Automatyzacja zadań za pomocą Hosta skryptów systemu Windows

Host skryptów systemu Windows (WSH) umożliwia wykonywanie bardziej skomplikowanych zadań niż programy wsadowe. Dzięki niemu możesz kontrolować niemal każdy element Microsoft Windows oraz wielu programów Windows.

Aby uruchomić skrypt, wystarczy wpisać w wierszu polecenia nazwę skryptu lub dwukrotnie kliknąć ikonę skryptu w oknie Eksploratora Windows. Host skryptów systemu Windows to w gruncie rzeczy dwa niemal identyczne programy – Wscript.exe and Cscript.exe – które za pomocą billioteki DLL (np. Vbscript.dll) umożliwiającej interpretację języka uruchamiają skrypty napisane w języku VBScript lub w innych językach skryptowych.

Dzięki WSH pliki mogą być zapisane w kilku różnych językach, w tym VBScript (odmiana języka Microsoft Visual Basic) oraz JScript (odmiana JavaScriptu). Host skryptów systemu Windows jest w rzeczywistości tym, co mówi jego nazwa: hostem (gospodarzem) dla języków skryptowych. Interpretery VBScript i JScript są częścią Windows XP, interpretery Perla oraz innych języków są dostępne oddzielnie.

Skrypty WSH zapewniają dużą elastyczność, ponieważ mogą korzystać z technologii ActiveX. Host skryptów systemem Windows oferuje kilka obiektów, które pozwalają na sterowanie, na poziomie podstawowym, systemem Windows oraz komputerem. Za pomocą formantów ActiveX możesz kontrolować wiele różnych programów. Na przykład możesz utworzyć skrypt WSH, który będzie wyświetlał wykres w programie Microsoft Excel. Na początek przedstawiamy prosty skrypt WSH "Hello World". Jest to najkrótszy skrypt z możliwych i można go napisać w dowolnym języku programowania:

```
WScript.Echo "Hello World"
```

Otwórz dowolny edytor czystego tekstu, na przykład Notatnik, i wpisz ten tekst (zamiast słów „Hello World" możesz wpisać dowolny komunikat, który będzie wyświetlany na ekranie). Plik zapisz z rozszerzeniem .vbs (na przykład Hello.vbs) – masz już gotowy skrypt! Aby go uruchomić, wystarczy dwukrotnie kliknąć ikonę skryptu w Eksploratorze Windows.

Zasoby dotyczące pisania skryptów

Nie licz na to, że nauczymy cię w tym rozdziale pisać skrypty. Musisz najpierw poznać język skryptowy, w czym pomogą ci zasoby opisane w tym podrozdziale. (Jeśli znasz język Visual Basic, to znaczy, że znasz już język skryptowy). Naszym celem jest jedynie pokazanie ci możliwości Hosta skryptów systemu Windows i pomoc w znalezieniu informacji, które umożliwią efektywne korzystanie z niego.

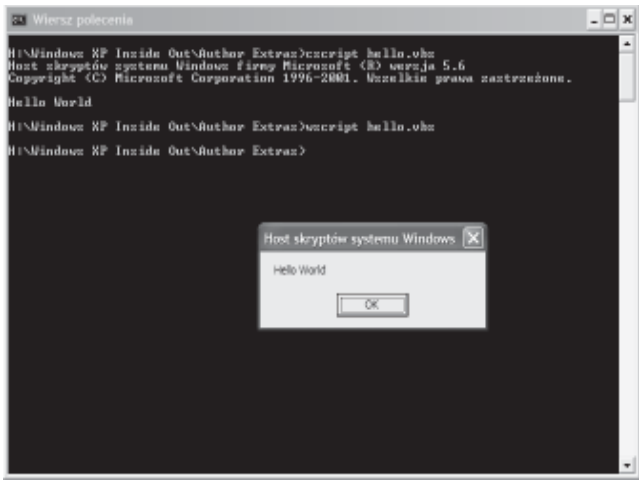
Jedną z największych przeszkód w używaniu WSH jest trudność znalezienia potrzebnych informacji. Język skryptowy, zarówno VBScript, jak i JScript, nie jest tym samym co obiekty używane w skryptach i do każdego z nich dostępna jest oddzielna dokumentacja. Musisz więc znaleźć podręczniki do obu języków skryptowych oraz do używanych obiektów. Z tego rozdziału dowiesz się, gdzie znaleźć odpowiednią dokumentację. Większość materiałów można znaleźć w witrynie firmy Microsoft. Udo-

Wscript i Cscript

Windows XP zawiera dwa programy uruchamiające skrypty WSH. Cscript.exe to wersja dla wiersza polecenia, a Wscript.exe to wersja graficzna. Wbrew pozorom różnica między tymi programami jest naprawdę niewielka. Spróbuj uruchomić jednolinijkowy skrypt Hello.vbs w obu tych programach. W wierszu polecenia wpisz te dwie linie:

```
cscript hello.vbs
wscript hello.vbs
```

Na rysunku 10-6 widoczne są rezultaty. Cscript wyświetlił słowa „Hello World" w oknie Wiersza polecenia. Wscript wyświetla okno dialogowe z komunikatem „Hello World" oraz przyciskiem OK. W wypadku Cscript musisz użyć parametrów wiersza polecenia, aby zmienić właściwości skryptu. Natomiast w wypadku Wscript możesz to zrobić w oknie dialogowym. Aby je wyświetlić, wpisz **wscript** w wierszu polecenia.



Rysunek 10-6. Cscript.exe, wersja dla konsoli, wyświetla rezultaty w oknie Wiersza polecenia, natomiast Wscript.exe w oknie dialogowym.

