Hits("/aspbook/c10/pc/html/welcome.asp") & " razy"
End If
%>
```

Pierwszy wiersz kodu to instrukcja `Option Explicit`:

```
Option Explicit
```

Następnie deklarowana jest zmienna obiektowa licznika stron:

```
Dim MyPageCounter
```

Tworzona jest zmienna, która będzie przechowywała liczbę wywołań strony informacyjnej:

```
Dim CompInfoCount
```

Ta zmienna będzie zapamiętywała liczbę wywołań strony powitania:

```
Dim WelcomeCount
```

Następnie tworzony jest obiekt licznika stron:

```
Set MyPageCounter = Server.CreateObject("IISSample.PageCounter")
```

Metoda `PageHit` używana jest do inkrementacji liczby wywołań tej strony:

```
MyPageCounter.PageHit
```

Następnie musimy stworzyć wiadomości tekstowe, informujące o liczbie wywołań obu stron. Metoda `Hits` obiektu `Page Counter` wyszukuje tę wartość. Najpierw jednak patrzymy na liczbę wywołań tej strony:

```
If MyPageCounter.Hits = 1 Then
```

Kod tego bloku `If` nadaje odpowiednią formę gramatyczną słowu „raz”:

```
CompInfoCount = "1 raz"
```

Zauważ, że używamy metody `Hits` bez parametru, ponieważ na razie zajmujemy się liczbą wywołań bieżącej strony:

```
else
  CompInfoCount = MyPageCounter.Hits & " razy"
End If
```

Dalej te same czynności są realizowane dla strony powitania. Zauważ, że teraz metoda `Hits` zawiera parametr, wirtualną ścieżkę do strony, której liczbę wywołań chcemy poznać:

```
If MyPageCounter.Hits("/aspbook/c10/pc/html/welcome.asp") = 1 Then
  WelcomeCount = "1 raz"
else
  WelcomeCount = MyPageCounter.Hits("/aspbook/c10/pc/html/welcome.asp") & " razy"
End If
```

Następnie w HTML-u wypisywana jest informacja o wywołaniach wspólnie z dwoma ciągami:

```
<P>Ta strona była przeglądana <% response.write CompInfoCount %>.
<BR>Strona powitania była przeglądana <% response.write WelcomeCount %>.
```

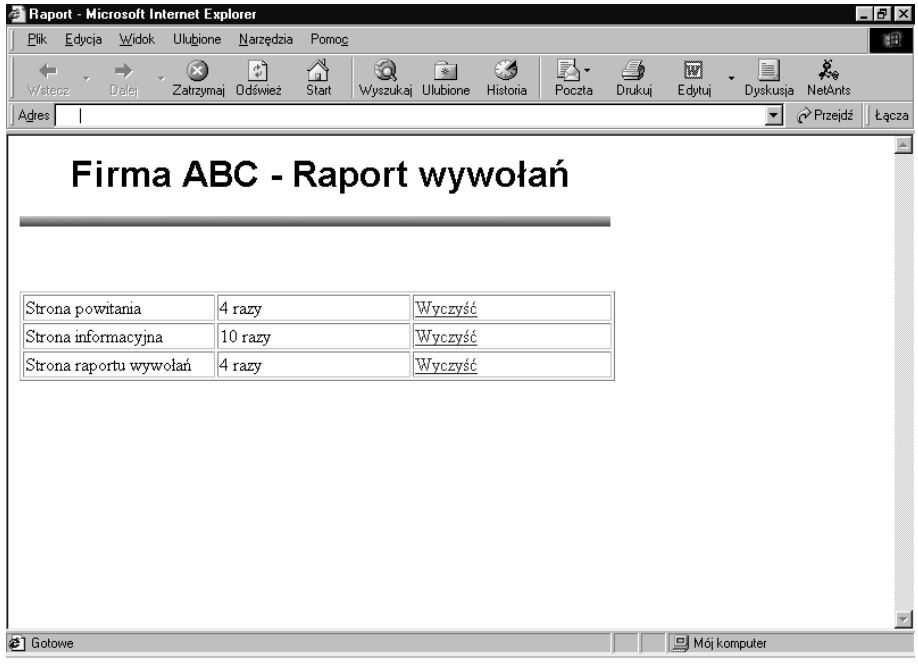
Składnik `Page Counter` ma jeszcze jedną metodę, `Reset`, która wpisuje do licznika strony wartość zero. Metoda ma następującą składnię:

```
MyPageCounter.Reset OptPage
```

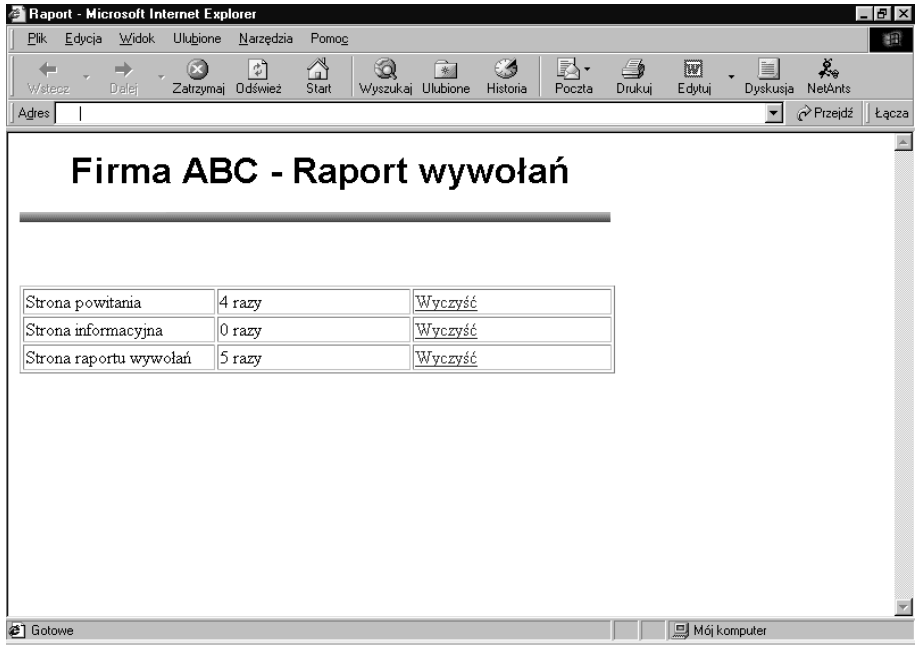
Zmienna `MyPageCounter` musi być prawidłowym obiektem `Page Counter`. Metoda posiada jeden nieobowiązkowy parametr. Określa on wirtualną ścieżkę do strony, której licznik chcesz wyzerować. Jeśli nie podasz tego parametru, wyzerowany zostanie licznik strony bieżącej.

Uzupełnijmy naszą przykładową witrynę o tę metodę, dodając do niej stronę raportu wyszczególniającą liczbę wywołań każdej ze stron i pozwalającą gościowi na zerowanie liczników. Taką stronę pokazano na rysunku 10.10. Zawiera ona w pierwszej kolumnie nazwy wszystkich stron, a dalej liczbę ich wywołań i łącza pozwalające na wyzerowanie licznika każdej ze stron. Kliknięcie łącza spowoduje wpis do licznika wartości 0, a strona zostanie ponownie wyświetlona. Jeśli na przykład klikniesz łącze *Wyczyść* dla strony informacyjnej, jej licznik zostanie skasowany, jak to pokazano na rysunku 10.11. Kod strony raportu musi dodać jeden do liczby swoich wywołań, wyzerować licznik żądanej strony i ponownie wyświetlić liczbę wywołań dla każdej strony. Główny blok kodowy jest taki:

```
<%
Option Explicit
Dim MyPageCounter
Dim CompInfoCount
Dim WelcomeCount
Dim HitReportCount
Set MyPageCounter = Server.CreateObject("IISSample.PageCounter")
MyPageCounter.PageHit
If not isempty(Request.QueryString("ClearPage")) Then
  If Request.QueryString("ClearPage") = "Welcome" Then
    MyPageCounter.Reset("/aspbook/c10/pc/html/welcome.asp")
  ElseIf Request.QueryString("ClearPage") = "CI" Then
    MyPageCounter.Reset("/aspbook/c10/pc/html/company_info.asp")
  ElseIf Request.QueryString("ClearPage") = "Report" Then
    MyPageCounter.Reset
  End If
End If
If MyPageCounter.Hits("/aspbook/c10/pc/html/company_info.asp") = 1 Then
  CompInfoCount = "1 raz"
```



Rysunek 10.10. *Strona raportu wywołań*



Rysunek 10.11. *Strona raportu po wyzerowaniu licznika*

