4

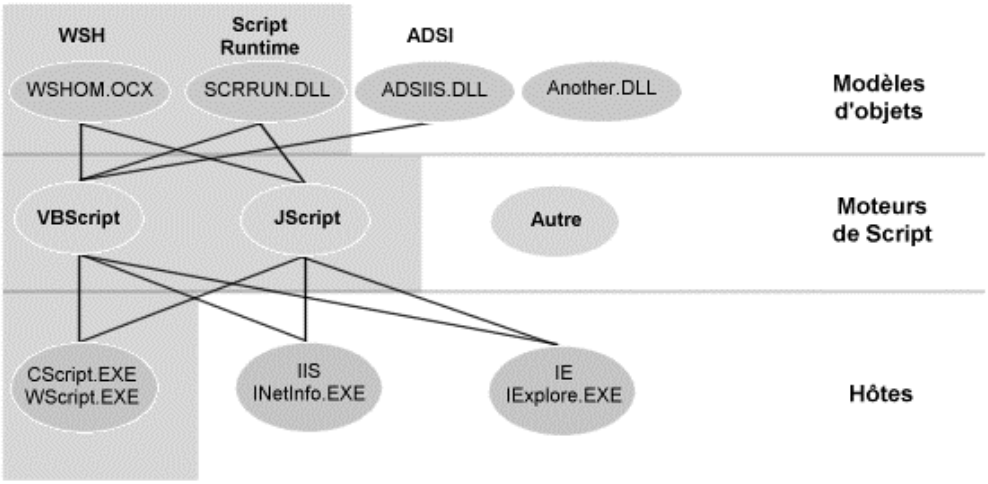
Modèles d'objets de Windows Script Host

4- Modèles d'objets de Windows Script Host

Comme mentionné au premier chapitre, les scripts que vous écrivez utilisent un hôte afin de pouvoir s'exécuter. Cet hôte peut être `CScript.exe` ou `WScript.exe` afin d'exécuter des scripts *Windows* mais pourrait également être `IIS` afin d'exécuter des scripts serveur *(ASP)* ou `Internet Explorer` afin d'exécuter des scripts clients.

Ces différents hôtes permettent aux moteurs de script *VBScript* et *JScript* de lire, d'interpréter et d'exécuter les instructions spécifiés à l'aide du langage adéquat.

Cependant, rien dans ces outils ne permettra de manipuler la base de registre, de connecter des lecteurs réseau, d'effectuer des backup, etc. Cependant, les scripts peuvent demander l'activation de composants exécutables compilés et résidents sur le système. Ces composants sont des fichiers possédant généralement l'extension **.dll* ou **.ocx* et ont été écrits et compilés à l'aide de langages de programmation tels le *C++*, *Delphi* ou *Visual Basic*.



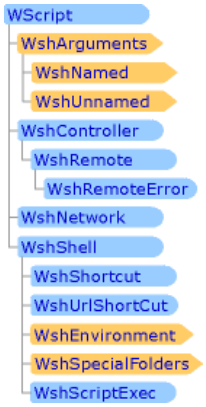
Quoique les scripts ne puissent créer eux-mêmes leurs propres composants, ils peuvent exploiter les fonctionnalités de ceux qui existent déjà. Parmi ceux-ci, nommons les plus courants :

- **WSH** est le modèle de base mis à la disposition de *Windows Script Host* et permet la création de raccourcis, la connexion de lecteurs réseau, l'exécution de commandes, ect.
- **Script Runtime** permet la manipulation des lecteurs, des répertoires et des fichiers.
- **ADSI** permet la manipulation des annuaires *Active Directory* de Windows 2000 et de Windows.NET Server.

Les objets WScript

L'illustration ci-contre représente le modèle d'objets de *Windows Script Host* permettant d'exécuter un grand nombre de tâches administratives.

Le modèle d'objet de *Windows Script Host* est constitué de 14 objets dont l'objet racine est l'objet `WScript` à la base de tous. Chacun de ces objets possèdent des propriétés et des méthodes.



Objet	Ce que permet cet objet
WScript	Retourner et déterminer les différents arguments de la ligne de commande. Déterminer le nom du fichier de script. Déterminer l'hôte de script et la version de l'hôte en exécution. Créer, connecter et déconnecter des objets COM. Stopper l'exécution d'un script. Interagir avec l'utilisateur à l'aide de boîtes de message.
WshArguments	Collection permettant d'accéder à l'ensemble des arguments de la ligne de commande.
WshNamed	Accéder à l'ensemble des arguments nommés de la ligne de commande.
WshUnnamed	Accéder à l'ensemble des arguments non-nommés de la ligne de commande.
WshNetwork	Connecter et déconnecter des partages réseau, des imprimantes partagées. Accéder aux informations au sujet de l'utilisateur en cours.
WshController	Créer un processus afin d'exécuter un script sur un autre ordinateur du réseau.
WshRemote	Administrer un autre ordinateur du réseau et manipuler ses scripts.
WshRemoteError	Accéder au code d'erreur produit par l'exécution d'un script distant.
WshShell	Démarrer un programme. Manipuler la base de registre et les variables d'environnement. Créer des raccourcis. Accéder à un répertoire système.
WshShortcut	Créer un raccourci.
WshUrlShortcut	Créer un raccourci internet.
WshEnvironment	Accéder aux variables d'environnement.
WshSpecialFolders	Accéder aux répertoires spéciaux de Windows.
WshScriptExec	Déterminer l'état ou le code d'erreur à propos d'un script. Accéder aux canaux *StdIn*, *StdOut* et *StdErr*.

L'objet WScript

L'objet `WScript` est l'objet racine du modèle d'objets de *Windows Script Host* et expose les fonctionnalités de base de celui-ci. L'objet `WScript` est un objet intrinsèque au modèle d'objets de *Windows Script Host* et n'a pas besoin d'être instancié avant d'être utilisé. Une instance de cet objet est omniprésente dans tout les scripts que vous créerez.

Propriété	Description	Exemple

Arguments	Retourne la collection d'objets `WshArguments` correspondants à l'ensemble des paramètres passés à la ligne de commande le cas échéant.	`Set Args = WScript.Arguments`
		`For I = 0 to Args.Count - 1`
		`  WScript.Echo Args(I)`
		`Next`

FullName	Retourne le chemin complet de l'hôte exécutant le présent fichier de script. Permet de connaître l'emplacement de l'hôte à partir de lequel le script est exécuté.	`WScript.Echo WScript.FullName`
		
		`' Produit le résultat suivant:`
		`C:\winnt\system32\cscript.exe`

Interactive	Détermine ou retourne le mode de script. Le script affiche les boîtes de message et d'erreur lorsque cette propriété prend la valeur *True* et n'affiche aucune information lorsqu'elle prend la valeur *False*.	`WScript.Interactive = False`
Name	Retourne le nom de l'objet WScript	`WScript.Echo WScript.Name`

Path	Retourne le chemin du répertoire contenant l'hôte exécutant le présent fichier de script.	`WScript.Echo WScript.Path`
		
		`' Produit le résultat suivant:`
		`C:\winnt\system32`

ScriptFullName	Retourne le chemin complet du présent fichier de script. Permet de connaître l'emplacement à partir de lequel le script est exécuté peu importe l'hôte utilisé.	`N = WScript.ScripFullNamePath`
		`WScript.Echo N`
		
		`' Produit le résultat suivant:`
		`C:\scripts\test.vbs`

ScriptName	Retourne le nom du fichier du présent fichier de script. Permet de connaître le nom du fichier à partir de lequel le script est exécuté peu importe l'hôte utilisé.	`N = WScript.ScripFullNamePath`
		`WScript.Echo N`
		
		`' Affiche  test.vbs`

StdErr	Permet l'accès en écriture seulement à la sortie standard pour les messages d'erreur. Utilisable sous l'hôte *CScript.exe* seulement.	
StdIn	Permet la lecture de l'entrée standard. Utilisable sous l'hôte *CScript.exe* seulement.	
StdOut	Permet l'accès en écriture seulement à la sortie standard. Utilisable sous l'hôte *CScript.exe* seulement.	`Z = "Allô la planète"` `WSript.StdOut.Write Z`

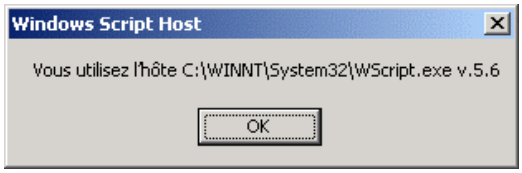
Version	Permet de connaître la version de *Window Script Host* utilisée pour l'exécution du script.	`N = WScript.Version`
		`WScript.Echo "WSH " & N`
		`' Produirait WSH 5.6`

Méthode	Description	Exemple
ConnectObject	Établi un lien avec l'objet spécifié afin d'en gérer les événements.	`Set oIE = CreateObject ▼ ("InternetExplorer.Application") WScript.ConnectObject(oIE, "IE_")`
CreateObject	Procède à la création d'un objet COM à l'aide du *ProgID* associé et retourne un pointeur sur l'objet ainsi créé.	`Set Pres = CreateObject ▼ ("Powerpoint.Application") Pres.Visible = True`
DisconnectObject	Supprime le lien établi avec l'objet spécifié afin de ne plus en gérer les événements.	`DisconnectObject oIE`
Echo	Affiche un message à l'utilisateur au sein d'une boîte de message ou de l'invite de commande.	`WScript.Echo "Allô la planète"`
GetObject	Retourne un pointeur sur un objet existant correspondant au *ProgID* ou au fichier spécifié.	`Set Obj = ▼ GetObject("C:\CAD\SCHEMA.CAD") MyApp = Obj.Application`
Quit	Force la terminaison de l'exécution du script courant.	`Wscript.Quit`
Sleep	Suspend l'exécution du script pour un lapse de temps spécifié en millisecondes.	`WScript.Sleep 500`

Voici l'exemple d'un script simple utilisant certaines des méthodes et propriétés de l'objet `WScript` et permettant de connaître l'hôte utilisé pour exécuter un script ainsi que sa version :

```
HName   =  WScript.FullName
HVersion = WScript.Version
SName = WScript.ScriptFullName
WScript.Echo "Vous utilisez l'hôte " & HName & " v." & HVersion
WScript.Echo "Vous exécutez le script " &  Sname
```

CH04\WScript.vbs



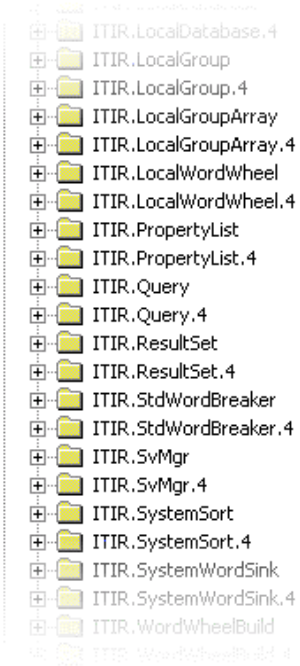
La méthode WScript.CreateObject

La méthode `CreateObject` de l'objet `WScript` permet la création de différents autres objets disponibles sur le système sur lequel le script s'exécute. Cette méthode est très utile dans l'exécution de scripts Windows Script Host puisqu'elle permettra la création des autres objets du modèle `WScript` ainsi que des autres modèles abordés, l'activation d'Internet Explorer, l'établissement d'une connexion sur un système de base de données, la gestion d'IIS, etc.