Zwróć uwagę, że opcje wiersza polecenia dla Cscript i Wscript wymagają dwóch ukośników. Różni się to od opcji wykonywania skrytów, które wymagają tylko jednego ukośnika. Aby przejrzeć wszystkie dostępne opcje wiersza polecenia, wpisz **cscript //?** (lub **wscript //?**) w wierszu polecenia.

stępnia ona nawet osobną witrynę poświęconą zagadnieniom związanym ze skryptami: *http://msdn.microsoft.com/scripting*. Jednak korzystanie z tej witryny nie zawsze jest łatwe, ponieważ obok opisu języków oraz związanych z nim obiektów zawiera także informacje dla programistów, którzy chcą wprowadzić obsługę skryptów do swoich programów – a to już jest zupełnie inny temat.

Skrypty i bezpieczeństwo

Host skryptów systemu Windows wielu osobom kojarzy się nieodmiennie z ryzykiem i brakiem poczucia bezpieczeństwa. Elastyczność i siła tego narzędzia może być wykorzystana w złych zamiarach równie łatwo, jak w dobrych. Na przykład znane robaki I Love You i Anna Kournikova były w rzeczywistości załącznikami w języku VBScript.

Na szczęście możesz wprowadzić kilka prostych zmian, które zmniejszą ryzyko przypadkowego uruchomienia niewłaściwego skryptu.

Najpierw należy upewnić się, że rozszerzenia skryptów zawsze są wyświetlane. (Znajomość rozszerzenia daje cenną wskazówkę – wiele osób nie otworzyłoby lekkomyślnie załącznika o nazwie Anna Kournikova.jpg.vbs. Ponieważ rozszerzenia nie są domyślnie wyświetlane, wielu fanów tej tenisistki spodziewało się ujrzeć jej zdjęcie). Po drugie, należy zmienić domyślną akcję dla skryptów z Otwórz na Edytuj. Jeśli dwukrotnie klikniesz skrypt, zostanie on domyślnie (i bezpiecznie) otworzony w Notatniku. Aby wprowadzić tę zmianę, wykonaj następujące czynności:

1. W Eksploratorze Windows wybierz Narzędzia, Opcje folderów.
2. Wybierz zakładkę Typy plików.
3. Wybierz typ pliku JS (JScript Script File) i kliknij przycisk Zaawansowane.
4. Zaznacz pole wyboru Zawsze pokazuj rozszerzenia.
5. Z listy Akcje wybierz Edytuj i kliknij przycisk Ustaw domyślną. Kliknij przycisk OK.
6. Powtórz kroki 3 -5 dla typów plików JSE (JScript Encoded Script File), VBE (VBScript Encoded Script File), VBS (VBScript Script File) oraz WSF (Windows Script File).
7. Gdy zabezpieczysz wszystkie typy plików, kliknij przycisk Zamknij.

Zmiana domyślnej akcji na Edytuj utrudnia uruchamianie potencjalnie niebezpiecznych skryptów przesyłanych w załącznikach. Jednak z drugiej strony utrudnia to także uruchamianie skryptów lokalnych, a także oczekiwanych i bezpiecznych załączników: musisz najpierw zapisać załącznik, a następnie w Eksploratorze Windows kliknąć skrypt prawym przyciskiem myszy i wybrać z menu polecenie Otwórz. Dobrym rozwiązaniem jest utworzenie skrótu do bezpiecznego skryptu i uruchamianie go za jego pomocą. (Pamiętaj, aby w polu Element docelowy okna dialogowego właściwości skrótu nazwę skrótu poprzedzić poleceniem *wscript.exe* lub *cscript.exe*; jeśli podasz tylko nazwę skryptu, ten trik nie zadziała). Dwukrotne kliknięcie skrótu do skryptu uruchomi go.

Wybór języka

Dla Hosta skryptów systemu Windows nie ma znaczenia, czy korzystasz z języka VBScript, JScript, lub innego języka skryptowego. Wszystkie obiekty dostępne są we wszystkich językach i w większości wypadków możesz wybrać ten język, który znasz najlepiej. Przykłady w tej książce są napisane w języku VBScript. Możliwe jest także używanie różnych języków w jednym programie.

VBScript jest w rzeczywistości okrojoną wersją Visual Basic. Jeśli znasz język Visual Basic, to z pewnością będziesz potrafił zrobić niemal wszystko w VBScript. Jedną z największych różnic jest to, że VBScript ma tylko zmienne wariantowe. Musisz po prostu użyć instrukcji Dim i nazwy zmiennej; nie możesz podać typu zmiennej, ponieważ wszystkie są takie same. Dokumentację języka VBScript znajdziesz pod adresem *http://msdn.microsoft.com/scripting/vbscript/techinfo/vbsdocs.htm*.

Oto krótki przykładowy skrypt o nazwie Folders.vbs, który ilustruje niektóre elementy skryptu napisanego w języku VBScript:

```
Option Explicit

Dim objWSShell
Dim strMsg
Dim intCtr

Set objWSShell = WScript.CreateObject("WScript.Shell")
WScript.Echo "Your desktop is " & objWSShell.SpecialFolders("Desktop") _
  & vbNewLine

strMsg = "All your special folders are:" & vbNewLine
For intCtr = 0 To objWSShell.SpecialFolders.Count - 1
strMsg = strMsg & objWSShell.SpecialFolders.Item(intCtr) & vbNewLine
Next
WScript.Echo strMsg
```

Plik Folders.vbs oraz inne przykładowe skrypty opisane w tym rozdziale znajdziesz na płycie CD dołączonej do tej książki.