```
else
  CompInfoCount = MyPageCounter.Hits("/aspbook/c10/pc/html/company_info.asp") & " razy"
End If
If MyPageCounter.Hits("/aspbook/c10/pc/html/welcome.asp") = 1 Then
  WelcomeCount = "1 raz"
else
  WelcomeCount = MyPageCounter.Hits("/aspbook/c10/pc/html/welcome.asp") & " razy"
End If
If MyPageCounter.Hits = 1 Then
  HitReportCount = "1 raz"
else
  HitReportCount = MyPageCounter.Hits & " razy"
End If
%>
```

Na wstępie pojawia się instrukcja `Option Explicit`:

```
Option Explicit
```

Następnie tworzona jest zmienna dla obiektu `Page Counter`:

```
Dim MyPageCounter
```

Potrzebujemy również zmiennych do zapamiętania liczby wywołań każdej ze stron:

```
Dim CompInfoCount
Dim WelcomeCount
Dim HitReportCount
```

Tworzony jest obiekt `Page Counter`:

```
Set MyPageCounter = Server.CreateObject("IISSample.PageCounter")
```

Zapisywane jest wywołanie strony raportu:

```
MyPageCounter.PageHit
```

Następnie kod sprawdza, czy któryś licznik powinien być wyzerowany. Łącza służące do zerowania liczników przekazują nazwę strony poprzez pole `ClearPage` za pomocą zbioru `QueryString` (będziesz się mógł o tym przekonać nieco później). Jeśli więc pole to nie jest puste, należy wyzerować licznik strony:

```
If not isempty(Request.QueryString("ClearPage")) Then
```

Jeśli wartością pola będzie ciąg `Welcome`, musi być wyzerowany licznik strony powitania:

```
If Request.QueryString("ClearPage") = "Welcome" Then
```

Zauważ, że metoda `Reset` musi przekazać parametr określający wirtualną ścieżkę strony, której licznik jest zerowany, ponieważ metoda ta odnosi się do strony innej niż bieżąca:

```
MyPageCounter.Reset("/aspbook/c10/pc/html/welcome.asp")
```

Podobne sprawdzenie i akcja realizowane są dla strony informacyjnej:

```
ElseIf Request.QueryString("ClearPage") = "CI" Then
  MyPageCounter.Reset("/aspbook/c10/pc/html/company_info.asp")
```

Zwróć uwagę na to, że metoda `Reset`, odnosząca się do strony raportu, nie posiada parametru. Jest to spowodowane tym, że zerujemy liczbę wywołań strony bieżącej, tak więc parametr nie jest potrzebny:

```
ElseIf Request.QueryString("ClearPage") = "Report" Then
  MyPageCounter.Reset
```

Następnie kod przygotowuje odpowiedni tekst informacji o wywołaniach strony informacyjnej:

```
If MyPageCounter.Hits("/aspbook/c10/pc/html/company_info.asp") = 1 Then
  CompInfoCount = "1 raz"
else
  CompInfoCount = MyPageCounter.Hits("/aspbook/c10/pc/html/company_info.asp") & " razy"
End If
```

To samo robione jest dla strony powitania:

```
If MyPageCounter.Hits("/aspbook/c10/pc/html/welcome.asp") = 1 Then
  WelcomeCount = "1 raz"
else
  WelcomeCount = MyPageCounter.Hits("/aspbook/c10/pc/html/welcome.asp") & " razy"
End If
```

jak również dla strony raportu:

```
If MyPageCounter.Hits = 1 Then
  HitReportCount = "1 raz"
else
  HitReportCount = MyPageCounter.Hits & " razy"
End If
```

Tabela HTML musi być wypełniona właściwym tekstem dla każdej strony.

```
<TD WIDTH=164><P>Strona powitania</TD>
<TD WIDTH=164><P><% response.write WelcomeCount %></TD>
<TD WIDTH=164><P><A HREF="./hitreport.asp?ClearPage=Welcome">
<B>Wyczyść</B></A></TD>
<TD WIDTH=164><P>Strona informacyjna</TD>
<TD WIDTH=164><P><% response.write CompInfoCount %></TD>
<TD WIDTH=164><P><A HREF="./hitreport.asp?ClearPage=CI">
<B>Wyczyść</B></A></TD>
<TD WIDTH=164><P>Strona raportu wywołań</TD>
<TD WIDTH=164><P><% response.write HitReportCount %></TD>
<TD WIDTH=164><P><A HREF="./hitreport.asp?ClearPage=Report">
<B>Wyczyść</B></A></TD>
```

Zauważ, że pierwsza komórka tabeli HTML zawiera jedynie nazwę strony:

```
<TD WIDTH=164><P>Strona powitania</TD>
```

Następnie do przeglądarki wpisywana jest informacja o trafnych wywołaniach:

```
<TD WIDTH=164><P><% response.write WelcomeCount %></TD>
```

Zwróć uwagę, że łącze zerowania licznika odnosi się do tej samej strony. Ponadto poprzez zbiór `QueryString` przekazywana jest wraz z łączem nazwa tej strony, której licznik należy wyzerować:

```
<TD WIDTH=164><P><A HREF="./hitreport.asp?ClearPage=Welcome">
<B>Wyczyść</B></A></TD>
```

Składnik Counters

Składnik Counters zapewnia prosty interfejs do przechowywania liczb całkowitych, które możesz odczytywać, zapisywać, usuwać i inkrementować na wszystkich stronach Twojej witryny, a nawet poza granicami aplikacji ASP. Tak jak to było w wypadku ostatniego składnika, również ten jest częścią zestawu IIS Resource Kit. Oto jego lokalizacja:

\IIS Resource Kit\Component\Counters\DLL\i386\counters.dll

Umieść plik w katalogu *WinNT\System32*, a następnie zarejestruj składnik przy użyciu polecenia *Uruchom* (*Run*) z menu *Start*, wprowadzając zapis *Regsvr32 counters.dll*. Po pomyślnym zarejestrowaniu składnika możesz zacząć z niego korzystać.

Składnię do tworzenia składnika `Counters` przedstawiono poniżej:

```
set MyCounter = Server.CreateObject("MSWC.Counters")
```

Instancja składnika `Counters` tworzona jest przy użyciu metody `CreateObject` obiektu `Server`. Metodzie przekazuje się ciąg zawierający aplikację i klasę — `MSWC.Counters`. Obiekt klasy `Counters` jest zwracany w zmiennej `MyCounter`.

Składnik `Counters` przechowuje wartości liczników w pliku tekstowym położonym w tym samym miejscu, co plik biblioteki. Wartości te dostępne są na każdej stronie Twojej witryny WWW, nawet jeśli witryna umiejscowiona jest w innej aplikacji ASP.

Takie zmienne mają zasięg przekraczający zasięg wszystkich innych omówionych do tej pory. Zwróciliśmy już uwagę na zmienne obiektu `Session`, które są dostępne dla wszystkich stron aplikacji ASP, lecz w czasie trwania określonej sesji. Przyjrzeliśmy się również zmiennym obiektu `Application` osiągalnym dla wszystkich stron i w czasie trwania wszystkich sesji, ale w obrębie jednej aplikacji ASP. Natomiast zmienne licznika przekraczają te ograniczenia, ponieważ są dostępne dla wszystkich stron bez względu na to, do której aplikacji ASP należą.

Licznik tworzy się, wyszukując jego wartości lub je ustawiając. Wyszukiwanie wartości licznika realizuje się przy użyciu metody `Get` składnika `Counters`:

```
TheValue = MyCounter.Get(NameOfCounter)
```

Zmienna `MyCounter` musi być instancją składnika `Counters`. Metoda `Get` pobiera jeden parametr, którym jest nazwa licznika. Metoda zwraca aktualną wartość licznika i umieszcza ją w zmiennej `TheValue`. Jeśli licznik o podanej nazwie nie istnieje, zostaje utworzony i zwracana jest wartość 0.

W tym przykładzie:

```
<%
Option Explicit
Dim MyCounter
set MyCounter = Server.CreateObject("MSWC.Counters")
Response.Write MyCounter.Get("Product10Views")
%>
```

w przeglądarce wyświetlona zostanie wartość licznika `Product10Views`. Jeśli taki licznik nie istnieje, zostaje utworzony, a w przeglądarce wyświetlana jest liczba 0.

Aby wpisać do licznika jakąś określoną wartość, możesz wykorzystać metodę Set. Ustawia ona w liczniku wskazaną przez Ciebie wartość, a jeśli licznik o podanej nazwie nie istnieje, składnik tworzy nowy licznik. Metoda ma następującą wartość:

```
MyCounter.Set NameOfCounter, ValueOfCounter
```

Zmienna MyCounter musi być prawidłowym obiektem Counters. Metodzie Set przekazywane są dwa parametry: pierwszy jest nazwą licznika, którego wartość chcemy ustawić, drugi jest tą właśnie wartością.

Na przykład:

```
<%
Option Explicit
Dim MyCounter
set MyCounter = Server.CreateObject("MSWC.Counters")
MyCounter.Set "AllPageViews", 123
%>
```

Ten kod wpisze wartość 123 do licznika o nazwie AllPageViews. Jeśli taki licznik nie istnieje, zostaje utworzony i wpisywana jest do niego żądana wartość.

Kolejną metodą składnika jest metoda Increment, która zwiększa wartość licznika o jeden. Metoda ma następującą składnię:

```
MyCounter.Increment NameOfCounter
```

Zmienna MyCounter musi być prawidłową instancją składnika Counters. NameOfCounter jest nazwą inkrementowanego licznika.

Zwróć uwagę na ten przykład:

```
<%
Option Explicit
Dim MyCounter
set MyCounter = Server.CreateObject("MSWC.Counters")
MyCounter.Increment "AllPageViews"
%>
```

Tutaj stan licznika AllPageViews jest zwiększany o jeden, jeśli więc jego wartość wynosiła wcześniej 123, teraz będzie to 124.

Składnik Counters ma jeszcze jedną metodę, Remove, która usuwa licznik z pliku liczników. Jeśli po użyciu tej metody następuje odwołanie do usuniętego licznika, jest on tworzony na nowo, a jego wartość wynosi 0 (jeśli licznik jest tworzony przy użyciu metody Set).

Metoda ma następującą składnię:

```
MyCounter.Remove NameOfCounter
```

Zmienna MyCounter musi być instancją składnika Counters. Metoda pobiera jeden parametr — nazwę usuwanego licznika.

Na przykład:

```
<%
Option Explicit
```

```
Dim MyCounter
set MyCounter = Server.CreateObject("MSWC.Counters")
MyCounter.Remove "AllPageViews"
%>
```

W tym kodzie licznik `AllPageViews` jest usuwany z pamięci. Jeśli usunięty licznik jest następnie wywoływany przy użyciu metody `Set` lub `Get`, jest tworzony ponownie.

W następnym bloku kodu użyto wszystkich omówionych metod, aby zademonstrować działanie składnika.

```
<%
Option Explicit
Dim MyCounter
set MyCounter = Server.CreateObject("MSWC.Counters")
Response.Write "<B>Teraz tworzony jest nowy licznik.<P>"
Response.Write "Jego wartość wynosi: "
Response.Write MyCounter.Get("Counter1") & "<P>"
MyCounter.Set "Counter1", 33
Response.Write "Teraz wpisana została do niego wartość: "
Response.Write MyCounter.Get("Counter1") & "<P>"
MyCounter.Increment "Counter1"
Response.Write "Licznik został inkrementowany: "
Response.Write MyCounter.Get("Counter1") & "<P>"
MyCounter.Remove "Counter1"
Response.Write " Licznik został usunięty. " & "<P>"
MyCounter.Set "Counter2", 250
Response.Write "Ustawiona została wartość nowego licznika : "
Response.Write MyCounter.Get("Counter2") & "<P>"
%>
```

Na początku umieszczamy instrukcję `Option Explicit`:

```
Option Explicit
```

Tworzona jest zmienna przechowująca obiekt `Counters`:

```
Dim MyCounter
```

Zmienna zostaje zainstancjonowana:

```
set MyCounter = Server.CreateObject("MSWC.Counters")
```

Poprzez wywołanie metody `Get` tworzony jest licznik i wyświetlana jest jego początkowa wartość:

```
Response.Write "<B>Teraz tworzony jest nowy licznik.<P>"
Response.Write "Jego wartość wynosi: "
Response.Write MyCounter.Get("Counter1") & "<P>"
```

W ten sam licznik wpisywana jest liczba 33:

```
MyCounter.Set "Counter1", 33
```

która następnie wyświetlana jest w przeglądarce przy użyciu metody `Get`:

```
Response.Write "Teraz wpisana została do niego wartość: "
Response.Write MyCounter.Get("Counter1") & "<P>"
```

Licznik `Counter1` jest inkrementowany:

```
MyCounter.Increment "Counter1"
```

Wartość licznika ponownie jest wyświetlana w przeglądarce:

```
Response.Write "Licznik został inkrementowany: "
Response.Write MyCounter.Get("Counter1") & "<P>"
```

Wykorzystana zostaje metoda `Remove` w celu usunięcia licznika `Counter1`:

```
MyCounter.Remove "Counter1"
Response.Write " Licznik został usunięty. " & "<P>"
```

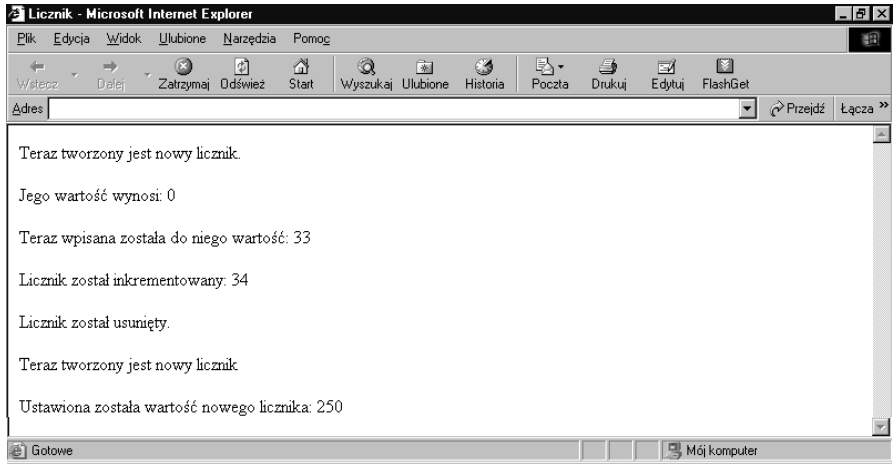
Następnie tworzony jest drugi licznik, który będzie wyświetlany na kolejnej stronie, znajdującej się w obrębie innej aplikacji ASP:

```
MyCounter.Set "Counter2", 250
```

a wartość tego licznika prezentowana jest tutaj:

```
Response.Write "Ustawiona została wartość nowego licznika : "
Response.Write MyCounter.Get("Counter2") & "<P>"
```

Wygląd strony przedstawia rysunek 10.12.



Rysunek 10.12. *Rezultat działania kodu licznika*

Pamiętaj, że liczniki mają zasięg przekraczający zakres aplikacji ASP, tak więc na kolejnej stronie należącej do innej aplikacji ASP możemy odwoływać się do licznika stworzonego w tym kodzie:

```
<%
Option Explicit
Dim MyCounter
set MyCounter = Server.CreateObject("MSWC.Counters")
Response.Write "<B>Ta strona wyszukuje wartość drugiego licznika "
Response.Write "stworzonego w innej aplikacji ASP. "
Response.Write "<P>Jego wartość: "
Response.Write MyCounter.Get("Counter2") & ".<P>"
%>
```

Na wstępie pojawia się instrukcja `Option Explicit`:

```
Option Explicit
```

Następnie tworzona jest zmienna licznika:

```
Dim MyCounter
```

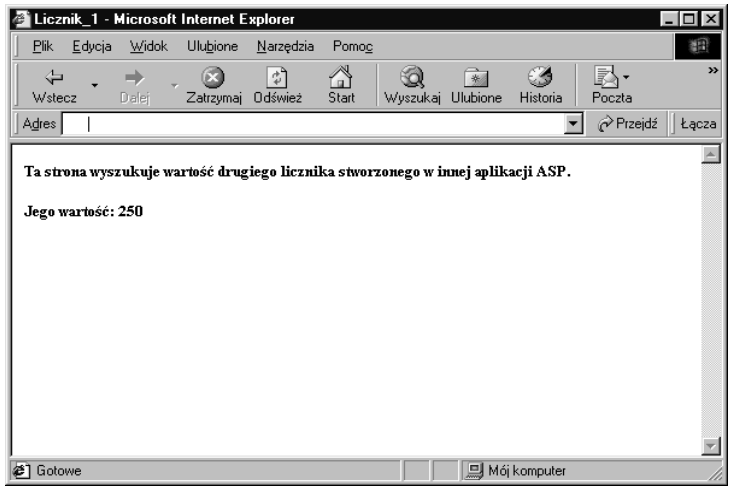
i zostaje zainicjalizowana:

```
set MyCounter = Server.CreateObject("MSWC.Counters")
```

Wartość licznika utworzonego na innej stronie, wyszukana poprzez metodę `Get`, jest wpisywana do przeglądarki:

```
Response.Write "<B>Ta strona wyszukuje wartość drugiego licznika "
Response.Write "stworzonego w innej aplikacji ASP. "
Response.Write "<P>Jego wartość: "
Response.Write MyCounter.Get("Counter2") & ".<P>"
```

Rezultat działania tej strony, będącej częścią innej aplikacji ASP, pokazany został na rysunku 10.13.



Rysunek 10.13. *Wygląd drugiej strony licznika*

Składnik Content Linking

Składnik Content Linking pozwala na połączenie stron w taki sposób, aby mogły być przeglądane w seriach, sekwencyjnie. Przykładowo książka składa się ze stron ułożonych sekwencyjnie, a plik pomocy składa się z serii stron. Jeśli masz taki scenariusz, to często będziesz chciał dysponować łączami, które dynamicznie przenosiłyby gości do strony następnej lub z powrotem, do poprzedniej. Być może będziesz chciał również mieć tabelę zawartości dla serii stron. Składnik `Content Linking` łączy w sobie metody umożliwiające realizację takich zadań.

Składnik jest częścią standardowej instalacji IIS. Jego plik biblioteki nosi nazwę *nextlink.dll*. Podstawą działania składnika jest plik tekstowy, zawierający informacje o wszystkich stronach, które chcesz połączyć, oraz o kolejności, w jakiej te strony mają być połączone. W tym specjalnym pliku tekstowym podajesz nie tylko nazwę strony, ale również jej adres URL. Plik nie musi nosić jakiejś określonej, specjalnej nazwy — musi być jedynie dostępny w strukturze katalogu tej witryny WWW, w której będzie wykorzystany.

Każdy wiersz pliku tekstowego ma następującą strukturę:

```
PageURL Description Comment
```

Każde pole oddzielone jest od siebie znakiem tabulacji, a każda pozycja musi być umieszczona w osobnym wierszu tekstowego pliku indeksu. Pierwsze pole każdego wiersza przechowuje adres URL danej strony. Drugie pole jest nazwą lub opisem strony. Trzecie, które nie jest wymagane, może zawierać dowolny komentarz, który po prostu jest ignorowany przez składnik.

Spójrz na ten przykładowy plik indeksu:

```
TOC.asp   Table of Contents Page
Page1.asp   Page 1 of Chapter 1
Page2.asp   Page 2 of Chapter 1
```

Zauważ, że adres URL i opis są od siebie oddzielone znakiem tabulacji. Ponadto zwróć uwagę na kolejność pozycji. Jest to ważne, ponieważ metody nawigacyjne używają tej kolejności w celu określenia, która strona jest następna, a która poprzednia. Stwierdziłem, że najlepszym miejscem do zapisania tego pliku jest katalog, w którym umieszczone są używające go strony. Dzięki temu prostsze staje się używanie metod, ponieważ ścieżkę dostępu do strony można przedstawić za pomocą samej jej nazwy.

W Twoim kodzie możesz stworzyć instancję składnika `Content Linking`, który będzie używał tego pliku tekstowego w celu określenia koniecznych do podjęcia działań. Aby stworzyć instancję składnika `Content Linking`, należy użyć takiego kodu:

```
Set MyCL = Server.CreateObject("MSWC.NextLink")
```

Jak to było w wypadku innych składników, do tworzenia instancji `Content Linking` używamy metody `CreateObject` obiektu `Server`. Tutaj instancja zostanie umieszczona w zmiennej obiektowej `MyCL`. Od tej chwili możesz używać pliku tekstowego wraz z ze stworzonym obiektem dla potrzeb nawigacyjnych tego typu aplikacji.

Składnik posiada osiem metod, które omówiono poniżej i sumarycznie przedstawiono w tabeli 10.2.

Metoda `GetListCount` zwraca liczbę pozycji w pliku indeksu. Jej składnia jest następująca:

```
TheCount = MyCL.GetListCount(Path2IndexFile)
```

Metodzie przekazywana jest wirtualna ścieżka do pliku indeksu (`Path2IndexFile`), jeśli więc plik ten położony jest w tym samym miejscu, co strona dokonująca wywołania, wprowadzona musi być jedynie nazwa pliku. Zwracaną przez metodę wartością jest liczba pozycji w pliku indeksu, jeśli więc stworzysz taki kod odnoszący się do pliku omówionego wcześniej:

```
Response.Write MyCL.GetListCount("CLList.txt")
```

to do przeglądarki wysłana zostanie liczba 3.

Tabela 10.2. *Metody składnika Content Linking*

Metoda	Działanie
`GetListCount`	Zwraca liczbę pozycji w pliku indeksu.
`GetListIndex`	Zwraca numer pozycji bieżącej strony w pliku indeksu.
`GetNextURL`	Zwraca adres URL strony następnej względem strony bieżącej.
`GetNextDescription`	Zwraca opis strony następnej względem strony bieżącej.
`GetPreviousURL`	Zwraca URL strony poprzedniej względem strony bieżącej.
`GetPreviousDescription`	Zwraca opis strony poprzedniej względem strony bieżącej.
`GetNthURL`	Zwraca URL pozycji indeksu na podstawie liczby przekazanej metodzie.
`GetNthDescription`	Zwraca opis pozycji indeksu na podstawie liczby przekazanej metodzie.

Następną metodą jest `GetListIndex`, która zwraca numer pozycji bieżącej strony w pliku indeksu. Metoda ma następującą składnię:

```
TheIndex = MyCL.GetListIndex(Path2IndexFile)
```

Zmienna `MyCL` musi być prawidłową instancją składnika `Content Linking`. Metodzie przekazywany jest parametr `Path2IndexFile`, który stanowi ścieżkę wirtualną do tekstowego pliku indeksu. Gdybyś poszukiwał strony *Page1.asp* w pliku indeksowym opisanym wcześniej, to następujący kod:

```
Response.Write MyCL.GetListIndex("CLList.txt")
```

wpisałby do przeglądarki liczbę 2.

Metoda `GetNextURL` zwraca URL następnej pozycji pliku indeksu. Jeśli bieżąca strona jest ostatnią na liście, metoda `GetNextURL` zwróci pierwszą stronę z listy, przenosząc gościa na początek pętli. Metoda ma następującą składnię:

```
TheURL = MyCL.GetNextURL(Path2IndexFile)
```

Zmienna `MyCL` musi być prawidłową instancją składnika `Content Linking`. Parametr `Path2IndexFile` jest ścieżką wirtualną do pliku indeksu. Metoda zwraca adres URL następnej pozycji w pliku indeksu, jeśli więc znajdowalibyśmy się na stronie *Page1.asp*, to ten kod:

```
Response.Write MyCL.GetNextURL("CLList.txt")
```

wpisze do przeglądarki *Page2.asp*.

Metoda `GetNextDescription` wyszukuje pole opisu tej pozycji pliku indeksu, która następuje za bieżącą stroną. Jeśli bieżąca strona jest stroną ostatnią, wtedy składnik zwraca opis pierwszej strony z listy. Metoda ma następującą składnię:

```
TheDescription = MyCL.GetNextDescription(Path2IndexFile)
```

Zmienna `MyCL` musi być instancją składnika `Content Linking`. Metodzie przekazywana jest ścieżka do pliku indeksu, a wartością zwracaną jest opis następnej w pliku indeksu strony. Jeśli więc znajdowałbyś się na stronie *Page1.asp* z poprzedniego pliku indeksu, to taki kod:

```
Response.Write MyCL.GetNextDescription("CLList.txt")
```

wpisałby do przeglądarki następujący tekst:

```
Page 2 of Chapter 1
```

Metoda GetPreviousURL zwraca URL poprzedniej pozycji w tekstowym pliku indeksu. Jeśli bieżąca strona jest pierwsza na liście, metoda GetPreviousURL zwróci adres URL ostatniej strony z listy, przechodząc w pętli na koniec pliku. Metoda ma następującą składnię:

```
TheURL = MyCL.GetPreviousURL(Path2IndexFile)
```

Zmienna MyCL musi być instancją składnika Content Linking. Parametr Path2IndexFile jest ścieżką wirtualną do pliku indeksu. Metoda zwraca URL poprzedniej pozycji pliku indeksu, jeśli więc znajdowalibyśmy się na stronie *Page1.asp* i napisalibyśmy taki kod:

```
Response.Write MyCL.GetNextURL("CLList.txt")
```

w przeglądarce ukazałby się URL *TOC.asp*.

Metoda GetPreviousDescription zwraca pole opisu tej pozycji pliku indeksu, która poprzedza bieżącą stronę. Jeśli strona bieżąca jest stroną pierwszą, to składnik zwróci opis strony znajdującej się na końcu listy. Metoda ma następującą składnię:

```
TheDescription = MyCL.GetPreviousDescription(Path2IndexFile)
```

Zmienna MyCL musi być instancją składnika Content Linking. Metodzie przekazywana jest ścieżka do pliku indeksu. Zwracaną wartością jest opis strony poprzedniej w pliku indeksu, jeśli więc znajdowałbyś się na stronie *Page1.asp*, to następujący wiersz kodu:

```
Response.Write MyCL.GetPreviousDescription("CLList.txt")
```

wysłałby do przeglądarki tekst:

```
Table of Contents Page
```

Zostały nam do omówienia jeszcze dwie metody, GetNthURL i GetNthDescription, które również zwracają odpowiednio URL i opis z pliku indeksu. Jednak w tych metodach określasz numer pozycji, która ma być zwrócona przez funkcję.

Metoda GetNthURL ma następującą składnię:

```
TheURL = MyCL.GetNthURL(Path2ToIndexFile, NumericPosition)
```

Zmienna MyCL musi być kopią składnika Content Linking. Metodzie przekazywane są dwa parametry: pierwszy z nich to wirtualne położenie tekstowego pliku indeksu; drugi parametr zapamiętuje numer pozycji, której URL chcesz otrzymać i ten właśnie URL jest zwracany przez metodę.

Jeśli przykładowo stworzyłeś taki kod, odnoszący się do pliku indeksu omówionego wcześniej:

```
Response.Write MyCL.GetNthURL("CLList.txt", 2)
```

to do przeglądarce pojawi się łącze *Page1.asp*.

Metoda GetNthDescription ma następującą składnię:

```
TheDescription = MyCL.GetNthDescription(Path2ToIndexFile, NumericPosition)
```

Zmienna MyCL musi być instancją składnika Content Linking. Metodzie przekazywane są dwa parametry: pierwszy z nich to wirtualne położenie tekstowego pliku indeksu; drugi parametr zapamiętuje numer pozycji, której opis chcesz otrzymać i ten właśnie opis jest zwracany przez metodę.

Jeśli stworzysz taki przykładowy kod, odnoszący się do omówionego wcześniej tekstowego pliku indeksu:

```
Response.Write MyCL.GetNthDescription("CLList.txt", 2)
```

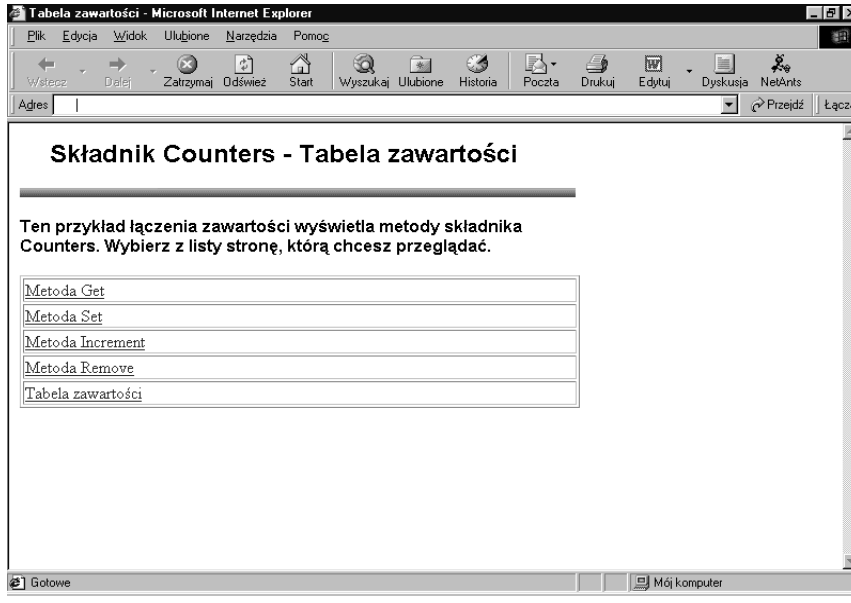
to jego wynikiem będzie pojawienie się w przeglądarce tekstu `Page 1 of Chapter 1`.

Jeśli połączysz ze sobą te metody, możesz stworzyć całkiem niezłą aplikację ASP, umożliwiającą gościom kolejne przechodzenie przez strony Twojej witryny, zarówno w przód, jak i w tył oraz wyświetlającą tabelę zawartości. Zwróćmy uwagę na taką witrynę WWW, która opisuje omówione w poprzednim podrozdziale metody składnika `Counters`.

Poniżej przedstawiony jest tekst pliku indeksu, który położony jest w tym samym katalogu, co strony ASP:

```
get_method.asp   Metoda Get
set_method.asp   Metoda Set
increment_method.asp   Metoda Increment
remove_method.asp   Metoda Remove
get_method.asp   Metoda Get
toc.asp   Tabela zawartości
```

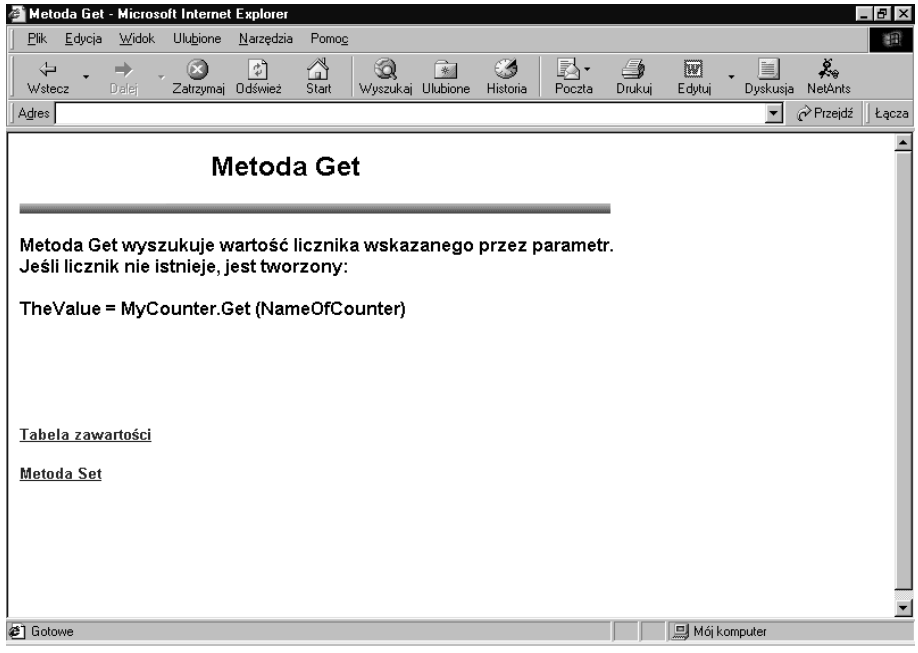
Wyszczególnione zostały cztery metody wraz ze swoimi adresami URL i opisami. Tabelę zawartości, stworzoną na podstawie indeksu, prezentuje rysunek 10.14.



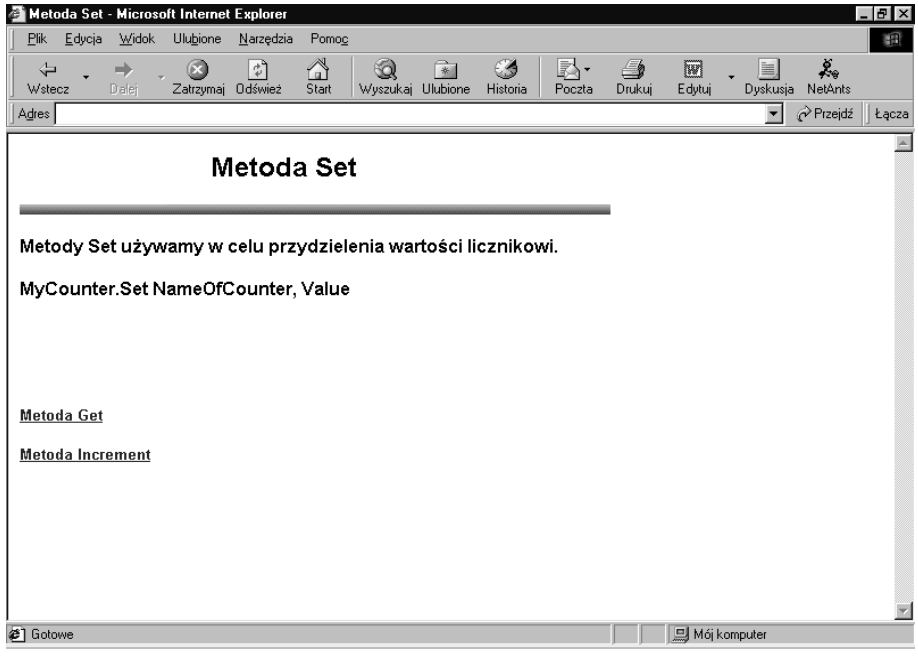
Rysunek 10.14. *Strona tabeli zawartości witryny wykorzystującej składnik Content Linking*

Zwróć uwagę na to, że kolejność pozycji na rysunku 10.14. jest taka sama, jak w tekstowym pliku indeksu. Kiedy goście klikną łącze *Metoda Get*, ujrzą stronę pokazaną na rysunku 10.15.

Zwróć uwagę na łącza znajdujące się na stronie metody `Get`. Łącze ze stroną poprzednią odsyła gości do tabeli zawartości, ponieważ strona metody `Get` jest pierwszą stroną na liście. Łącze ze stroną następną wyświetla tekst *Metoda Set*, który odsyła gości do strony pokazanej na rysunku 10.16.

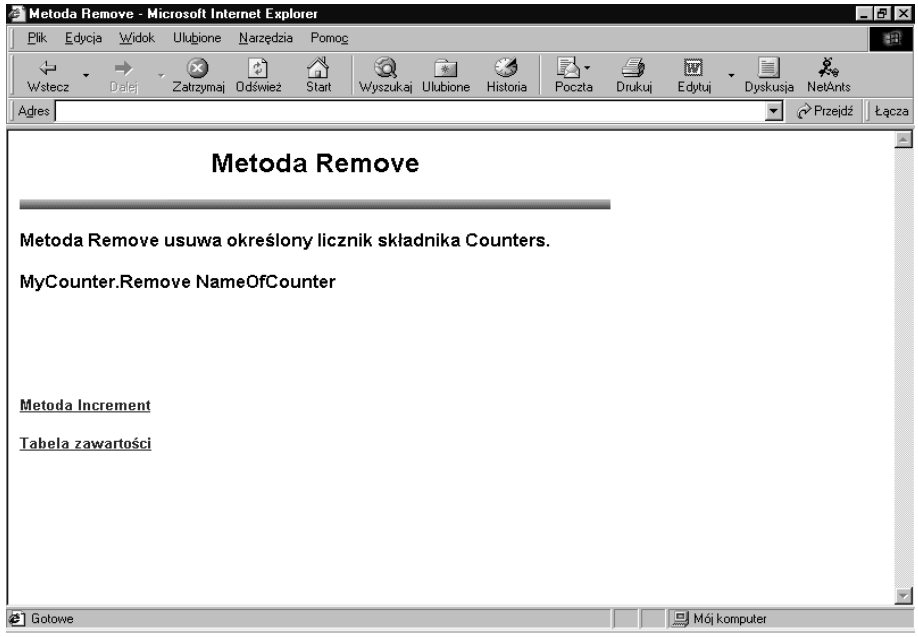


Rysunek 10.15. *Strona metody Get witryny wykorzystującej składnik Content Linking*



Rysunek 10.16. *Strona metody Set witryny wykorzystującej składnik Content Linking*

Za stroną metody `Increment` pojawia się strona metody `Remove`, pokazana na rysunku 10.17. Zauważ, że łącze ze stroną następną odsyła gości do tabeli zawartości, ponieważ jest to ostatnia pozycja z listy stron. Łącze z poprzednią stroną przenosi gości do strony metody `Increment`.



Rysunek 10.17. *Strona metody Remove witryny wykorzystującej składnik Content Linking*

Kod użyty do stworzenia tej witryny składa się z dwóch części: jedna z nich służy wykonaniu strony tabeli zawartości; następnie strony wszystkich metod wykorzystują tę samą strukturę kodu w celu wyświetlenia łącz ze stroną następną i poprzednią.

Pierwszy blok kodu dla tabeli zawartości jest następujący:

```
<%
Option Explicit
Dim MyCL
Dim TheCount
Dim I
Set MyCL = Server.CreateObject("MSWC.NextLink")
TheCount = MyCL.GetListCount("CLList.txt")
%>
```

Na początku pojawia się instrukcja `Option Explicit`:

```
Option Explicit
```

Następnie tworzona jest zmienna przechowująca obiekt `Content Linking`:

```
Dim MyCL
```

Kolejna zmienna będzie przechowywała całkowitą liczbę pozycji w tekstowym pliku indeksu:

```
Dim TheCount
```

Inna zmienna jest tworzona w celu zapamiętania liczby iteracji w drugim bloku kodu:

```
Dim I
```

Tworzona jest instancja składnika `Content Linking`:

```
Set MyCL = Server.CreateObject("MSWC.NextLink")
```

Do zmiennej `TheCount` wpisywana jest liczba pozycji w tekstowym pliku indeksu:

```
TheCount = MyCL.GetListCount("CLList.txt")
```

Następny blok kodu tworzy tabelę HTML, zawierającą wszystkie pozycje wyszczególnione w pliku indeksu.

```
<%
For I = 1 to TheCount
%>
  <TR>
    <TD WIDTH=423><P><A HREF="
      <% Response.Write MyCL.GetNthURL("CLList.txt", i) %>">
      <B><% Response.Write MyCL.GetNthDescription("CLList.txt", i) %>
      </B></A>
    </TD>
  </TR>
<%
Next
%>
```

Kod wykorzystuje pętlę `For`, w której przechodzi kolejno przez wszystkie pozycje tekstowego pliku indeksu:

```
For I = 1 to TheCount
```

Następnie każda pozycja wpisywana jest do tabeli HTML. Adres URL wpisywany jest przy użyciu metody `GetNthURL`, a opis — przy użyciu metody `GetNthDescription`. Kod przechodzi w pętli do następnego rekordu:

```
Next
```

Kod wszystkich innych stron jest taki sam. Pierwszy blok kodu tworzy składnik `Content Linking`.

```
<%
Option Explicit
Dim MyCL
Set MyCL = Server.CreateObject("MSWC.NextLink")
%>
```

Na początku instrukcja `Option Explicit`:

```
Option Explicit
```

Następnie tworzona jest zmienna, która będzie przechowywała składnik `Content Linking`:

```
Dim MyCL
```

Tworzony jest obiekt:

```
Set MyCL = Server.CreateObject("MSWC.NextLink")
```

Kolejny blok kodu wpisuje do przeglądarki łącza ze stroną poprzednią i następną.

```
<P><A HREF="<% Response.Write MyCL.GetPreviousURL("CLList.txt")%>"><B>
<% Response.Write MyCL.GetPreviousDescription("CLList.txt") %></A></B>
<P><A HREF="<% Response.Write MyCL.GetNextURL("CLList.txt") %>"><B>
<% Response.Write MyCL.GetNextDescription("CLList.txt") %></A></B>
```

Na początku wpisywane jest łącze ze stroną poprzednią:

```
<P><A HREF="<% Response.Write MyCL.GetPreviousURL("CLList.txt")%>"><B>
```

Następnie opis łącza ze stroną poprzednią:

```
<% Response.Write MyCL.GetPreviousDescription("CLList.txt") %></A></B>
```

Łącze ze stroną następną:

```
<P><A HREF="<% Response.Write MyCL.GetNextURL("CLList.txt") %>"><B>
```

W końcu opis łącza ze stroną następną:

```
<% Response.Write MyCL.GetNextDescription("CLList.txt") %></A></B>
```

Składnik Content Rotator

Składnik Content Rotator umożliwia prezentowanie innej zawartości za każdym razem, kiedy strona jest ładowana. Przykładem zastosowania tego składnika może być włączenie do strony cytatu dnia, odmiennych wiadomości powitania lub porady dnia. Te same funkcje można realizować, ładując dane z bazy, jednak ten składnik dostarcza alternatywne mechanizmy, które nie wymagają ładowania bazy danych tylko po to, aby zmienić poradę dnia.

Składnik nie jest częścią standardowej wersji instalacyjnej serwera IIS, można go znaleźć w zestawie IIS Resource Kit. Biblioteka jest położona w następującym miejscu:

\IIS Resource Kit\Component\Content Rotator\DLL\i386\controt.dll

Umieść plik w katalogu *WinNT\System32*, a następnie zarejestruj składnik przy użyciu polecenia *Uruchom* (*Run*) z menu *Start*, wpisując `Regsvr32 controt.dll`.

Kiedy już go zainstalujesz, możesz stworzyć jego kopię za pomocą następującego wiersza kodu:

```
Set objCR = Server.CreateObject("IISSample.ContentRotator")
```

Do stworzenia instancji składnika używamy metody `CreateObject` obiektu `Server`. W tym przykładzie zmienna `objCR` będzie stanowiła instancję tego składnika.

Składnik bazuje na oddzielnym pliku tekstowym, łączącym w sobie różną zawartość, która ma zostać wyświetlona. W pliku nadajesz danej treści poziom odniesienia, określający, jak często powinna się ona ukazywać względem reszty zawartości. Możesz również wpisać swój komentarz, a następnie wprowadzić samą zawartość.

Zawartość umieszczana jest wprost w HTML-u, możesz więc umieścić znaczniki HTML w pozycjach pliku. Plik powinien być zapisany jako zwykły plik tekstowy w takim miejscu, aby można było go adresować ze strony, która będzie korzystała z jego zawartości. Przekonałem się, że najlepiej umieścić go w tym samym katalogu, co strona korzystająca z zawartości, wtedy wywołanie pliku jest znacznie prostsze.

Każda pozycja pliku ma następującą strukturę:

```
%% #Ranking // Komentarz
Tekst zawartości
```

Każda pozycja rozpoczyna się podwójnym znakiem procentu. Po tym znaczniku może następować *ranking* (nie jest wymagany), który określa częstość pojawiania się jednej pozycji względem innych znajdujących się na liście; jeśli nie określisz rankingu, domyślnie ustalana jest dla niego wartość 1.

Po rankingu lub wartości odniesienia pojawia się komentarz, który również nie jest wymagany. Następny wiersz to treść pozycji. Może ona obejmować wiele wierszy, które traktowane są jako treść, dopóki ponownie nie pojawi się podwójny znak procentu lub dopóki plik się nie skończy.

Składnik ma dwie metody służące wyszukiwaniu zawartości z pliku. Pierwszą z nich jest metoda `ChooseContent`, zwracająca pozycję z listy zawartości. Pojawienie się zwracanej pozycji jest bardziej prawdopodobne, jeśli ma ona wyższy ranking. Metoda ma następującą składnię:

```
TheText = objCR.ChooseContent(Path2ContentFile)
```

Zmienna `objCR` musi być instancją `Content Rotator`. Metodzie przekazywany jest parametr `Path2ContentFile`. Jest to wirtualna ścieżka do pliku zawartości. Metoda zwraca tekst poszczególnych pozycji z pliku zawartości.

Druga metoda, `GetAllContent`, zwraca zawartość wszystkich pozycji pliku i przesyła je wprost do przeglądarki. Metoda ma następującą składnię:

```
objCR.GetAllContent(Path2ContentFile)
```

Zmienna `objCR` musi być instancją kopią składnika `Content Rotator`. Parametr `Path2ContentFile` jest ścieżką do tekstowego pliku zawartości. Metoda wpisuje całą zawartość prosto do przeglądarki. Każda pozycja zostanie oddzielona linią poziomą (znacznik HR HTML-a).

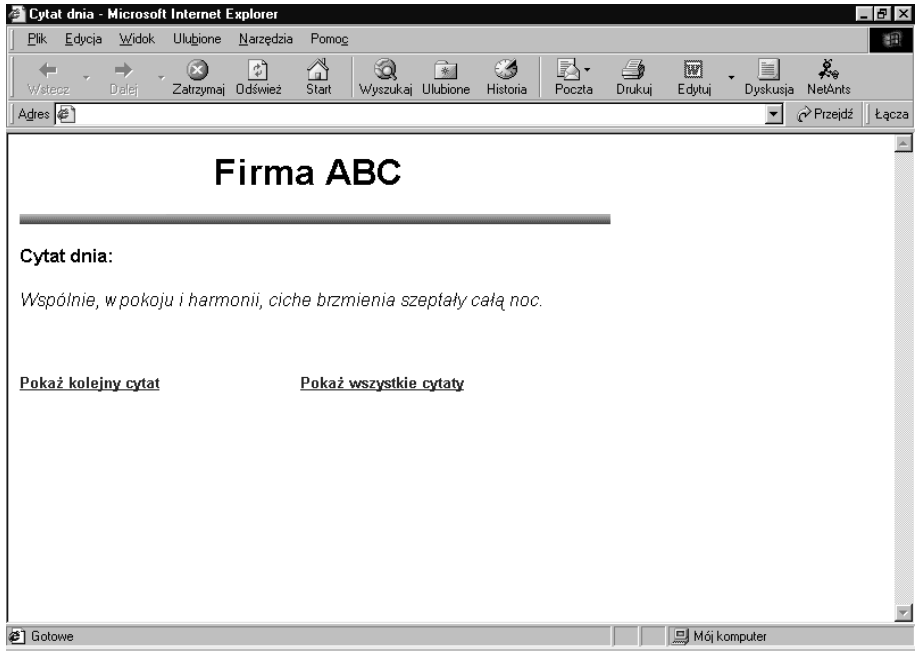
Spójrzmy na próbną stronę cytatu dnia, która wykorzystuje ten składnik. Kiedy goście wywołają stronę po raz pierwszy, ujrzą widok pokazany na rysunku 10.18.

Jeśli goście klikną łącze *Pokaż kolejny cytat*, ujrzą tekst pokazany na rysunku 10.19.

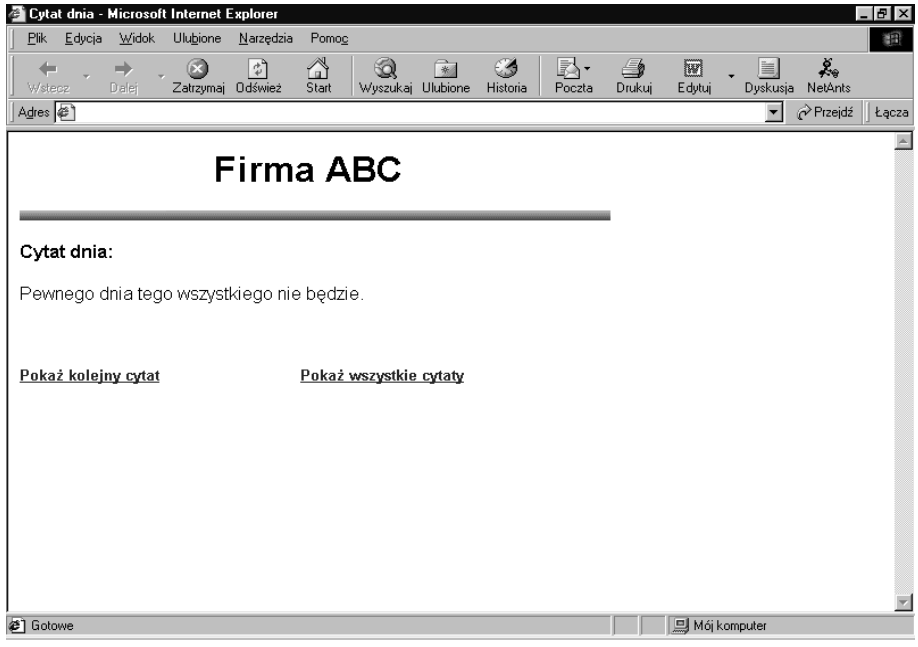
Jeśli goście klikną łącze *Pokaż wszystkie cytaty*, zobaczą stronę pokazaną na rysunku 10.20. Strona używa pliku tekstowego, zawierającego te cytaty, jak to pokazano tutaj:

```
%% #2 // To jest komentarz
<B><FONT FACE="Arial,Helvetica,Univers,Zurich BT">
Żyj tak, jak gdyby każdy dzień był twoim pierwszym i ostatnim dniem.
</FONT></B>

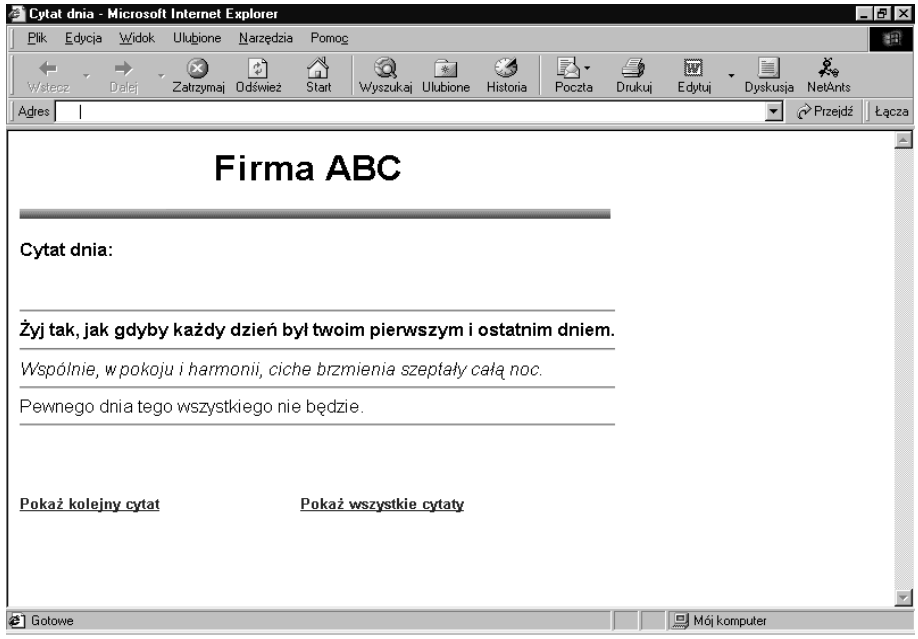
%% #3
<I><FONT FACE="Arial,Helvetica,Univers,Zurich BT">
Wspólnie, w pokoju i harmonii, ciche brzmienia szeptały całą noc.
</FONT></I>
```



Rysunek 10.18. *Pierwsza strona cytatu dnia*



Rysunek 10.19. *Strona ukazująca się po kliknięciu łącza Pokaż kolejny cytat*



Rysunek 10.20. *Strona ukazująca wszystkie cytaty*

```
%% #1 // Ten cytat ma najmniejszą wartość
<FONT FACE="Arial,Helvetica,Univers,Zurich BT">
Pewnego dnia tego wszystkiego nie będzie.
</FONT>
```

Pierwsza pozycja ma ranking 2. Jeśli dodamy do siebie wszystkie liczby rankingu, otrzymamy 6. Z wartością 2 — pierwsza pozycja ma 33 procent (dwie szóste) szans na pojawienie się w przeglądarce:

```
%% #2 // To jest komentarz
<B><FONT FACE="Arial,Helvetica,Univers,Zurich BT">
Żyj tak, jak gdyby każdy dzień był twoim pierwszym i ostatnim dniem.
</FONT></B>
```

Zauważ, że pozycja zawiera komentarz, a treść pozycji zajmuje więcej niż jeden wiersz.

Druga pozycja ma poziom 3, więc ma 50 procent (trzy szóste) szans na wyświetlenie. Zauważ, że ta pozycja nie ma komentarza:

```
%% #3
<I><FONT FACE="Arial,Helvetica,Univers,Zurich BT">
Wspólnie, w pokoju i harmonii, ciche brzmienia szeptały całą noc.
</FONT></I>
```

Ostatnia pozycja ma poziom 1, powinna więc pojawiać się tylko raz na sześć wyświetleń strony:

```
%% #1 // Ten cytat ma najmniejszą wartość
<FONT FACE="Arial,Helvetica,Univers,Zurich BT">
Pewnego dnia tego wszystkiego nie będzie.
</FONT>
```

Kod strony cytatu dnia wyświetla jedną lub wszystkie pozycje pliku, w zależności od tego, jakie łącze kliknął gość. Pierwszy blok kodu strony tworzy składnik `Content Rotator`.