Notez que cette méthode a été empruntée directement au langage *VBScript* lui-même. Ainsi, puisque le langage possède intrinsèquement cette fonctionnalité, vous pouvez utiliser `CreateObject` sans le précéder du nom d'objet `WScript`. Ainsi, vous utiliserez la méthode native au langage *VBScript* et le résultat sera le même. La seule différence se situera dans le nombre de touches que vos doigts auront dû activer sur le clavier pour rédiger le code.

Les objets disponibles sur le poste sont listés dans la base de registres de celui-ci sous la clé `HKEY_CLASSES_ROOT`. Cette clé liste d'abord les différentes extensions de fichiers reconnus à l'aide d'inscriptions du type `*.doc`, `*.txt`, etc. Ensuite, une série d'inscriptions sont listées et possèdent généralement le format *"Librairie.Objet"* et peuvent ressembler à `IISWAM.Application`, `Imaging.Document` ou `Excel.Worksheet`. Ces inscriptions sont appelées **ProgID** est constituent chacune un objet que votre script peut créer.

La première section d'un ProgID identifie le modèle d'objets, c'est à dire la librairie exécutable à laquelle appartient l'objet. La seconde section du ProgID identifie l'objet lui-même.



Ainsi, si notre script doit provoquer l'affichage de *Microsoft Word*, notre script pourrait s'exécuter comme suit :

```
Dim Wrd
Set Wrd = WScript.CreateObject("Word.Application")
Wrd.Visible = True
```

CH04\Afficher Word.vbs

Notez cependant que chaque modèle d'objet possède ses fonctionnalités propres et son architecture propre. Nous étudierons plusieurs d'entre eux mais il est hors des prétentions de cet ouvrage de tout les couvrir. Il vous faudra donc recourir à l'aide en ligne de ces objets, au navigateur d'objets de *Microsoft Visual Basic* ou *Microsoft Access* plus accessible ou à diverses sources disponibles sur l'Internet.

Cet exemple affiche un document Html en créant un objet Internet Explorer à l'aide de la méthode `CreateObject` et lui demande d'afficher un document Html spécifique. Ensuite, le script attend cinq (5) secondes puis décharge le document Html après avoir affiché une boîte de message. Fort sympatique…

```
Dim oIE, Chemin


'***** Procède à la création de l'objet Internet Explorer *******'
Set oIE = WScript.CreateObject("InternetExplorer.Application")


'******* Détermine l'emplacement du fichier *.htm à afficher ****'
Chemin = WScript.ScriptFullName
Chemin = Left(Chemin, InStrRev(Chemin, "\"))


'*************  Attend que Internet Explorer s'affiche  *********'
Do While oIE.Busy
   WScript.Sleep 200
Loop


'*********************  Charge le document *.htm  ***************'
oIE.Navigate Chemin & "repos.htm"
oIE.Visible = True


'**********************  Patiente cinq secondes   ***************'
WScript.Sleep 5000


'************************   Ferme Internet Explorer   ************'
MsgBox "Bon, c'est assez, retournez au travail maintenant!", ▼
            vbExclamation, "C'est assez"

oIE.Quit
Set oIE = Nothing
```

CH04\Repos.vbs

L'objet WshArguments

L'objet `WshArguments` procure une collection de l'ensemble des paramètres passés en ligne de commande au présent script. Les paramètres se présentent au sein de la collection dans l'ordre dans lequel ils ont été passés au script. L'objet `WshArguments` est utilisé en référençant la propriété `Arguments` de l'objet `WScript`.

Voici la liste complète des méthodes et des propriétés prévues par cet objet.

Propriété	Description	Exemple

Count	Retourne le nombre total de paramètres passés dans la ligne de commande.	`Set Ag = WScript.Arguments`
		`For I = 0 to Ag.Count - 1`
		`  WScript.Echo Ag(I)`
		`Next`

Named	Retourne une collection de l'ensemble des paramètres nommés passés en ligne de commande.	`Set NamedAg = ▼` `WScript.Arguments.Named`
Unamed	Retourne une collection de l'ensemble des paramètres non-nommés passés en ligne de commande.	`Set UnamedAg = ▼` `WScript.Arguments.Unamed`

Méthode	Description	Exemple
ShowUsage	Affiche la documentation personnalisée concernant l'utilisation du présent script. La documentation ne peut être insérée qu'au sein des fichiers de script **.wsf*.	Voir l'exemple complet ci-bas. Notez que cette méthode ne peut être utilisée qu'au sein des fichiers **.wsf* qui seront présentés au chapitre 5.

Note : Les éléments de la collection `WshArguments` sont indexés à partir de 0. Ainsi, si vous désirez parcourir l'ensemble des paramètres, vous devrez utiliser une boucle débutant à 0 et se terminant à `WshArguments.Count –1`.

```
<job>
    <runtime>
        <description>Ce script réamorce un serveur</description>
        <named    name = "Server"
                  helpstring = "Serveur à réamorcer."
                  type = "string"
                  required = "true"
        />
        <example>Exemple: usage.wsf /server:scripting</example>
    </runtime>
    <script language="VBScript">
      If WScript.Arguments.Count <> 1 Then
            WScript.Arguments.ShowUsage
            WScript.Quit
      End If
    </script>
</job>
```

CH04\usage.wsf

Les objets WshNamed et WshUnamed

Imaginons temporairement un script permettant de connecter un lecteur réseau. Ce script pas trop hypothétique attendrait deux arguments :

- La lettre du lecteur à connecter
- Le chemin réseau à connecter sous ce lecteur

Lorsqu'un utilisateur lancera ce script, il pourra spécifier la valeur des arguments à l'aide d'une syntaxe comme la suivante :

```
MonScript.vbs z \\monServeur\chemin
```

À l'aide de cette syntaxe, les arguments sont spécifiés dans l'ordre sans aucune autre directive à l'intention du script afin que ce dernier distingue les différents arguments. Ainsi, le script devra supposer que le premier argument spécifié est bien la lettre à connecter tandis que le second argument est le chemin réseau à connecter. Dans cette situation, l'utilisateur ne peut mélanger l'ordre des arguments sous peine que le script ne génère une erreur.

Par contre, l'utilisateur peut exécuter le script en nommant les différents arguments :

```
MonScript.vbs  /lecteur:z  /chemin:\\monServeur\chemin
```

Ce qui lui procure l'avantage de ne pas à devoir tenir compte de l'ordre des paramètres tout en possédant l'assurance que le script s'exécutera correctement. Il ne s'agit pas là d'une prouesse fort complexe puisque le script récupère la valeur des différents arguments à l'aide de leur nom au lieu de les récupérer à l'aide de leur position ordinale.

Ainsi, les objets **`WshNamed`** et **`WshUnamed`** sont des éléments plus spécifiques de la collection `WshArguments`. Ce dernier fournissant la liste complète des arguments spécifiés par l'utilisateur, l'objet `WshNamed` représente un argument récupéré à l'aide de son nom et est récupéré à l'aide de la propriété **`Named`** de l'objet WshArguments. Quant à lui, l'objet `WshUnamed` représente un argument récupéré par le script à l'aide de sa position ordinale et est récupéré à l'aide de la propriété **`Unamed`**.

Voici la méthode et la propriété prévues par ces objets.

Propriété	Description	Exemple

Count	Retourne le nombre de paramètres passés dans la ligne de commande.	`Set Ag = WScript.Arguments.Named`
		`For N = 0 to Ag.Count - 1`
		`  WScript.Echo Ag(N)`
		`Next`

Méthode	Description	Exemple

Exists	Retourne si l'argument nommé a été spécifié par l'utilisateur ou non. Utilisé seulement avec `WshNamed`.	`If Exists("lecteur") = False Then`
		`  MsgBox "Paramètre Lecteur manquant"`
		`End If`

L'exemple complet suivant permet de connecter un lecteur réseau. Ce script pas trop attend deux arguments :

- **/Lecteur** La lettre du lecteur à connecter
- **/Chemin** Le chemin réseau à connecter sous ce lecteur

L'utilisateur peut utiliser cette commande avec une syntaxe similaire à la suivante :

```
Lecteur.vbs  /lecteur:z  /chemin:\\monServeur\chemin
```

```
Option Explicit

Dim objNet, Lct, Chm

'********* Vérifie l'existence des paramètres nommés *****************'
If WScript.Arguments.Named.Exists("lecteur") = False OR _
                  WScript.Arguments.Named.Exists("chemin") = False Then

   MsgBox "Utilisez la commande comme suit :" & vbCrLf & vbCrLf _
      & "Lecteur.vbs /Lecteur:lecteur /Chemin:chemin", vbInformation

Else

   '*****  Stocke la valeur des paramètres dans deux variables  ******'
   Lct = WScript.Arguments.Named("lecteur")
   Chemin = WScript.Arguments.Named("chemin")


   '************ Procède à la connexion du lecteur réseau ************'
   Set objNet = CreateObject("WScript.Network")
   On Error Resume Next
   Lct = Left(Lct, 1) & ":"
   objNet.MapNetworkDrive Lct, Chemin


   '******************  Détecte toute forme d'erreur  ***************'
   If Err.Number Then
       MsgBox "Le lecteur " & Lct & " était déjà connecté"
   Else
       MsgBox "Le lecteur " & Lct & " est connecté."
   End If
   On Error Goto 0
   Set objNet = Nothing

End If
```

CH04\Lecteur.vbs

L'objet WshNetwork

L'objet `WshNetwork` procure une pluralité de fonctionnalités concernant le réseau à lequel le poste est connecté ainsi que les chemins et imprimantes qui y sont accessibles. Vous créez donc un objet `WshNetwork` lorsque vous désirez connecter ou déconnecter un lecteur ou une imprimante réseau, obtenir des informations sur le réseau et sur l'utilisateur en cours.