Skrypt zaczyna się od instrukcji Option Explicit, która nakazuje interpreterowi języka VBScript, by wymagał od użytkownika użycia instrukcji Dim dla każdej zmiennej w skrypcie. Pomaga to zapobiec błędom wynikającym z błędnie napisanych nazw zmiennych i uważane jest za objaw dobrej znajomości warsztatu programistycznego.

Instrukcja Set tworzy obiekt WScript Shell, który umożliwia uzyskanie dostępu do folderów specjalnych. Jedną z właściwości obiektu Shell jest obiekt SpecialFolders. Obiekt SpecialFolders pozwala pobrać ścieżkę i nazwę pliku każdego folderu specjalnego znajdującego się w twoim systemie. Na przykład ten skrypt zwraca informację o folderze Pulpit (patrz rysunek 10-7).

Rysunek 10-7. Skrypt *Folders.vbs* wyświetla początkowo lokalizację folderu Pulpit.

Kolejna sekcja tego folderu, pętla For...Next, wyświetla wszystkie foldery specjalne. Zademonstrowana tu technika pozwala na właściwe działanie zarówno z Wscript, jak i Cscript. Zamiast wstawiać instrukcję WScript.Echo w pętli, dodaliśmy każdy wynik do ciągu strMsg i wyświetliliśmy ten ciąg po zakończeniu pętli. W wypadku uruchomienia za pomocą programu Cscript rezultaty są te same. Jednak jeśli skrypt zostanie uruchomiony za pomocą programuWscript, zebranie wyników w jeden ciąg oznacza, iż wyświetlone zostanie tylko jedno okno dialogowe dla całej pętli (patrz rysunek 10-8).

Rysunek 10-8. Pętla *For...Next* buduje ciąg, który zostanie wyświetlony w jednym oknie dialogowym.

Używanie formatu plików skryptu

W przypadku skryptów WSH możesz użyć programów w języku VBScript w plikach z rozszerzeniem .vbs i w języku JScript w plikach z rozszerzeniem .js. Host skryptów systemu Windows dodaje jeszcze jeden poziom znaczników, który zapewnia większą elastyczność i siłę. W rzeczywistości pliki WSH z rozszerzeniami .wsf są plikami języka Extensible Markup Language (XML), w których użyto znaczników, co widać w następującym przykładzie (Hello.wsf):

```
<?XML version="1.0"?>
<package>
<job id="job1">
<?job debug="true"?>
<script language="VBScript" src="MyScript.vbs"/>
<script language="VBScript">
<![CDATA[
  WScript.Echo "Hello World"
]]>
</script>
</job>
</package>
```

W tabeli 10-3 zebrano opisy najważniejszych znaczników.

Tabela 10-3. Przydatne znaczniki języka XML

Znacznik	Opis
<?XML version="1.0"?>	Oznacza kod jako zgodny ze specyfikacją XML 1.0. Obecnie znacznik ten jest opcjonalny, ale w przyszłości może być wymagany przez narzędzia XML.
<package> </package>	Umieszcza kilka zadań w jednym pliku. Znacznik <package> jest opcjonalny, jeśli masz tylko jedną parę znaczników <job>.
<job id="job1"> </job>	Identyfikuje zadania w pliku. Jeśli plik zawiera kilka zadań, możesz uruchomić jedno z tych zadań za pomocą składni: Cscript //Job:moje_pierwsze_zadanie Mój skrypt.wsf
<?job debug="true"?>	Pozwala na użycie debugera skryptów. Możesz dodać *error="true"* do tego znacznika, aby pozwolić na wyświetlanie komunikatów o błędach składni lub wykonania.
<script language= "VBScript" src= "MyScript.vbs"/>	Uwzględnia lub łączy inny plik z bieżącym. Znacznik ten pozwala na wielokrotne zastosowanie tego samego kodu.

Znacznik	Opis
<script language= "VBScript"> </script>	Otwiera i zamyka skrypt. W pojedynczym zadaniu możesz mieć kilka skryptów – nawet w różnych językach.
<![CDATA[ ]]>	Wskazuje, że parser powinien potraktować kod jako znak i nie interpretować znaków w kodzie. Użyj tego znacznika, jeśli korzystasz ze znacznika XML.
<object>	Definiuje obiekty, do których skrypt może się odwoływać.
<reference>	Zapewnia odwołanie do zewnętrznej biblioteki, pozwalając na użycie zdefiniowanych stałych z bibliotek tego typu.
<resource>	Izoluje tekst lub dane numeryczne, które nie powinny być zakodowane literalnie w skrypcie.

Debugowanie skryptów

Aby móc debugować skrypty, musisz najpierw zainstalować program Microsoft Script Debugger. Możesz go pobrać ze strony *http://msdn.microsoft.com/downloads/default.asp?URL=/downloads/sample.asp?url=/msdn-files/027/001/731/msdncompositedoc.xml*.

Jeśli chcesz zdebugować plik .wsf, musisz dodać do pliku skryptu linię *<?job debug="true"?>*. (Bezpośrednio pod znacznikiem <job>). Bez tej linii debuger nie zostanie otwarty.

UWAGA
Jeśli po zainstalowaniu debugera, podczas przeglądania strony internetowej natrafisz na błąd skryptu, wyświetlony zostanie inny niż dotychczas komunikat. W dolnej części okna dialogowego, pod pytaniem „Czy chcesz debugować?", znajdują się dwa przyciski. Jeśli klikniesz przycisk Tak, pojawi się okno debugera. Jednak ponieważ najprawdopodobniej przeglądasz cudzą stronę, należy kliknąć Nie!