```
<%
Option Explicit
Dim objCR
Set objCR = Server.CreateObject("IISSample.ContentRotator")
%>
```

Kod rozpoczynamy w tradycyjny sposób — od instrukcji `Option Explicit`. Kiedy używasz tej instrukcji, pamiętaj, że musi być ona pierwszym wierszem kodu strony:

```
Option Explicit
```

Następnie tworzona jest zmienna `Content Rotator`:

```
Dim objCR
```

i zostaje zainicjalizowana:

```
Set objCR = Server.CreateObject("IISSample.ContentRotator")
```

Kolejny blok kodowy pojawia się w tym miejscu strony, w którym ma wystąpić cytat, jest więc osadzony wraz z HTML-em:

```
<%
If IsEmpty(Request.QueryString("Show")) Then
  Response.Write objCR.ChooseContent("cr.txt")
Else
  objCR.GetAllContent("cr.txt")
End If
%>
```

Dwa łącza u dołu strony prowadzą do tej samej strony w celu wyświetlenia kolejnego cytatu. Różnią się jednak pomiędzy sobą. Kiedy gość kliknie łącze wyświetlające na stronie wszystkie cytaty, parametr przekazywany jest poprzez `QueryString`, dlatego wstępnie wyszukujemy następujące pole w zbiorze:

```
If IsEmpty(Request.QueryString("Show")) Then
```

Jeśli pole jest puste, należy wyświetlić jeden cytat:

```
Response.Write objCR.ChooseContent("cr.txt")
```

W innym wypadku używamy metody `GetAllContent` w celu wpisania do przeglądarki wszystkich cytatów. Zauważ, że nie realizujemy tego zadania przy użyciu metody `Write`. Nie ma takiej potrzeby, ponieważ metoda `GetAllContent` sama wysyła treść wprost do przeglądarki:

```
Else
  objCR.GetAllContent("cr.txt")
End If
%>
```

Składnik MyInfo

We wcześniejszej części rozdziału przyjrzeliśmy się składnikowi `Counters`. Powiedzieliśmy o jego ogromnym zasięgu, obejmującym wszystkie sesje oraz wszystkie aplikacje ASP Twojego serwera. Wadą tego składnika było jednak to, że zapamiętywał on tylko liczby. Chociaż składnik `MyInfo` nie dysponuje metodami `Increment` i `Remove` tak jak składnik `Counters`, to jednak pozwala na przechowywanie ciągów tekstowych dostępnych w obrębie serwera. Wartości te nie są tracone nawet po wyłączeniu serwera, ponieważ są przechowywane w pliku poza IIS.

Składnik MyInfo złożony jest z właściwości, które tworzysz poprzez ustalenie ich wartości. Właściwości te są następnie wyszukiwane przez określenie ich nazwy. Aby stworzyć składnik, użyj następującego wiersza kodu:

```
Set MyInfo = Server.CreateObject("MSWC.MyInfo")
```

Do utworzenia składnika używamy metody `CreateObject` obiektu `Server`. Zmienna `MyInfo` staje się instancją składnika o tej samej nazwie. Kiedy składnik jest już utworzony, możesz zacząć tworzyć i wykorzystywać wybrane przez siebie właściwości.

Na przykład:

```
<%
Option Explicit
Dim MyInfo
Set MyInfo = Server.CreateObject("MSWC.MyInfo")
MyInfo.CompanyName = "Firma ABC"
Response.Write "<B>" & MyInfo.CompanyName & "<P>"
%>
```

Na wstępie stwierdzamy, że będziemy deklarować nasze zmienne:

```
Option Explicit
```

Następnie deklarujemy zmienną `MyInfo`:

```
Dim MyInfo
```

i przypisujemy jej obiekt:

```
Set MyInfo = Server.CreateObject("MSWC.MyInfo")
```

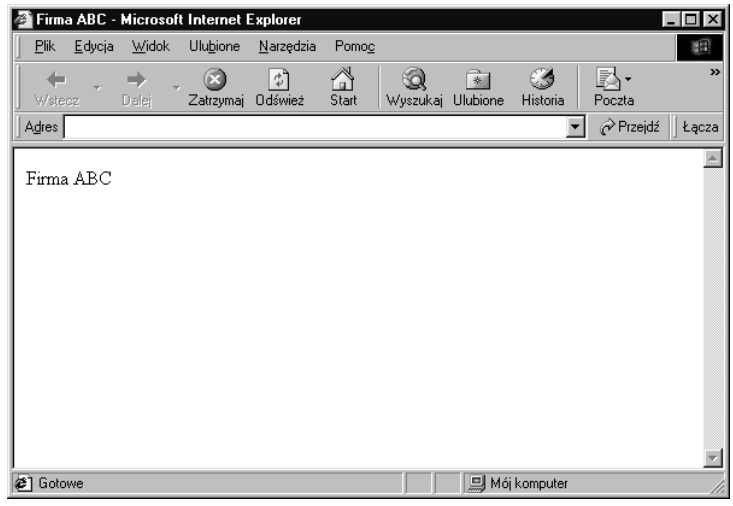
Następnie tworzymy właściwość tego składnika o nazwie `CompanyName` i w tym samym wierszu kodu ustawiamy jej wartość. Jeśli właściwość o tej nazwie już istnieje, jej wartość zostanie zastąpiona:

```
MyInfo.CompanyName = "Firma ABC"
```

Wartość właściwości wpisywana jest do przeglądarki:

```
Response.Write "<B>" & MyInfo.CompanyName & "<P>"
```

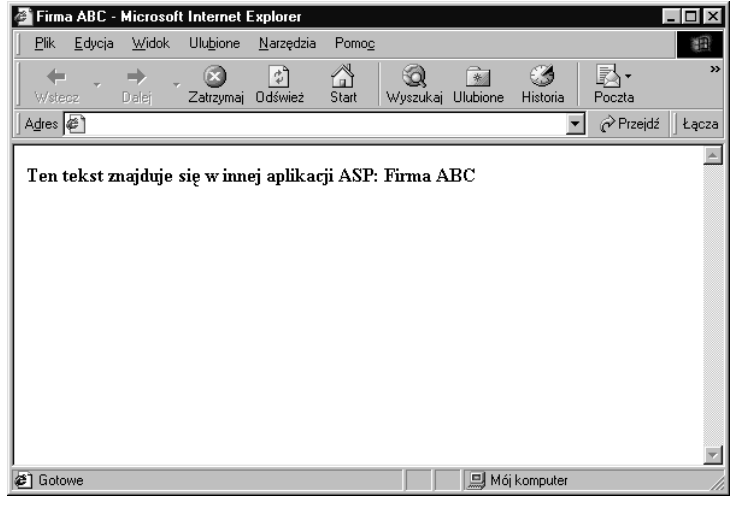
Rezultat działania tego kodu pokazano na rysunku 10.21. Pamiętaj jednak, że ta właściwość ma ogromny zasięg, możemy więc stworzyć również taką stronę, jaką pokazano poniżej, która znajduje się na naszym serwerze w zupełnie innej aplikacji ASP.



Rysunek 10.21. *Wyjgląd pierwszej strony używającej składnika MyInfo*

```
<%
Option Explicit
Dim MyInfo
Set MyInfo = Server.CreateObject("MSWC.MyInfo")
Response.Write "<B>Ten tekst znajduje się w innej aplikacji ASP:" _
   & MyInfo.CompanyName & "<P>"
%>
```

Tutaj właściwość wpisuje do przeglądarki tekst z innej aplikacji ASP, aby zademonstrować swój zasięg. Rezultat działania kodu prezentuje rysunek 10.22.



Rysunek 10.22. *Rezultat działania kodu wykorzystującego właściwość CompanyName strony znajdującej się w innej aplikacji ASP*