Voici la liste complète des méthodes et des propriétés prévues par cet objet.

Propriété	Description	Exemple
ComputerName	Retourne le nom NetBIOS du poste sur lequel s'exécute le script.	`Set Nt = ▼ CreateObject("WScript.Network") MsgBox Nt.ComputerName`
UserDomain	Retourne le nom du domaine à lequel appartient le poste sur lequel s'exécute le script.	`Set Nt = ▼ CreateObject("WScript.Network") MsgBox Nt.UserDomain`
UserName	Retourne le nom de l'utilisateur exécutant le script.	`Set Nt = ▼ CreateObject("WScript.Network") MsgBox Nt.UserName`

Méthode	Description	Exemple
AddWindowsPrinter Connection	Connecte en Windows l'imprimante spécifiée sur l'imprimante réseau spécifiée.	`Nt.AddWindowsPrinterConnection "\\Serveur\Printer"`
AddPrinterConnection	Connecte en MS DOS l'imprimante spécifiée sur l'imprimante réseau spécifiée.	`Nt.AddPrinterConnection ▼ "LPT1", "\\Serveur\Printer"`
EnumNetworkDrives	Retourne une collection contentant le nom des lecteurs réseau actuellement connectés.	`Set oDrv = Nt.EnumNetworkDrives For i=0 to oDrv.Count -1 Step 2 WScript.Echo oDrv(i) ▼ & " = " & oDrv(i+1) Next`
EnumPrinterConnections	Retourne une collection contentant le nom des imprimantes réseau (MS DOS et Windows) actuellement connectés.	`Set oPrn = ▼ Nt.EnumPrinterConnections For i=0 to oPrn.Count -1 Step 2 WScript.Echo oPrn(i) ▼ & " = " & oPrn(i+1) Next`
MapNetworkDrive	Connecte le lecteur spécifié sur le chemin réseau spécifié.	`Nt.MapNetworkDrive "z:", ▼ "\\Serveur\chemin"`
RemoveNetworkDrive	Déconnecte le lecteur spécifié.	`Nt.RemoveNetworkDrive "z:"`
RemovePrinter Connection	Déconnecte l'imprimante spécifiée.	`Prn = "\\prnSvr\DefaultPrinter" Nt.RemovePrinterConnection Prn`
SetDefaultPrinter	Spécifie l'imprimante par défaut.	`Prn = "\\prnSvr\DefaultPrinter" Nt.AddPrinterConnection ▼ "LPT1", "\\Serveur\Printer" Nt.SetDefautlPrinter Prn`

L'exemple suivant récupère plusieurs informations au sujet de l'utilisateur ainsi que du poste sur lequel ce dernier est actuellement connecté *(liste des lecteurs réseau et des imprimantes)*. Les informations sont concaténés au sein d'une phrase avant d'être affichées en totalité à l'aide d'une boîte de message.

```
Option Explicit

Dim Texte, NT
Dim Drv, Prn, K

Set NT = WScript.CreateObject("WScript.Network")
Set Drv = NT.EnumNetworkDrives()
Set Prn = NT.EnumPrinterConnections()


'****************** Informations générales ************************'
Texte = "Bonjour " & NT.UserName & ", " & vbCrLf & vbCrLf
Texte = Texte & "vous êtes connecté sur l'ordinateur "  ▼
          & NT.ComputerName
Texte = Texte & " appartenant au domaine " & NT.UserDomain & "." ▼
          & vbCrLf & vbCrLf


'*************** Liste des lecteurs réseau ************************'
Texte = Texte & "Ce poste possède les lecteurs réseaux suivants:" ▼
          & vbCrLf
If Drv.Count = 0 Then
   Texte = Texte & "   * Aucun lecteur réseau détecté." & vbCrLf
Else
   For K = 0 to Drv.Count - 1 Step 2
       Texte = Texte & "   * " & Drv(K) & " = " & Drv(K+1) & vbCrLf
   Next
End If


'**************** Liste des imprimantes **************************'
Texte = Texte & vbCrLf & "Ce poste possède les imprimantes " ▼
          & "suivantes:" & vbCrLf
If Prn.Count = 0 Then
   Texte = Texte & "   * Aucune imprimante détectée." & vbCrLf
Else
   For K = 0 to Prn.Count - 1 Step 2
      Texte = Texte & "   * " & Prn(K) & " = " & Prn(K+1) & vbCrLf
   Next
End If


'****************** Affiche le message ************************'
MsgBox Texte, vbInformation, "Bonjour"
```

CH04\Network.vbs

L'objet WshShell

L'objet `WshShell` procure une série de fonctionnalités permettant de piloter le système d'exploitation et d'en extirper différentes informations utiles comme les valeurs des variables d'environnement, le chemin des répertoires spéciaux, etc.

Voici la liste complète des méthodes et des propriétés prévues par cet objet.

Propriété	Description	Exemple
CurrentDirectory	Retourne ou spécifie le répertoire actif.	`Rep = Sh.CurrentDirectory`
Environment	Retourne l'objet `WshEnvironment` contenant la liste des variables d'environnement et leurs valeurs. Tapez `Set` à l'invite de commande pour connaître la liste des variables d'environnement de votre système.	`Set Env = ▼` `Sh.Environment("systemroot")`

SpecialFolders	Retourne le chemin complet d'un répertoire spécial. On entend par répertoire spéciaux les répertoires *system32*, *program files*, *desktop*, etc. Voyez ci-dessous la liste des constantes existants à ce sujet.	`Rep = ▼`
		`Sh.SpecialFolders("Desktop")`
		`MsgBox Rep`

Voici la liste des chaînes de caractères constantes que vous pouvez utiliser avec la méthode `SpecialFolders`. Notez que la casse n'importe pas lorsque vous précisez la valeur.

SpecialFolders	Description
AllUsersDesktop	Éléments du bureau partagé par tous les profils utilisateurs.
AllUsersStartMenu	Éléments du menu `Démarrer` partagé par tous les profils utilisateurs.
AllUsersPrograms	Éléments du menu `Démarrer/Programmes` partagé par tous les profils utilisateurs.
AllUsersStartup	Éléments du menu `Démarrer/Programmes/Démarrage` partagé par tous les profils utilisateurs.
Desktop	Éléments du menu `Démarrer` de l'utilisateur en cours.
Favorites	Éléments du menu `Démarrer/Favoris` de l'utilisateur en cours.
Fonts	Éléments du répertoire `Fonts` contenant les polices de caractères installées.
MyDocuments	Éléments du répertoire `MesDocuments` de l'utilisateur en cours.
NetHood	Éléments de voisinage réseau de l'utilisateur en cours.
PrintHood	Éléments des imprimantes réseau de l'utilisateur en cours.
Programs	Éléments du menu `Démarrer/Programmes` de l'utilisateur en cours.
Recent	Éléments du menu `Démarrer/Documents` de l'utilisateur en cours.
SendTo	Éléments du répertoire `SendTo` de l'utilisateur en cours.
StartMenu	Éléments du menu `Démarrer` de l'utilisateur en cours.
StartUp	Éléments du menu `Démarrer/Programmes/Démarrage` de l'utilisateur en cours.
Templates	Éléments du répertoire `Modèles` contenant les modèles de documents vides.

Méthode	Description	Exemple

AppActivate	Active une fenêtre existante par son titre et lui donne le focus.	`Sh.Run "notepad"`
		`WScript.Sleep 200`
		`Sh.Activate "notepad"`
		`Sh.SendKeys "Allô"`

CreateShortcut	Procède à la création d'un objet `WshShortcut` permettant de créer un raccourci. Voyez la description plus bas pour plus de détails.	`chemin = "c:\raccourci.lnk"`
		`Set Lnk = ▼`
		`Sh.CreateShortcut(chemin)`
		`Lnk.TargetPath = "notepad.exe"`
		`Lnk.Save`

ExpandEnvironment Strings	Retourne la valeur étendue d'une variable d'environnement.	`WinDir = ▼`
		`Sh.ExpandEnvironmentStrings(▼`
		`"%windir%")`
		`MsgBox "WinDir = " & WinDir`

LogEvent	Ajoute une entrée dans le journal d'événements `Application`. Voyez le tableau plus bas pour connaître les types de messages disponibles.	`Msg = "Le script a terminé."` `Sh.LogEvent 0, Msg`
Popup	Boîte de dialogue partagée par tous les langages de script. Préférez l'instruction `MsgBox` intrinsèque à Visual Basic.	`Sh.Popup "Allô la planète!",,▼` `"Titre", 64`
RegDelete	Supprime une clé de la base de registres.	`Sh.RegDelete "HKCU\Soft\ACME\"`
RegRead	Lit la valeur d'une clé de la base de registres.	`Valeur = ▼` `Sh.RegRead("HKCU\Soft\ACME\")`
RegWrite	Spécifie la valeur d'une clé de la base de registres.	`Sh.RegWrite "HKCU\Soft\ACME\",▼` `42, "REG_BINARY"`
Run	Démarre une application par son chemin complet. Voyez les *Technique avancées de Script* au chapitre 5 pour des exemples approfondies.	`Sh.Run "c:\winnt\notepad.exe"`
SendKeys	Envoie le signal de touches enfoncées comme si l'utilisateur tapait sur le clavier.	`Sh.Activate "notepad"` `Sh.SendKeys "Allô"`

Voici la liste des valeurs que vous pouvez utiliser avec la méthode `LogEvent`. Notez que Windows 9x et ME consigneront le journal au sein d'un fichier `WSH.log` situé dans le répertoire `Windows`.