Dokumentację programu Script Debugger możesz znaleźć pod adresem *http://msdn.microsoft.com/downloads/default.asp?URL=/downloads/sample.asp?url=/msdn-files/027/001/731/msdncompositedoc.xml*. Pamiętaj, że możesz użyć tego samego debugera do debugowania skryptów klienta i serwera stron sieci Web; zdecydowana większość informacji w dokumentacji dotyczy właśnie tego. (Jednak Host skryptów systemu Windows może nawet nie zostać wspomniany).

Debuger uruchamiany jest za pomocą przełączników poleceń Cscript lub Wscript. Przełącznik //X uruchamia debuger, ładuje do niego skrypt i zatrzymuje przy pierwszej linii skryptu. Przełącznik //D uruchamia debuger, tylko gdy napotkany zostanie błąd. On również powoduje załadowanie programu do debugera, ale zatrzymuje go w wierszu, w którym znajduje się błąd.

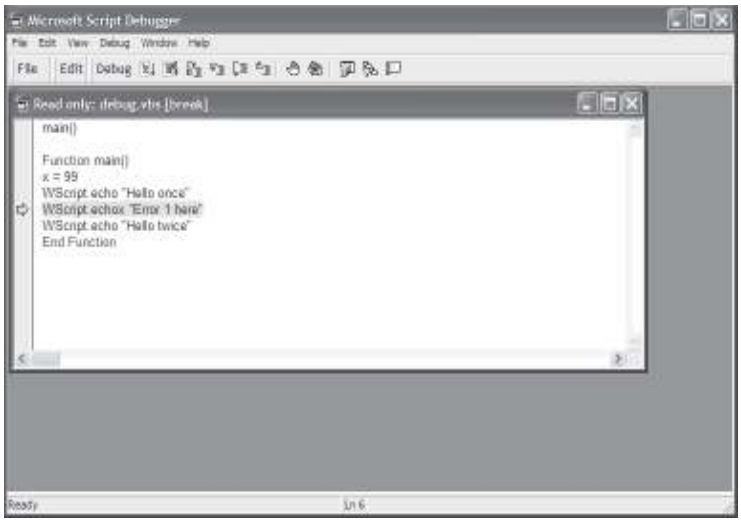
Oto przykład debugowania bardzo prostego skryptu o nazwie Debug.vbs:

```
main()
Function main()
x = 99
WScript.echo "Hello once"
WScript.echox "Error 1 here"
WScript.echo "Hello twice"
End Function
```

Po utworzeniu takiego skryptu wpisz w wierszu polecenia:

```
wscript //d debug.vbs
```

Słowa „Hello once" powinny pojawić się w małym oknie dialogowym. Gdy klikniesz przycisk OK, pojawi się okno programu Script Debugger, w którym wyróżniony będzie błąd (patrz rysunek 10-9).



Rysunek 10-9. Script Debugger wyświetli błąd, gdy uruchomisz skrypt z przełącznikiem //D.

Debuger pomaga w odszukaniu błędu, ale nie pozwala na jego naprawienie. Jak widać w górnej części okna dokumentu, plik otwierany jest w trybie tylko do odczytu. Możesz otworzyć inne wystąpienie pliku w debugerze i dokonać jego edycji. Debuger zapewnia edytor podobny funkcjonalnie do Notatnika.

Inne okna debugera

Debuger oferuje także dodatkowe informacje, jednak jeśli udało ci się poprawić skrypt i nie ma w nim innych błędów, nie musisz z nich korzystać. Trzy dodatkowe okna, wszystkie dostępne z menu View oraz z paska narzędzi, pozwalają na przejrzenie dodatkowych informacji o skrypcie:

- **Running Documents**. To okno pokazuje wszystkie działające obecnie skrypty. Oprócz Hosta skryptów systemu Windows możesz zobaczyć tu również przeglądarkę Microsoft Internet Explorer. Jeśli klikniesz prawym przyciskiem myszy skrypt, w menu skrótów pojawi się jedno polecenie: Break At Next Statement. Jest to sposób na „zaproszenie" skryptu do debugera.

- **Call Stack**. To okno wyświetla historię wywołań do bieżącego punktu bieżącego skryptu. Może ci to pomóc dowiedzieć się, w jaki sposób dostałeś się do tego miejsca.

- **Command**. To okno pozwala przeglądać i manipulować zmiennymi w skrypcie. W tym przykładzie pierwsza linijka „wydrukowała" wartość *x*. (Znak zapytania jest skrótem polecenia Print języka Visual Basic). Następnie wartość *x* została zmieniona na 100 i wyświetlona ponownie.

Pracując w oknie Command, musisz używać tego samego języka, co uruchomiony w danym momencie skrypt. Jeśli masz na przykład uruchomiony skrypt w języku VBScript, wpisz znak zapytania, wstaw spację i podaj nazwę zmiennej, aby wyświetlić jej wartość. Jeśli uruchomiłeś plik w języku JScript, wystarczy wpisać nazwę zmiennej. W oknie Command możesz także przeglądać i zmieniać właściwości obiektu.

Debugowanie krokowe

Gdy uruchomisz debuger za pomocą przełącznika wiersza polecenia //X, w oknie debugera wyróżniona zostanie pierwsza linijka z błędem. Możesz przejść krokowo przez cały skrypt i zobaczyć, jak działa pozostała część. Wszystkie informacje o debugowaniu są dostępne, możesz więc sprawdzić wartość zmiennych i właściwości obiektów, a także zobaczyć, jak są zmienianie w trakcie działania skryptu.

Pasek narzędzi Debug – jeden z trzech w oknie programu Script Debugger – oferuje narzędzia, które pozwalają na krokowe przejście skryptu. W tabeli 10-4 zebrano opis przycisków dostępnych na pasku narzędzi Debug.