Type de message	Description
0	SUCCESS – L'application s'est exécuté avec succès.
1	ERROR – Enregistrement d'une erreur ayant causée des problèmes graves.
2	WARNING – Enregistrement d'un avertissement pouvant éventuellement causer des problèmes.
4	INFORMATION – Enregistrement d'une information particulière.
8	AUDIT_SUCCESS – Enregistrement d'une tentative d'accès sécurisé qui a réussie.
16	AUDIT_FAILURE – Enregistrement d'une tentative d'accès sécurisé qui a échouée.

Création de raccourcis

Les objets `WshShortcut` et `WshURLShortcut` permettent des créer des raccourcis sur le disque de l'utilisateur. Pour ce faire, il suffit d'obtenir une référence sur un nouvel objet de type `WshShortcut` ou `WshURLShortcut`, d'en configurer ensuite la destination, l'icône, etc. avant de le sauvegarder à l'aide de la méthode `Save`. Aucun raccourci ne sera créé si vous omettez d'invoquer la méthode `Save`.

La première étape consiste donc à créer un nouvel objet de type `WshShortcut`. L'extrait suivant procède à la création d'un raccourci nommé `monRaccourci` à la racine du disque c :

```
Dim MonRaccourci, Sh
Set Sh = WScript.CreateObject("WScript.Shell")
Set MonRaccourci = Sh.CreateShortcut("c:\monRaccourci.lnk")
```

Ensuite, il faut piloter le nouveau raccourci en attribuant les valeurs désirées à ses propriétés avant de le sauvegarder à l'aide de la méthode `Save` :

```
MonRaccourci.TargetPath = "c:\program files\microsoft " _
      & office\office\winword.exe"

MonRaccourci.WindowStyle = 1

MonRaccourci.IconLocation = "c:\program files\microsoft " _
      & office\office\winword.exe, 0"

MonRaccourci.Description = "Démarrez Microsoft Word"

MonRaccourci.Save
```

Notez que la propriété `IconLocation` peut pointer sur un fichier exécutable `*.exe` ou `*.dll` possédant des icônes ou sur un icône `*.ico` standard. Dans le premier cas, vous devez spécifiez l'identificatif numérique de l'icône au sein du fichier binaire. Dans le second cas, seul le nom du fichier icône suffit.

La propriété `WindowStyle` permet de spécifier l'état de la fenêtre dans laquelle le logiciel pointé s'affichera. La propriété peut prendre les valeurs suivantes :

Valeur WindowStyle	Description
0	Aucune fenêtre ne sera créée pour le logiciel qui s'exécutera de façon invisible.
1	Le logiciel s'affichera dans une fenêtre normale et obtiendra le focus au démarrage.
2	Le logiciel s'affichera dans une fenêtre réduite en icône et obtiendra le focus au démarrage.
3	Le logiciel s'affichera dans une fenêtre pleines dimensions et obtiendra le focus au démarrage.
4	Le logiciel s'affichera dans une fenêtre normale et n'obtiendra pas le focus au démarrage.
6	Le logiciel s'affichera dans une fenêtre réduite en icône et n'obtiendra pas le focus au démarrage.

L'exemple suivant crée sur le bureau de tous les utilisateurs du système un raccourci vers `Internet Explorer`. Le script se sert de la propriété `SpecialFolders` afin d'obtenir le chemin du répertoire représentant le bureau partagé par tous les profils utilisateurs.

```
Dim MonRaccourci, Sh
Dim Bureau

Set Sh = WScript.CreateObject("WScript.Shell")


'********* Récupère le répertoire représentant le Bureau ***********'
Bureau = Sh.SpecialFolders("AllUsersDesktop")


'**************** Crée le raccourci sur le Bureau *****************'
Set MonRaccourci = Sh.CreateShortcut(Bureau & "\internet.lnk")


'***************** Configure le nouveau raccourci *****************'
MonRaccourci.TargetPath = "c:\program files\internet " _
      & "explorer\iexplore.exe"
MonRaccourci.WindowStyle = 3
MonRaccourci.IconLocation = "c:\program files\internet " _
      & "explorer\iexplore.exe, 0"
MonRaccourci.Description = "Se connecter à l'internet"


'******************** Sauvegarde le raccourci *********************'
MonRaccourci.Save
```

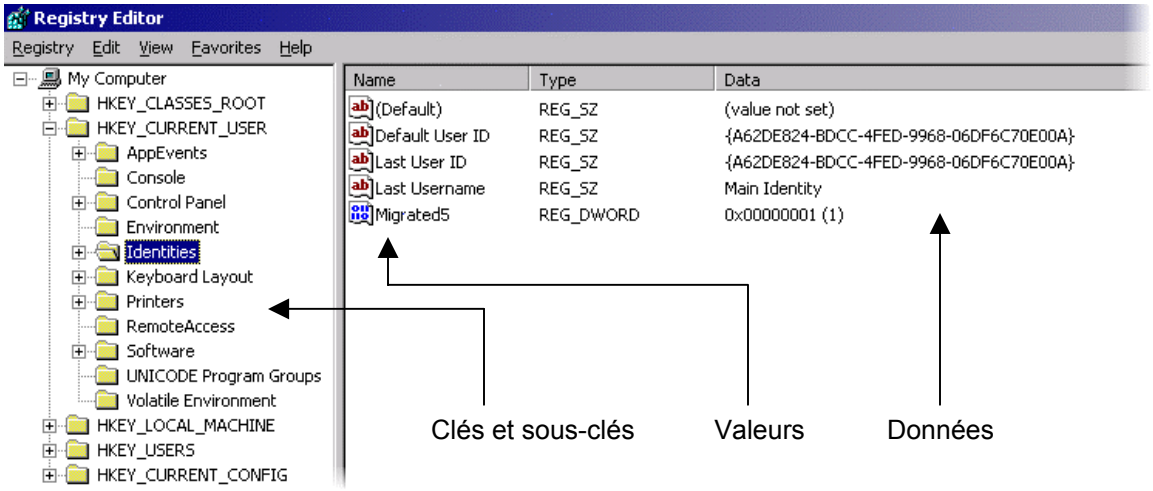
CH04\Raccourci.vbs

Notez que certains logiciels antivirus n'apprécieront peut-être pas ce type de manipulations des éléments du bureau des utilisateurs. Ainsi, il vous faudra vous assurer de la bonne configuration de votre antivirus si vous escomptez utiliser ces fonctions en environnement de production.

Manipulation de la base de registres

La base de registres est une base de données utilisée par *Microsoft®* depuis *Windows 95* où sont centralisées les configurations matérielle et logicielle, les préférences des utilisateurs, etc. Il est possible d'ajouter, modifier, lire et supprimer des clés de la base de registres à l'aide de *Windows Script Host*. Notez d'abord que *Microsoft®* conseille que l'utilisateur utilise le panneau de configuration de son poste afin de modifier les configurations consignées au sein de la base de registres et prévient que toute modification manuelle des valeurs qui y sont contenues pourrait mettre en péril le bon fonctionnement du système d'exploitation. Et *Microsoft®* n'a pas tort à ce sujet. Prenez garde de bien connaître les valeurs que vous désirez modifier ainsi que leur utilité avant d'en altérer le contenu. Vous êtes averti.

La base de registres présente les informations de manière hiérarchique où chacune des données est associée à une valeur et où chacune des valeurs appartient à appartient à une sous-clé ou une clé.



Les informations sont réparties thématiquement au sein de clés de base qui ne peuvent être altérées et qui regroupent les sous-clés selon leur utilité :

Clé de base	Abrév.	Description
HKEY_CLASSES_ROOT	HKCR	Contient les extensions des fichiers et les programmes et actions associées à ces fichiers ainsi que la liste d'objets COM pouvant être créés sur le poste à l'aide de l'instruction `CreateObject`.
HKEY_CURRENT_USER	HKCU	Contient les préférences logicielles et le profil matériel de l'utilisateur en cours.
HKEY_LOCAL_MACHINE	HKLM	Contient les configurations logicielles et matérielles applicables à l'ensemble de l'ordinateur.
HKEY_USERS	*N/a*	Contient les préférences partagées par l'ensemble des utilisateurs. Contient également le profil appliqué à tout nouvel utilisateur authentifié pour la première fois.
HKEY_CURRENT_CONFIG	*N/a*	Contient les différences entre la configuration matérielle actuelle et celle utilisée lors du démarrage du système. Cette clé permet le redémarrage en utilisant la dernière bonne configuration connue.

La méthode **RegRead** permet de lire la valeur d'une clé de la base de registres. La fonction attend en paramètre la clé à ouvrir et, en échange, retourne la valeur qui y est contenue. Notez que si le nom de la clé à ouvrir se termine par une barre oblique (\ ), la valeur par défaut sera récupérée, celle qui est indiquée sous le nom *"(par défaut)"* au sein de l'éditeur de registre.

Il est donc possible de récupérer le navigateur web par défaut sur le poste en cours en examinant quel logiciel est associé aux fichiers possédant l'extension `*.htm`.

Au sein de la clé de base `HKEY_CLASSES_ROOT`, l'ensemble des extensions connues des fichiers sont stockées et contiennent, en valeur par défaut, une description du fichier appelée `FriendlyName`. Une autre clé ayant pour nom la même valeur que ce `FriendlyName` possède le chemin du logiciel à exécuter lorsque ce type de fichier est exécuté :

- Lorsque l'utilisateur active un fichier `*.htm`, *Windows* recherche la clé .htm au sein de la clé de base `HKEY_CLASSES_ROOT`;
- Si la clé est trouvée, la valeur par défaut est prise en note. Ensuite, Windows recherche une clé de ce nom située à la racine de la clé de base `HKEY_CLASSES_ROOT`;
- Si la clé est trouvée, *Windows* recherche la sous-clé `"\shell\open\command"` et récupère la valeur par défaut qui y est stockée;
- Si la clé et la valeur sont trouvée, *Windows* exécute le logiciel spécifié.

Suivons le même cheminement que *Windows* et trouvons le chemin du fichier exécutable servant de navigateur web par défaut :

```
Dim IE, Sh

Set Sh = WScript.CreateObject("WScript.Shell")

IE = Sh.RegRead("HKEY_CLASSES_ROOT\.htm\")
IE = Sh.RegRead("HKEY_CLASSES_ROOT\" & IE & "\shell\open\command\")

MsgBox IE, vbInformation, "Navigateur web par défaut"
```

CH04\NavigateurWeb.vbs

La méthode **`RegWrite`** permet d'écrire la valeur d'une clé de la base de registres et d'en créer une nouvelle si celle spécifiée n'existe pas. La fonction attend en paramètres la clé à ouvrir, la valeur à y inscrire et, optionnellement, le type de données à y inscrire.

En *Windows Script Host* , les données stockées au sein de la base de registres peuvent être des valeurs d'un des types suivants. Par défaut, le type `REG_SZ` sera utilisé.

Type de données	Description
REG_BINARY	Valeur numérique stockée en format binaire.
REG_DWORD	Valeur numérique sur 16-bits.
REG_SZ	Chaîne de caractère.
REG_EXPAND_SZ	Chaîne de caractère extensible (Ex : *"%windir%\\notepad.exe"*).

Notez que si le nom de la clé à ouvrir se termine par une barre oblique (\ ), la valeur par défaut sera écrite ou créée, celle qui est indiquée sous le nom *"(par défaut)"* au sein de l'éditeur de registre.

L'exemple suivant ajoute une commande associée aux fichiers `*.dll`. Il commence par récupérer le `FriendlyName` associé aux fichiers possédant l'extension `*.dll` puis écrit une nouvelle clé correspondant à la nouvelle action et y stocke le nom du logiciel à exécuter lorsque cette commande est invoquée par l'utilisateur. En occurrence, le logiciel `RegSvr32.exe` servira à enregistrer automatiquement une librairie COM à l'extension `*.dll` au sein de la base de registres :

```
Dim DLL, Sh

Set Sh = WScript.CreateObject("WScript.Shell")

DLL = Sh.RegRead("HKEY_CLASSES_ROOT\.dll\")
Sh.RegWrite "HKEY_CLASSES_ROOT\" & DLL & "\shell\" _
        & "Enregistrer dans le registre\command\", "regsvr32.exe %1"

MsgBox "Les fichiers *.dll peuvent être enregistrés dans le " _
    & registre", vbInformation, "Association réussie"
```

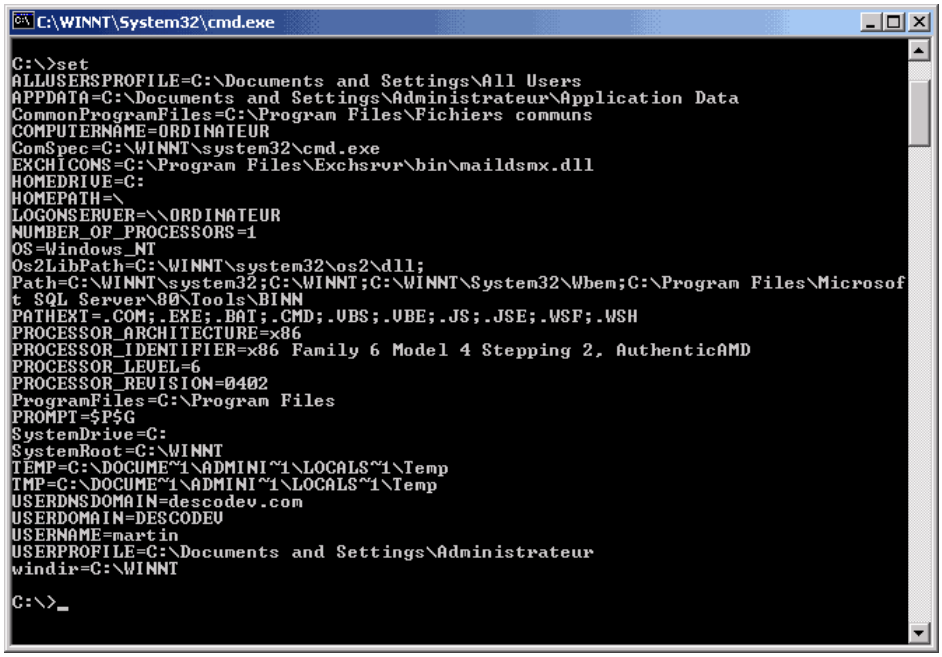
CH04\EnregistrerDLL.vbs

Dès lors, lorsque l'utilisateur activera le menu contextuel sur un fichier `*.dll`, le menu contextuel suivant lui sera affiché. Notez que l'utilisateur peut désormais **Enregistrer dans le registre** les différents fichiers `*.dll`.

Gestion des variables d'environnement

Les variables d'environnement sont des chaînes de caractères contenant des informations concernant le système associée à un nom symbolique et sont utilisées par le système d'exploitation pour référencer différentes ressources : répertoires, lecteurs, nom de l'utilisateur ou de l'ordinateur, etc.

À l'invite de commandes, tapez la commande `SET` afin de visualiser la liste des variables d'environnement de votre poste ainsi que les valeurs qui leurs sont associées.



Les variables d'environnement sont regroupées au sein de trois grandes familles :

- Les variables `SYSTEM` retournent des valeurs assignées au niveau du système sans considération à l'utilisateur en cours.
- Les variables `USER` retournent des valeurs concernant l'utilisateur en cours.
- Les variables `PROCESS` retournent des valeurs concernant le processus en cours. Ce sont les seules variables utilisables sous les systèmes 95, 98 et ME.

Lorsque vous désirez obtenir une référence vers une variable d'environnement, vous devez créer un objet de type `WshEnvironment` contenant un tableau des variables d'environnement de la famille spécifiée. Vous pouvez spécifier la famille de variables pour laquelle vous désirez obtenir la variable. Cependant, cette dernière information est facultative. Si vous omettez cette information, la variable d'environnement vous sera retournée sans considération à sa famille. Le script suivant obtient le nombre de processeurs installés sur la machine en consultant la variable d'environnement `NUMBER_OF_PROCESSORS` :

```
Set Ws = WScript.CreateObject("WScript.Shell")
Set SysEnv = Ws.Environment("SYSTEM")
MsgBox SysEnv("NUMBER_OF_PROCESSORS") & " processeurs."
```

Voici la liste des variables d'environnement mises à votre disposition en *Windows Script Host* :

Variable	Description	SYSTEM	USER	PROCESS NT/2000/XP	PROCESS 95/98/ME
NUMBER_OF_ PROCESSORS	Nombre de processeurs en cours sur la machine.	x	-	x	-
PROCESSOR_ ARCHITECTURE	Type de processeur (x86, Alpha, Mips)	x	-	x	-
PROCESSOR_ IDENTIFIER	Chaîne de caractères identifiant le processeur.	x	-	x	-
PROCESSOR_ LEVEL	Type de processeur 2 = 286 3 = 386 4 = 486 5 = Pentium 6 = Pentium Pro ou Pentium II et plus	x	-	x	-
PROCESSOR_ REVISION	Numéro de révision du processeur.	x	-	x	-
OS	Chaîne de caractères identifiant le système d'exploitation.	x	-	x	-
COMSPEC	Chemin du fichier exécutable de l'invite de commande habituellement cmd.exe ou command.com	x	-	x	x
HOMEDRIVE	Lecteur local principal, habituellement le C:\	-	-	x	-
HOMEPATH	Répertoire par défaut pour les utilisateurs.	-	-	x	-
PATH		x	x	x	x
PATHEXT	Extension des fichiers exécutables.	x	-	x	-
PROMPT	Invite de commande.	-	-	x	x
SYSTEMDRIVE	Lecteur hébergeant le répertoire système, généralement le C:\	-	-	x	-
SYSTEMROOT	Chemin du répertoire système (c:\winnt).	-	-	x	-
WINDIR	Chemin du répertoire système (c:\winnt).	x	-	x	x
TEMP	Répertoire contenant les fichiers temporaires.	-	x	x	x
TMP	Répertoire contenant les fichiers temporaires.	-	x	x	x

Les objets Scripting

Les objets `Scripting` permettent de manipuler les lecteurs, dossiers et fichiers du système. Vous pouvez donc, à l'aide de ce modèle d'objets, créer, modifier ou supprimer des dossiers ou des fichiers.

Objet	Ce que permet cet objet
FileSystemObject	Objet de base permettant d'accéder à l'ensemble des autres objets du modèle.
Drives	Collection permettant d'accéder à l'ensemble des lecteurs du système peu importe s'ils sont amovibles, fixes ou distants.
Drive	Accéder aux informations d'un lecteur spécifié.
Files	Collection permettant d'accéder à l'ensemble des fichiers du dossier spécifié.
File	Accéder aux informations d'un fichier spécifié.
Folders	Collection permettant d'accéder à l'ensemble des dossiers du lecteur spécifié.
Folder	Accéder aux informations d'un dossier spécifié.
TextStream	Lire et écrire des fichiers en format texte.

L'objet FileSystemObject

L'objet `FileSystemObject` est l'objet racine du modèle d'objets de *Scripting* et permet de créer et d'accéder aux différents autres objets du modèle en plus d'inclure de manière redondante plusieurs fonctionnalités de ces derniers dans le seul but de simplifier la programmation des scripts.

Voici la liste complète des méthodes et des propriétés prévues par cet objet.