Proces debugowania polega na kolejnym wykonaniu poszczególnych linijek, aż do natrafienia na błąd. Gdy upewnisz się, że w danej części skryptu nie ma żadnego błędu, możesz oznaczyć miejsce (punkt wstrzymania), od którego należy zacząć dalsze sprawdzenie. Kliknięcie przycisku Run spowoduje szybkie przejście do tego punktu.

Tabela 10-4. Pasek Debug programu Script Debugger

Przycisk	Nazwa	Opis
	Run	Wykonuje skrypt aż do punku wstrzymania, błędu lub końca skryptu.
	Stop Debugging	Uruchamia skrypt poza debugerem.
	Break At Next Statement	Aktywuje otwarty skrypt serwera w debugerze; nie używane dla plików WSH.
	Step Into	Przechodzi do następnej instrukcji; wchodzi w funkcje i podprocedury.
	Step Over	Przechodzi do następnej instrukcji; wykonuje, ale nie wyświetla funkcji i podprocedur.
	Step Out	Przechodzi dalej, aż do końca bieżącej funkcji lub podprocedury.
	Toggle Breakpoint	Wstawia lub usuwa punkt wstrzymania w lub z linii z punktem wstawienia.
	Clear All Breakpoints	Usuwa wszystkie punkty wstrzymania.
	Running Documents	Wyświetla okno dialogowe Running Documents.
	Call Stack	Wyświetla okno dialogowe Call Stack.
	Command Window	Wyświetla okno dialogowe Command.

Kiedy zyskasz pewność, że twoje funkcje i podprogramy działają właściwie, możesz przyspieszyć debugowanie, klikając przycisk Step Over, aby wykonać je w jednym kroku, zamiast używać przycisku Step Into.

Wprowadzanie obiektów

Bez obiektów nie możesz zrobić zbyt wiele w WSH. *Obiektem* nazywamy zmienną składającą się zarówno z procedur, jak i danych, która traktowana jest jako całość. Niektóre obiekty są wbudowane w język skryptowy; inne są udostępniane przez system operacyjny. W jednym z zaprezentowanych już skryptów użyliśmy obiektu WScript.Shell w celu uzyskania dostępu do powłoki Windows.

Tabela 10-5 zawiera opis obiektów będących częścią języka VBScript. Obiekty te są opisane w dokumentacji tego języka, którą można znaleźć pod adresem *http://msdn.microsoft.com/scripting/vbscript/techinfo/vbsdocs.htm*.

Tabela 10-5. Obiekty VBScript

Obiekt	Opis
Class	Zapewnia dostęp do zdarzeń utworzonej klasy.
Dictionary	Przechowuje dane w parach klucz/element.
Drive	Zapewnia dostęp do właściwości dysku lokalnego lub dysku udostępnionego w sieci.
Drives - kolekcja	Zbiór wszystkich dostępnych obiektów Drive.
Err	Zapewnia informacje o błędach wykonania.
File	Zapewnia dostęp do wszystkich właściwości pliku.
Files - kolekcja	Zbiór wszystkich obiektów File w obrębie folderu.
FileSystemObject	Zapewnia dostęp do systemu plików komputera.
Folder	Zapewnia dostęp do wszystkich właściwości folderu.
Folders - kolekcja	Zbiór wszystkich obiektów Folder zawartych w obiekcie Folder.
Match	Zapewnia dostęp do właściwości przetwarzania regularnych wyrażeń przeznaczonych tylko do odczytu.
Matches - kolekcja	Kolekcja obiektów Match regularnych wyrażeń.
RegExp	Zapewnia prostą obsługę regularnych wyrażeń.
TextStream	Umożliwia sekwencyjny dostęp do pliku.

Host skryptów systemu Windows zawiera obiekty przedstawione w tabeli 10-6. Są one udokumentowane, wraz z Hostem skryptów systemu Windows, pod adresem *http://msdn.microsoft.com/scripting/*. Na tej stronie kliknij Windows Script Host, Documentation, Reference.

Windows Management Instrumentation (WMI) oferuje znacznie więcej obiektów, które możesz wykorzystać do zarządzania środowiskiem Windows. Dokumentacja WMI znajduje się pod adresem *http://msdn.microsoft.com/library/en-us/dnw2k/html/wmioverview.asp*. Kliknij przycisk „toc" znajdujący się u góry tej strony, aby włączyć panel spisu treści.

Tabela 10-6. Obiekty Hosta skryptów systemu Windows

Obiekt	Opis
WScript	Udostępnia właściwości, które określają ścieżkę bieżącego hosta skryptów (Wscript.exe lub Cscript.exe), jego argumenty oraz tryb pracy (interaktywny lub wsadowy); zapewnia także metody tworzenia i czytania obiektów.
WshArguments	Umożliwa dostęp do zbioru argumentów wiersza polecenia.
WshEnvironment	Pobiera zmienne środowiskowe.
WshNetwork	Mapuje sieć, ułatwiając łączenie i rozłączanie zdalnych dysków i drukarek.
WshShell	Rozpoczyna nowe procesy, tworzy skróty i zapewnia kolekcję Environment, umożliwiającą obsługę zmiennych środowiskowych takich jak SystemRoot, Path i Prompt.
WshShortcut	Tworzy obiektową referencję do skrótu.
WshSpecialFolders	Daje dostęp do folderów systemowych Windows takich jak folder Pulpit, Meni Start i Moje dokumenty.
WshUrlShortcut	Tworzy obiektową referencję do skrótu URL.

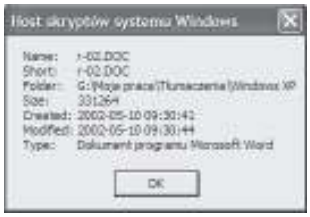
Używanie FileSystemObject

Jednym z obiektów, których prawdopodobnie będziesz często używał, jest FileSystemObject. Obiekt ten daje dostęp do plików i folderów w twoim komputerze. Poniższy przykład, FileProp.vbs, prezentuje sposób użycia tego obiektu, natomiast na rysunku 10-10 widoczny jest rezultat.

```
' Wyświetla właściwości pliku w wierszu poleceń

  Option Explicit
  Dim strArg, objFileSys, objFile, strMsg

If WScript.Arguments.Count < 1 Then
    WScript.Echo "Usage: FileProp <filename>" & vbNewLine & _
      "Or drag and drop a file on this file."
    WScript.Quit (1)
  End If

  Set objFileSys = CreateObject("Scripting.FileSystemObject")

  strArg = objFileSys.GetAbsolutePathName(WScript.Arguments(0))

  Set objFile = objFileSys.GetFile(strArg)
  strMsg = "Name: " & vbTab & objFile.Name & vbNewLine
  strMsg = strMsg & "Short: " & vbTab
  strMsg = strMsg & objFile.ShortName & vbNewLine
  strMsg = strMsg & "Folder: " & vbTab
  strMsg = strMsg & objFile.ParentFolder & vbNewLine
  strMsg = strMsg & "Size: " & vbTab
  strMsg = strMsg & objFile.Size & vbNewLine
  strMsg = strMsg & "Created: " & vbTab
  strMsg = strMsg & objFile.DateCreated & vbNewLine
  strMsg = strMsg & "Modified:" & vbTab
  strMsg = strMsg & objFile.DateLastModified & vbNewLine
  strMsg = strMsg & "Type: " & vbTab
  strMsg = strMsg & objFile.Type & vbNewLine
  WScript.Echo strMsg
  Set objFile = Nothing
  Set objFileSys = Nothing
```

Ten skrypt sprawdza najpierw, czy wiersz polecenia zawiera jakieś parametry. Jeśli nie znajdzie żadnych parametrów, wyświetla komunikat i kończy działanie. Skrypt współdziała z technologią „przeciągnij i upuść", ponieważ Windows dodaje nazwę upuszczonych plików do wiersza poleceń. Jeśli upuścisz więcej niż jeden plik na ten skrypt, wyświetli informacje tylko o pierwszym z nich.

Po sprawdzeniu, czy nie ma żadnych błędów, skrypt tworzy ciąg FileSystemObject i używa go do uzyskania pełnej ścieżki do pliku. Wykorzystując pełną ścieżkę, otrzymuje obiekt File.



Rysunek 10-10. Skrypt FileProp.vbs wyświetla właściwości plików, które nie są łatwo dostępne w Eksplorerze.

Następnie skrypt tworzy ciąg poprzez konkatenację różnych właściwości obiektu File. Gdy cały ciąg jest już gotowy, skrypt wyświetla go metodą Echo obiektu WScript. Do wyświetlenia ciągu możesz także użyć funkcji MsgBox. Pozwala ona na wybór wyświetlanych przycisków, zmianę paska tytułu z Host skryptów systemu Windows na wybrany komunikat oraz odpowiedź na przycisk kliknięty przez użytkownika. Chociaż metoda WScript.Echo działa inaczej w programie Cscript, a inaczej w programie Wscript, funkcja MsgBox zachowuje się w obu tak samo: zawsze wyświetla okno dialogowe. Aby zobaczyć różnicę, zastąp wiersz WScript.Echo następującym:

```
MsgBox strMsg, vbOKOnly, "File Properties"
```

Na końcu skrypt zwalnia obiekt FileSystemObject oraz obiekt File, określając jako wartość zmiennej wbudowaną zmienną Nothing, która zwalnia zasoby używane przez obiekty. W małym skrypcie, takim jak ten, nie jest to takie istotne, ponieważ zakończenie skryptu ma to samo działanie, ale w bardziej złożonych skryptach zwolnienie niepotrzebnego już obiektu pozwala na ograniczenie zużycia zasobów wymaganych przez ten skrypt.

Dokumentacja obiektu FileSystemObject znajduje się pod adresem *http://msdn.microsoft.com/library/default.asp?URL=/library/devprods/vs6/vbasic/vbenlr98/vaobjFileSystemObject.htm*.

Przykładowe skrypty

W tej sekcji prezentujemy dwa przykładowe skrypty demonstrujące niektóre techniki, których możesz używać w Hoście skryptów systemu Windows.