Propriété	Description	Exemple

Drives	Retourne la collection d'objets `Drive` représentant à chacun d'eux un lecteur présent sur le système.	`Set FSO = _`
		`  CreateObject("Scripting." _`
		`  & "FileSystemObject")`
		`Drv = FSO.Drives`

Méthode	Description	Exemple

BuildPath	Retourne un chemin valide constitué de deux expressions à concaténer.	`Chemin = FSO.BuildPath( ▼`
		`  "c:\temp", "dossier")`
		`' Retourne c:\temp\dossier`

CopyFile	Copie un ou plusieurs fichiers à l'endroit spécifié.	`FSO.CopyFile ▼`
		`  "c:\Documents\*.doc", ▼`
		`  "c:\dossier\"`

CopyFolder	Copie un ou plusieurs répertoires et son contenu à l'endroit spécifié.	`FSO.CopyFolder ▼`
		`  "c:\Documents\*", ▼`
		`  "c:\dossier\"`

CreateFolder	Crée le nouveau répertoire spécifié.	`FSO.CreateFolder("c:\temp")`

CreateTextFile	Crée le fichier spécifié et renvoie un objet `TextStream` qui peut être utilisé pour lire ou écrire dans le fichier.	`Set Fch = FSO.CreateTextFile(▼`
		`  "c:\test.txt")`
		`Fch.WriteLine "Test"`
		`Fch.Close`

DeleteFile	Supprime le fichier spécifié.	`FSO.DeleteFile "c:\test.txt"`
DeleteFolder	Supprime le répertoire spécifié.	`FSO.DeleteFolder "c:\dossier"`
DriveExists	Retourne `True` si le lecteur spécifié existe et `False` dans le cas contraire.	`If FSO.DriveExists("c:") Then` `End If`

FileExists	Retourne `True` si le fichier spécifié existe et `False` dans le cas contraire.	`If FSO.FileExists( ▼`
		`  "c:\test.txt") Then`
		`End If`

FolderExists	Retourne `True` si le répertoire spécifié existe et `False` dans le cas contraire.	`If FolderExists( ▼`
		`  "c:\dossier") Then`
		`End If`

GetAbsolutePathName	Retourne un chemin valide et absolu selon une expression probablement ambigu possédant des caractères frimes.	`chm=FSO.GetAbsolutePathName(▼`
		`  "c:\*.*\docs")`
		
		`' Retourne "c:\dossier\docs"`

GetBaseName	Retourne un chemin de dossier sans le nom de fichier à partir d'une expression spécifiée.	`chemin ="c:\dossier\texte.txt"`
		`chm=FSO.GetBaseName(chemin)`
		
		`' Retourne "c:\dossier\"`

GetDrive	Retourne un objet de type `Drive` à partir d'un chemin spécifié.	`drv = FSO.GetDrive("c:\")` `MsgBox Drv.VolumeName`
GetDriveName	Retourne la lettre du lecteur correspondant au chemin spécifié.	`chemin ="c:\dossier\texte.txt"` `drv = FSO.GetDriveName(chemin)`

GetExtensionName	Retourne l'extension du fichier spécifié.	`Fc = "c:\boot.ini"`
		`Ext = FSO.GetExtensionName(Fc)`
		
		`' Retourne "ini"`

GetFile	Retourne un objet de type `File` correspondant au fichier du chemin spécifié.	`Fch=FSO.GetFile("c:\boot.ini")` `MsgBox Fch.DateCreated`

GetFileName	Retourne le nom complet du fichier sans le chemin à partir d'une expression spécifiée.	`chemin ="c:\dossier\texte.txt"`
		`Fch=FSO.GetFileName(chemin)`
		`MsgBox Fch`

GetFolder	Retourne un objet de type `Folder` correspondant au chemin spécifié.	`Fld=FSO.GetFile("c:\dossier\")` `MsgBox Fld.DateCreated`

GetParentFolderName	Retourne le nom du dossier parent au dossier spécifié.	`Chemin = "c:\dossier\tmp"`
		`Parent = ▼`
		`FSO.GetParentFolderName(Chemin)`
		`MsgBox Chemin & " appartient" _`
		`  & " au dossier " & Parent`

GetSpecialFolder	Retourne un objet de type `Folder` correspondant au dossier spécial spécifié.	`Set Fld = ▼` `FSO.GetSpecialFolder(SystemFolder)`

GetTempName	Retourne un nom de fichier aléatoire et unique pouvant servir de nom de fichier temporaire.	`tmp = FSO.GetTempName()`
		
		`Set Fch = FSO.CreateTextFile(tmp)`
		`Fch.WriteLine "Test"`
		`Fch.Close`

MoveFile	Déplace un fichier de la source spécifiée à la destination spécifiée.	`FSO.MoveFile "c:\a.txt", ▼` `"d:\a.txt"`
MoveFolder	Déplace un répertoire et son contenu de la source spécifiée à la destination spécifiée.	`FSO.MoveFolder "c:\dossier\", ▼` `"d:\dossier\"`

OpenTextFile	Ouvre le fichier spécifié et renvoie un objet `TextStream` qui peut être utilisé pour lire ou écrire dans le fichier.	`Set Fch = FSO.OpenTextFile(▼`
		`"c:\fichier.txt")`
		`Fch.WriteLine "Test"`
		`Fch.Close`

L'exemple suivant permet d'effectuer une sauvegarde *(backup)* du répertoire `c:\inetpub\` en un répertoire `d:\backup\inetpub\`. L'utilisation des objets `FileSystemObject` possède l'avantage de créer le répertoire cible si ce dernier n'existe pas.

```
Dim FSO
Set FSO = CreateObject("Scripting.FileSystemObject")
FSO.CopyFolder "c:\inetpub\*", "d:\Backup\inetpub\"
```

CH04\BackupFSO.vbs

L'objet Drive

L'objet `Drive` représente un lecteur physique ou réseau et peut être obtenu à partir de la propriété `Drives` de l'objet `FileSystemObject`. On obtient une référence sur un lecteur à l'aide de la lettre l'identifiant comme le démontre l'exemple suivant qui obtient un objet `Drive` référençant le lecteur "c:" du système :

```
Dim Fso, Drv
Set Fso = CreateObject("Scripting.FileSystemObject")
Set Drv = Fso.Drives("c")
```

Voici la liste complète des propriétés prévues par cet objet.

Propriété	Description	Exemple
AvailableSpace	Retourne le nombre d'octets disponibles sur le lecteur. La valeur peut différer de la propriété `FreeSpace` puisque celle-ci ne tient pas compte des quotas appliqués à l'utilisateur sur le lecteur.	`nEspace = drv.AvailableSpace` `MsgBox nEspace / 1024 & "Ko"`

DriveLetter	Retourne la lettre du lecteur.	`If drv.IsReady Then`
		`  MsgBox drv.DriveLetter ▼`
		`  & " est prêt!"`
		`End If`

DriveType	Retourne le type de lecteur. Voyez la liste des types possibles de lecteur plus bas.	`If drv.DriveType = 4 Then`
		`  MsgBox "Impossible d'écrire ▼`
		`  sur un support CD"`
		`End If`

FileSystem	Retourne le système de fichiers utilisé sur le lecteur.
	Peut retourner les valeurs "FAT", "NTFS" ou "CDFS".
		`Fs = drv.FileSystem`
		`If LCase(fs)<> "ntfs" Then`
		`  MsgBox "Le lecteur doit être▼`
		`  en NTFS pour continuer."`
		`End If`

FreeSpace	Retourne le nombre d'octets disponibles sur le lecteur. La valeur peut différer de la propriété `AvailableSpace` puisque celle-ci tient compte des quotas appliqués à l'utilisateur sur le lecteur.	`Espace = drv.FreeSpace / 1024` `MsgBox Espace " Ko disponibles"`

IsReady	Retourne `True` si le lecteur est prêt et `False` dans le cas contraire.	`If drv.IsReady = True Then`
		`  MsgBox drv.DriveLetter ▼`
		`  & " est prêt!"`
		`End If`

Path	Retourne le chemin d'accès au lecteur.	`MsgBox drv.Path`
RootFolder	Retourne un objet de type Folder référençant la racine du lecteur.	
SerialNumber	Retourne le numéro de série du lecteur.	`MsgBox drv.SerialNumber`

ShareName	Retourne le nom de partage du lecteur dans le cas d'un lecteur de type réseau *(DriveType = 3)* ou retourne une chaîne vide dans tous les autres cas.	`MsgBox drv.ShareName`
TotalSize	Retourne le nombre total d'octets du lecteur.	`nEspace = drv.FreeSpace / 1024` `MsgBox nEspace " Ko total"`
VolumeName	Détermine ou retourne le nom de volume du lecteur.	`MsgBox drv.VolumeName`

Voici la liste des types possibles de lecteurs retournés par la propriété `DriveType` :

Type de lecteur	Valeur
Inconnu	0
Disquette et périphérique amovible	1
Fixe	2
Lecteur réseau	3
CD-Rom et DVD-Rom	4
Disque virtuel	5

Le code suivant parcours l'ensemble des objets `Drive` disponibles dans la collection `Drives` du `FileSystemObject` et en fait l'affichage du nom de volume et de l'espace disque disponible ou du nom de partage le cas échéant :

```
Dim Fso, Drv, Dc, n, Capc, Disp

Set Fso = CreateObject("Scripting.FileSystemObject")

For Each Drv in Fso.Drives

   If Drv.DriveType = 3 Then
      n = Drv.ShareName
   Else

      If Drv.IsReady Then
            n = Drv.VolumeName
            Capc = Drv.TotalSize / 1024 / 1024 & " Mo"
            Capc = vbCrLf & "Capacité: " & Capc
            Disp = Drv.AvailableSpace / 1024 / 1024 & " Mo"
            Disp = vbCrLf & "Disponible: " & Disp
      Else
         n = "< Non prêt >"
         Capacite = ""
         Disponible = ""
      End If

   End If
   MsgBox Drv.DriveLetter & " - " & n & Capc & Disp
Next
```

CH04\Lecteurs.vbs

L'objet Folder

L'objet `Folder` représente un dossier obtenu par la méthode `RootFolder` de l'objet `Drive` ou la méthode `GetFolder` de l'objet `FileSystemObject`. L'exemple suivant crée un objet de type `Folder` représentant un dossier existant sur le système :

```
Dim Fso, Fld
Set Fso = CreateObject("Scripting.FileSystemObject")
Set Fld = Fso.GetFolder("c:\program files\monDossier\")
```

Voici la liste complète des méthodes et des propriétés prévues par l'objet `Folder`.