Przeglądanie artykułów bazy Microsoft Knowledge Base

Pierwszy skrypt wyświetla artykuły bazy Microsoft Knowledge Base (MSKB) w przeglądarce Internet Explorer. W wielu miejscach możesz znaleźć odniesienia do artykułów MSKB: książkach, takich jak ta, magazynach oraz w witrynach sieci Web. Odniesienia te łatwo rozpoznać – składają się z litery *Q* oraz sześciocyfrowej liczby, ale odnalezienie potrzebnego artykułu nie jest już takie proste – musisz kliknąć kilka łączy lub wpisać długi adres. Ten prosty skrypt o nazwie Kbquick.vbs otwiera artykuł bazy Microsoft Knowledge Base w przeglądarce Internet Explorer. Skrypt prosi o podanie numeru artykułu (Article ID), tworzy adres URL do artykułu i uruchamia Internet Explorera, otwierając daną stronę.

```
'*************************************************************
'* Plik: KBquick.vbs
'*
'* Function: Otwiera określony artykuł z bazy wiedzy Microsoft Knowledge Base.
'*
'* Użycie: Monituje o identyfikator artykułu, a następnie uruchamia program
'* Internet Explorer w celu wyświetlania artykułu.
'*
'*************************************************************
Option Explicit
Dim shell
Dim strKBID
Dim strURL
Dim strPrompt
Dim strTitle
Dim strDefault
```

```
' Pyta o identyfikator artykułu.

strPrompt = "Enter the Article ID of the Microsoft Knowledge " & _ "Base article."
strTitle = "Knowledge Base Assistant"
strDefault = "Q"
strKBID = InputBox(strPrompt, strTitle, strDefault)
' Jeśli użytkownik kliknie Anuluj lub nic nie wpisze, zakończ działanie.
if strKBID = "" Then
  WScript.Quit
End If

' Sprawdza długość i pierwszy znak.
While len(strKBID) <> 7 or UCase(left(strKBID, 1)) <> "Q
  strPrompt = "Knowledge Base IDs start with the letter Q,
  strPrompt = strPrompt & "which is followed by 6 digits."
  strKBID = InputBox(strPrompt, strTitle, strKBID)
  if strKBID = "" Then
    WScript.Quit
  End If
Wend

' Tworzy adres URL na podstawie identyfikatora artykułu
strURL = "http://support.microsoft.com/support/kb/articles/"
strURL = strURL & left(strKBID, 4) & "/"
strURL = strURL & mid(strKBID, 5, 1) & "/"strURL = strURL & _
  right(strKBID, 2) & ".asp"

' Otwiera artykuł w przeglądarce Internet Explorer
Set shell = CreateObject("WScript.Shell")
shell.Uruchom("IEXPLORE" & " " & strURL)
shell.AppActivate "IExplore"
```

Funkcja InputBox służy do przekazania prośby o numer artykułu. Ponieważ numery artykułów zawsze zaczynają się od litery *Q*, znak ten umieszczany jest domyślnie w polu tekstowym. Funkcja InputBox zwraca numer artykułu do skryptu, który następnie dokonuje prostego sprawdzenia błędów. Skrypt sprawdza, czy numer artykułu ma siedem znaków długości oraz czy zaczyna się od litery *Q*. Jeśli znaleziony zostanie błąd, funkcja InputBox jest wykonywana ponownie, żądając innego numeru artykułu. Pętla While ... Wend powoduje powrót do pytania, aż do momentu, gdy zostanie podany właściwy numer lub użytkownik kliknie przycisk Anuluj.

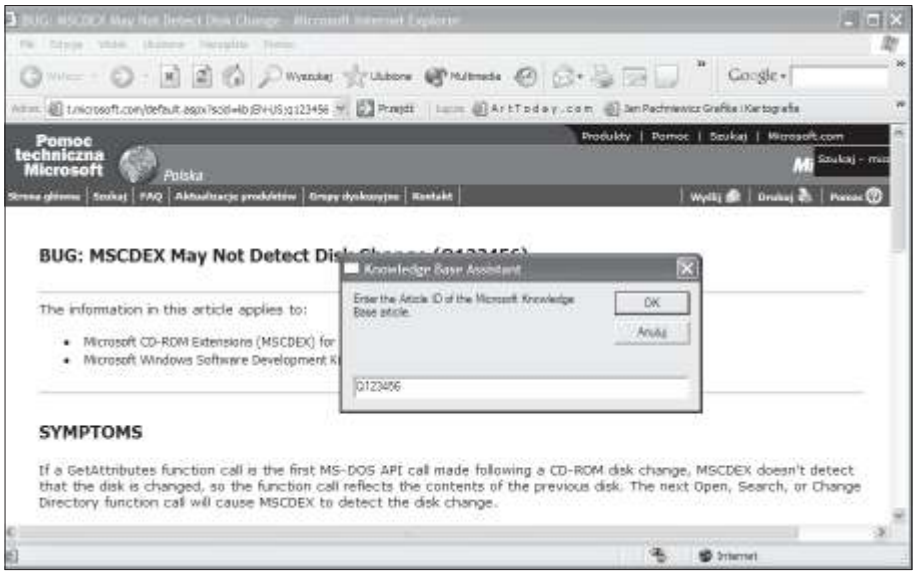
Następnym krokiem jest przeanalizowanie składniowe numeru i utworzenie adresu URL. Na przykład adres do artykułu Q123456 to *http://support.microsoft.com/support/kb/articles/q123/4/56.asp*.

Znając pełny adres URL, skrypt uruchamia przeglądarkę Internet Explorer i otwierany jest artykuł Microsoft Knowledge Base (rysunek 10-11).

WSKAZÓWKA

Użyj innego skrótu do artykułów Microsoft Knowledge Base

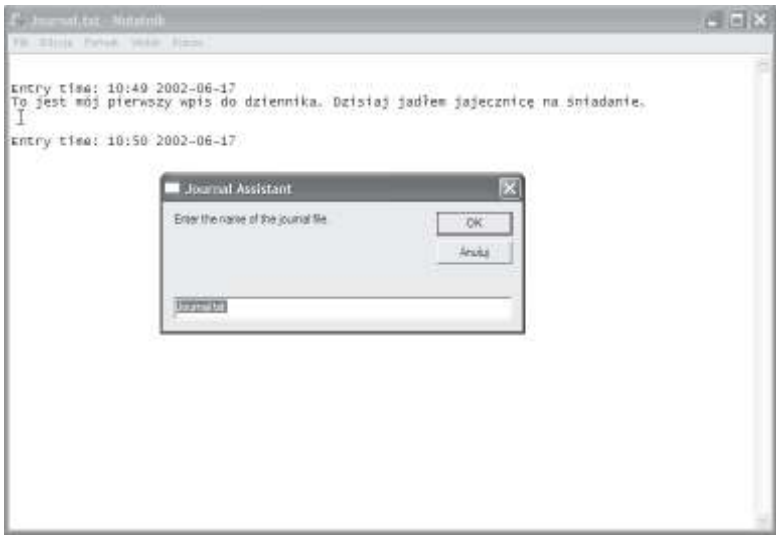
Jeżeli Internet Explorer jest tak skonfigurowany, aby korzystać z MSN do wyszukiwania fraz wpisanych w pasku Adres, stanowi to wbudowaną alternatywę dla zaprezentowanego powyżej skryptu. W pasku Adres przeglądarki Internet Explorer wpisz **mskb**, a następnie oddzielony spacją numer artykułu. Jeśli nie możesz zastosować tego triku, przeczytaj podrozdział „Wyszukiwanie za pomocą paska adresu” na stronie 574.



Rysunek 10-11. Skrypt prosi o numer artykułu MSKB, a następnie wyświetla artykuł w oknie przeglądarki.

Dziennik

Kolejny skrypt, o nazwie Journal.vbs, pomaga w prowadzeniu dziennika. Skrypt otwiera plik tekstowy i dodaje czas (patrz rysunek 10-12). Podobnie jak w innych skryptach, użyliśmy standardowej konwencji nazewniczej, wstawiliśmy procedury sprawdzania błędów i dodaliśmy inne elementy uzupełniające zasadniczą funkcję skryptu. W gruncie rzeczy skrypt jest doskonałym przykładem, w jaki sposób można za jego pomocą wykonać proste zautomatyzowane zadania: wystarczy, że uruchomi program lub odtworzy klawisze.



Rysunek 10-12. Skrypt Journal.vbs otwiera plik w programie Notatnik, a następnie wstawia datę i czas na końcu pliku.

```
'*************************************************************
'*
'* Plik: Journal.vbs
'*
'* Funkcja: Otwiera plik tekstowy i wstawia datownik.
'*
'* Użycie: Parametr z wiersza poleceń jest nazwą otwieranego pliku.
'* Jeśli nie ma parametru, monituje o nazwę pliku.
'*
'*************************************************************

Option Explicit
Dim shell
Dim strFileName
Dim strPrompt
Dim strTitle
Dim strDefault

' Sprawdza parametr z wiersza poleceń.
If WScript.Arguments.Count Then
  ' Używa parametru jako nazwy pliku.
  strFileName = WScript.Arguments(0)
Else
  ' Brak parametru. Pyta o nazwę pliku.
  strPrompt = "Enter the name of the journal file."
  strTitle = "Journal Assistant"
  strDefault = "Journal.txt" ' propose a name
  strFileName = InputBox(strPrompt, strTitle, strDefault)
  ' Jeśli użytkownik kliknie Anuluj lub nic nie wpisze, kończy działanie.
  If strFileName = "" Then
    WScript.Quit
  End If
End If

' Uruchamia Notatnik z podanym plikiem.
Set shell = CreateObject("WScript.Shell")
shell.Uruchom("Notepad " & strFileName)
shell.AppActivate "Notepad"
WScript.Sleep 100 ' Odczekanie na uruchomienie Notatnika

' Przejście do końca pliku i wstawianie datownika.
shell.SendKeys "^{END}{ENTER}{ENTER}" ' przejście do końca pliku
shell.SendKeys "Entry time: " ' dodanie tekstu
shell.SendKeys "{F5}{ENTER}" ' wstawienie datownika
```

Oto cztery sposoby określenia pliku tekstowego, który będzie używany jako dziennik:

- Wpisz nazwę pliku w oknie, które tworzy skrypt.
- Uruchom skrypt z wiersza polecenia, podając nazwę pliku.
- Uruchom skrypt za pomocą skrótu zawierającego nazwę pliku w polu Element docelowy.
- Przeciągnij plik z Eksploratora Windows i upuść go na skrypt lub skrót do skryptu. (Jeśli upuścisz plik na skrót, w którym określiłeś nazwę pliku docelowego, upuszczony plik będzie miał pierwszeństwo i to on zostanie otwarty).

Po uruchomieniu skrypt sprawdza, czy podano argument w wierszu polecenia. Jeśli tak, jest on używany jako nazwa pliku. Jeśli nie, skrypt używa funkcji InputBox, aby zażądać nazwy pliku.

Po podaniu nazwy pliku skrypt uruchamia program Notatnik i otwiera w nim wskazany plik.

Metoda SendKeys wysyła klawisze do programu. W składni SendKeys większość klawiszy reprezentowanych jest zwykłymi znakami. Klawisze, które nie dają znaków, takie jak [Enter] lub [Backspace], są reprezentowane przez specjalne kody, które przedstawiono w tabeli 10-7. Klawisze używane łącznie z innymi znakami (na przykład [Shift]) także są reprezentowane przez kody.

Tabela 10-7. Kody metody SendKeys

Klawisz	Kod
Backspace	{BACKSPACE}, {BS} lub {BKSP}
Break	{BREAK}
Caps Lock	{CAPSLOCK}
Delete	{DELETE} lub {DEL}
Strzałka w dół	{DOWN}
End	{END}
Enter	{ENTER} lub ~
Esc	{ESCAPE} lub {ESC}
Help	{HELP}
Home	{HOME}
Insert	{INSERT} lub {INS}
Strzałka w lewo	{LEFT}
Num Lock	{NUMLOCK}
Page Down	{PGDN}
Page Up	{PGUP}
Print Screen	{PRTSC}
Strzałka w prawo	{RIGHT}
Scroll Lock	{SCROLLLOCK}
Tab	{TAB}
Strzałka w górę	{UP}
Klawisze funkcyjne od F1 do F16	{F1} do {F16}
Shift	+
Ctrl	^
Alt	%