Propriété	Description	Exemple
Attributes	Détermine ou retourne les attributs du dossier. Voyez la liste des attributs possibles plus loin.	`Fld.Attributes = ReadOnly`
DateCreated	Retourne la date de création du dossier.	`MsgBox "Le dossier a été créé ▼ le " & Fld.DateCreated`
DateLastAccessed	Retourne la date à laquelle le dossier a été accédé pour la dernière fois.	`MsgBox "Le dossier a été accédé▼ le " & Fld.DateLastAccessed`
DateLastModified	Retourne la date à laquelle le dossier a été modifié pour la dernière fois.	`MsgBox"Le dossier a été modifié▼ le " & Fld.DateLastModified`
Drive	Retourne la lettre du lecteur contenant le dossier.	`MsgBox "Le dossier se trouve ▼ sur le lecteur " & Fld.Drive`
Files	Retourne une collection contenant des objets `File` représentant l'ensemble des fichiers du dossier.	`For Each Fl In Fld.Files MsgBox Fl.Name Next`
IsRootFolder	Retourne si le dossier est le dossier racine d'un lecteur.	`If Fld.IsRootFolder Then End If`
Name	Détermine ou retourne le nom du dossier.	`MsgBox Fld.Name`
ParentFolder	Retourne un objet `Folder` représentant le dossier parent.	`MsgBox Fld.ParentFolder.Path`
Path	Retourne le chemin complet du dossier.	`MsgBox Fld.Path`
ShortName	Retourne le nom du dossier sous le format compatible 8.3.	`MsgBox Fld.ShortName`
ShortPath	Retourne le chemin du dossier sous le format compatible 8.3.	`MsgBox Fld.ShortPath`
Size	Retourne la taille en octets du contenu entier du dossier.	`MsgBox Fld.Size / 1024 & " Ko"`
SubFolders	Retourne une collection contenant des objets `Folder` représentant l'ensemble des sous-dossiers du dossier.	`For Each subFld In Fld.SubFolders MsgBox subFld.Name Next`
Type	Retourne le type de dossier *"Dossier de fichiers"*	`MsgBox Fld.Type`

Méthode	Description	Exemple
Copy	Copie le répertoire et son contenu à l'endroit spécifié.	`Fld.Copy "c:\dossier\"`
Delete	Supprime le répertoire.	`Fld.Delete`
Move	Déplace le répertoire et son contenu à la destination spécifiée.	`Fld.Move "d:\dossier\"`

L'exemple suivant permet d'effectuer une sauvegarde *(backup)* du répertoire `c:\inetpub\` en un répertoire `d:\backup\inetpub\`.

```
Dim FSO, Fld

Set FSO = CreateObject("Scripting.FileSystemObject")

If FSO.FolderExists("d:\Backup\inetpub\") = False Then
   FSO.CreateFolder "d:\Backup\inetpub\"
End If

Set Fld = FSO.GetFolder("c:\inetpub\")
Fld.Copy "e:\Backup\inetpub\"
```

CH04\BackupFolder.vbs

Manipulation des attributs

La propriété `Attributes` permet de déterminer ou de connaître les attributs des dossiers et des fichiers par les biais des objets `Folder` et `File`. Les attributs possibles sont les suivants :

Attribut	Valeur	Description
Normal	0	Aucun attribut n'est défini.
ReadOnly	1	Le fichier ou le dossier est en lecture seule.
Hidden	2	Le fichier ou le dossier est caché.
System	4	Fichier système.
Volume	8	Étiquette de volume du lecteur.
Directory	16	Spécifie qu'il s'agit d'un dossier. Attribut en lecture seule.
Archive	32	Le fichier a été modifié depuis la dernière copie de sauvegarde.
Alias	64	Le fichier est un lien ou un raccourci. Attribut en lecture seule.
Compressed	128	Le dossier ou le fichier est compressé. Attribut en lecture seule.

Cependant, si l'on tente de récupérer les attributs d'un dossier caché en lecture seule à l'aide du script suivant :

```
Dim FSO, Fld
Set FSO = CreateObject("Scripting.FileSystemObject")
Set Fld = FSO.GetFolder("c:\MonDossier")

MsgBox "Le dossier possède l'attribut " & Fld.Attributes
```

Nous obtiendrons le résultat suivant :

Or, il n'existe aucun attribut possédant la valeur 19. Ce surprenant résultat provient du fait qu'un dossier ou un fichier peut se voir attribuer plusieurs attributs différents au sein de la même propriété `Attributes`. Les différentes valeurs sont accumulées à l'aide de l'opérateur binaire `OR` et sont dépilés à l'aide de l'opérateur binaire `AND`. Cette technique n'est pas propre à la propriété `Attributes` et est utilisée à chaque fois qu'une propriété ou une variable peut se voir attribuer plusieurs valeurs.

Le résultat 19 provient donc de l'accumulation de trois valeurs binaires. Considérant le tableau suivant, si vous prenez chacune des valeurs des attributs `Directory`, `Hidden` et `ReadOnly` et que vous les accumulez à l'aide d'une opération `OR`, vous obtiendrez le résultat 19 :

Attribut	Valeur	Valeur binaire
Directory	16	0000 1000
		OR
Hidden	2	0000 0010
		OR
ReadOnly	1	0000 0001
		=
Résultat :	19	0000 1011

Maintenant, il est possible de savoir si un dossier ou un fichier est en lecture seule en dépilant le la valeur `ReadOnly` du résultat 19 à l'aide d'une opération `AND` comme suit :

Attribut	Valeur	Valeur binaire
Résultat	19	0000 1011
		AND
ReadOnly	1	0000 0001
		=
Résultat :	1	0000 0001

Ainsi, si le résultat de l'opération `AND` vaut zéro (0), cela signifie que l'attribut `ReadOnly` ne s'y retrouve pas. Sinon, le résultat sera égal à la valeur de l'attribut recherché, 1 selon notre exemple.

Voyez l'exemple complet suivant qui permet de connaître les attributs du dossier spécifié par l'utilisateur.

```
Option Explicit
Dim FSO, Fld, strDossier, strAttributs

strDossier = InputBox("Spécifiez le chemin complet du dossier pour " _
lequel vous désirez connaître les attributs:")

If strDossier <> "" Then
  Set FSO = CreateObject("Scripting.FileSystemObject")


  '*************** Vérifie que le dossier existe vraiment **********'
  If FSO.FolderExists(strDossier) Then


     '********** Obtient un objet Folder sur le dossier ************'
     Set Fld = FSO.GetFolder(strDossier)


     '**************** Récupère les attributs ********************'
     If Fld.Attributes AND 1 Then
         strAttributs = "Dossier en lecture seule" & VbCrLf
     End If

     If Fld.Attributes AND 2 Then
         strAttributs = strAttributs & "Dossier caché" & VbCrLf
     End If

     If Fld.Attributes AND 128 Then
         strAttributs = strAttributs & "Dossier compressé" & VbCrLf
     End If

     MsgBox "Le dossier possède les attributs:" & vbCrLf & strAttributs

  Else

     MsgBox "Le dossier spécifié n'existe pas", vbExclamation

  End If

End If
```

Ch04\attributs.vbs

L'objet File

L'objet File représente un fichier obtenu par la propriété `Files` de l'objet `Folder` ou la méthode `GetFile` de l'objet `FileSystemObject`. L'exemple suivant crée un objet de type `File` représentant un dossier existant sur le système :

```
Dim Fso, Fl
Set Fso = CreateObject("Scripting.FileSystemObject")
Set Fl = Fso.GetFile("c:\program files\MonDossier\MonFichier.doc")
```

Voici la liste complète des méthodes et des propriétés prévues par l'objet `File`.

Propriété	Description	Exemple
Attributes	Détermine ou retourne les attributs du fichier. Voyez les attributs possibles listés précédemment.	`Fl.Attributes = ReadOnly`
DateCreated	Retourne la date de création du fichier.	`MsgBox "Le fichier a été créé ▼ le " & Fl.DateCreated`
DateLastAccessed	Retourne la date à laquelle le fichier a été accédé pour la dernière fois.	`MsgBox "Le fichier a été accédé▼ le " & Fl.DateLastAccessed`
DateLastModified	Retourne la date à laquelle le fichier a été modifié pour la dernière fois.	`MsgBox"Le fichier a été modifié▼ le " & Fl.DateLastModified`
Drive	Retourne la lettre du lecteur contenant le fichier.	`MsgBox "Le fichier se trouve ▼ sur le lecteur " & Fl.Drive`
Name	Détermine ou retourne le nom du fichier.	`MsgBox Fl.Name`
ParentFolder	Retourne un objet `Folder` représentant le dossier parent contenant le fichier.	`MsgBox Fl.ParentFolder.Path`
Path	Retourne le chemin complet du fichier.	`MsgBox Fl.Path`
ShortName	Retourne le nom du fichier sous le format compatible 8.3.	`MsgBox Fl.ShortName`
ShortPath	Retourne le chemin du fichier sous le format compatible 8.3.	`MsgBox Fl.ShortPath`
Size	Retourne la taille en octets du fichier.	`MsgBox Fl.Size / 1024 & " Ko"`
Type	Retourne le type de fichier.	`MsgBox Fl.Type`

Méthode	Description	Exemple
Copy	Copie le fichier à l'endroit spécifié.	`Fl.Copy "c:\fichier.doc"`
Delete	Supprime le fichier.	`Fl.Delete`
Move	Déplace le fichier à la destination spécifiée.	`Fl.Move "d:\fichier.doc"`
OpenAsTextStream	Obtient un objet `TextStream` permettant d'accéder au contenu du fichier.	Voyez l'exemple complet plus loin dans la description de l'objet `TextStream`.

L'objet TextStream

L'objet `TextStream` représente le contenu d'un fichier en format texte brut. L'objet `TextStream` permet au script d'ouvrir ou de créer un fichier texte afin d'en lire ou d'en altérer le contenu. Cet objet est entre autre pratique pour permettre au script de stocker des informations de configurations ou d'historiques.

On obtient un `TextStream` à l'aide de la méthode `OpenAsTextStream` de l'objet `File` ou à l'aide de la méthode `OpenTextFile` de l'objet `FileSystemObject`. Les méthodes `OpenAsTextStream` et `OpenTextFile` attendent des paramètres optionnels permettant de spécifier le mode d'ouverture du fichier. Voici les prototypes complets correspondants à ces deux méthodes :

FileSystemObject.`OpenTextFile` (*NomFichier [, ioMode [, ForceCreate [, Unicode]]]*)

File.`OpenAsTextStream` (*[ioMode [, Format]]*)

L'argument `ioMode` représente le mode d'ouverture du fichier qui peut être l'une des trois valeurs suivantes :

ioMode	Valeur	Description
ForReading	1	Le fichier est ouvert en lecture seule.
ForWriting	2	Le fichier est ouvert en écriture. Si un fichier existant porte le même nom, sont contenu sera écrasé.
ForAppending	8	Le fichier est ouvert en ajout seulement.

L'argument `Unicode` représente la façon dont les caractères seront traités en lecture et écriture au sein de ce fichier et peut être l'une des trois valeurs suivantes :

Unicode	Valeur	Description
UseDefault	-2	Utilise les paramètres par défaut du système.
True	-1	Traite les caractères en format Wide Unicode *(16-bits)*.
False	0	Traite les caractères en format Ascii *(8-bits)*.

L'exemple suivant crée un objet de type `TextStream` en ajout seulement et traitant les caractères en format `Unicode`. Le fichier est créé s'il n'existe pas.

```
Dim Fso, Txt
Set Fso = CreateObject("Scripting.FileSystemObject")
Set Txt = Fso.OpenTextFile("c:\MonFichier.txt", 8, True, True)
```

Voici la liste complète des méthodes et des propriétés prévues par l'objet `TextStream`.

Propriété	Description	Exemple

AtEndOfLine	Retourne `True` si le curseur du fichier est situé juste avant la fin de la ligne courante.	`Do Until Fichier.AtEndOfLine`
		`  st = Fichier.Read(1)`
		`Loop`

AtEndOfStream	Retourne `True` si le curseur du fichier est situé juste avant la fin du fichier.	`Do Until Fichier.AtEndOfStream`
		`  st = Fichier.ReadLine()`
		`Loop`

Column	Retourne le numéro de colonne où se trouve le curseur du fichier.	`Do Until Fichier.Column > 25` `Loop`
Line	Retourne le numéro de ligne où se trouve le curseur du fichier.	`Do Until Fichier.Line > 25` `Loop`

Méthode	Description	Exemple
Close	Ferme le fichier précédemment ouvert.	`Fichier.Close`

Read	Lit le nombre de caractères spécifiés à la position courante du curseur et retourne la chaîne de caractères résultante.	`Do Until Fichier.AtEndOfLine`
		`  st = Fichier.Read(1)`
		`Loop`

ReadAll	Lit le contenu entier du fichier et retourne la chaîne de caractères résultante.	`st = Fichier.ReadAll()`

ReadLine	Lit une ligne entière à la position courante du curseur et retourne la chaîne de caractères qui s'y trouvait.	`Do Until Fichier.AtEndOfStream`
		`  st = Fichier.ReadLine()`
		`Loop`

Skip	Ignore le nombre spécifiés de caractères et les exclu de la lecture du fichier.	`Fichier.Skip 10`
		
		`' Saute 10 caractères`

SkipLine	Ignore une ligne et l'exclue de la lecture du fichier.	`Fichier.SkipLine`
Write	Écrit une chaîne de caractère dans le fichier. Aucun saut de ligne n'aura lieu si la chaîne spécifiée ne possède pas de caractère de saut de ligne.	`Fichier.Write "allô" & vbCrLf`

WriteBlankLines	Insère dans le fichier le nombre de sauts de ligne spécifiés.	`Fichier.WriteBlankLines 10`
		
		`'Inscrit 10 lignes vides`

WriteLine	Écrit une chaîne de caractère dans le fichier. Un saut de ligne est automatiquement inséré après l'écriture de la chaîne de caractères.	`Fichier.WriteLine "allô"`

Le processus d'accès à un fichier est sensiblement le même dans tous les cas.

- Le fichier doit d'abord être ouvert à l'aide d'une des méthodes `OpenTextFile` ou `OpenAsTextStream`.
- Le fichier peut être lu ou écrit à l'aide des méthodes `Read` et `Write` nommées ci-haut.
- Le fichier est refermé à l'aide de la méthode `Close` expliquée ci-haut.

L'exemple suivant récupère des informations auprès de l'utilisateur avant de stocker celles-ci au sein d'un fichier. Remarquez comment le script détecte d'abord si le fichier existe ou non afin d'afficher le contenu actuel du fichier.

```
Dim FSO, Txt, Fichier, stNom, stPrenom
Set FSO = CreateObject("Scripting.FileSystemObject")

'****************** Chemin du fichier *.ini  ***********************'
Fichier = WScript.ScriptFullName
Fichier = Left(Fichier, InStrRev(Fichier, "\")) & "\infos.ini"

If FSO.FileExists(Fichier) Then
  Set Txt = FSO.OpenTextFile(Fichier, 1)
  stNom = Txt.ReadLine()
  stPrenom = Txt.ReadLine()
  Txt.Close

  MsgBox "Bonjour " & stPrenom & " " & stNom

  If MsgBox("Désirez-vous spécifier de nouvelles informations?", _
     vbQuestion OR vbYesNo) = vbNo Then
     WScript.Quit
  End If
End If

'********************  Réinitialise les variables  ******************'
stNom = ""
stPrenom = ""

'**********  Récupère les informations de l'utilisateur *************'
Do While stNom = ""
 stNom = InputBox("Entrez votre nom:")
 If stNom = "" Then
   If MsgBox("Quitter le script?", vbQuestion OR vbyesNo) = vbYes Then
      WScript.Quit
   End If
 End If
Loop

Do While stPrenom = ""
 stPrenom = InputBox("Entrez votre prénom:")
 If stPrenom = "" Then
   If MsgBox("Quitter le script?", vbQuestion OR vbyesNo) = vbYes Then
      WScript.Quit
   End If
 End If
Loop

'************* Inscrit les informations dans un fichier *************'
Set Txt = FSO.OpenTextFile(Fichier, 2, True)
Txt.WriteLine stNom
Txt.WriteLine stPrenom
Txt.Close
```

CH04\FichierInfos.vbs

Afin de clore ce survol des objets `WSH` et `FileSystemObject`, voici un exemple complexe permettant de lancer la recherche d'un fichier sur les disques du système. L'exemple possède l'avantage de récupérer le nom du fichier recherché au sein des arguments de la ligne de commande ou, s'il ne s'y trouve pas, récupère l'information à l'aide d'une boîte de saisie.

Ensuite, le script utilise les objets FSO afin de parcourir astucieusement la totalité des lecteurs et des répertoires s'y trouvant afin de trouver le fichier recherché et toutes ses occurrences possibles.

Finalement, le script ouvre un fichier texte afin d'y stocker le résultat de la recherche et en demande l'affichage avant de quitter.

```
Option Explicit

Dim Fso, Drv, K, Filename, Txt, stResultat, Fichier
Set FSO = CreateObject("Scripting.FileSystemObject")


'***********  Récupère le nom de fichier à rechercher  **************'
On Error Resume Next
Filename = WScript.Arguments(0)


'*******  Si aucun argument spécifié à la ligne de commande *********'
If Filename = "" Then
   Filename = InputBox("Entrez le nom du fichier à rechercher:")

   If Filename = "" Then
       WScript.Quit
   End If

End If


'**********  Parcours les lecteurs du système de A-Z  ***************'
For K = Asc("a") To Asc("z")
  If FSO.DriveExists(Chr(K)) Then

    Set Drv = FSO.GetDrive(chr(k) & ":\")
    If Drv.IsReady Then
       FindFile Filename, FSO.GetFolder(chr(k) & ":\")
    End If

  End If
Next
```

Le code se poursuit à la page suivante

```
'***********  Inscrit les résultats dans un fichier texte ***********'
Fichier = WScript.ScriptFullName
Fichier = Left(Fichier, InStrRev(Fichier, "\")) & "resultats.txt"


'*********************  Ouverture du fichier  ***********************'
Set Txt = FSO.OpenTextFile(Fichier, 2, True)

Txt.WriteLine "Résultat de la recherche du fichier '" & Filename & "'"
If Trim(stResultat) <> "" Then
  Txt.Write "Le fichier a été trouvé:" & vbCrLf & vbCrLf & stResultat
Else
  Txt.WriteLine "Le fichier n'a pu être trouvé. Il se peut qu'il se " _
  & "trouve dans un répertoire à lequel vous ne possédez pas d'accès."
End If

'***********************  Fermeture du fichier **********************'
Txt.Close

'************************ Démarrage du fichier **********************'
CreateObject("WScript.Shell").Run "notepad " & Fichier




'*********************************************************************'
'* FINDFILE                                                          *'
'*********************************************************************'
Function FindFile( ByVal Filename, ByVal Fld)
  Dim Fl, stFichier

  '***********  S'assure d'avoir un nom de fichier valide  **********'
  If Right(Fld.Path, 1) = "\" Then
    stFichier = Fld.Path & Filename
  Else
    stFichier = Fld.Path & "\" & Filename
  End If


  '***************** Teste si le fichier existe *********************'
  If FSO.FileExists(stFichier) Then
     stResultat = stResultat & " * " & stFichier & vbCrLf
     Exit Function
  End If


  '****************** Récursivité sur les sous-répertoires ************'
  For Each Fl In Fld.SubFolders
    FindFile Filename, Fl
  Next

End Function
```

CH04\Recherche.vbs