Un livre de Wikilivres.

Programmation VBScript

Introduction

Le langage **VBScript** ou **VBS** (pour Visual Basic Script) est un langage de programmation de la famille (visual) basic, destiné à être interprété. Il est dépendant du logiciel *Windows Script Host* (WSH) exclusivement fourni par Microsoft.

On donne aux scripts logiciels écrits dans ce langage, par convention, un nom qui se termine par .vbs. Le fait que le langage est un langage de script exécuté par un interpréteur signifie que le programme est exécuté sans compilation manuelle en fichier binaire. Le fait que l'interpréteur soit associé au suffixe vbs dans Windows rend le fichier directement exécutable sur double-clic depuis l'explorateur de Windows.

Comme le logiciel supporte plusieurs langages de programmation, il est utile d'indiquer sur une ligne le langage de programmation utilisé. Configuration HTML :

- Sur client : < script langage="VBScript" />
- Sur serveur : <% script langage="VBScript" %>

Opérateurs

+	addition
-	soustraction
*	multiplication
^	puissance
/	division à float
\	division entière
mod	modulo - reste de la division entière
<	plus petit
<=	plus petit ou égal
>	plus grand
>=	plus grand ou égal
<>	différent
not	non logique
and	et logique
or	ou logique
xor	xor logique
eqv	equivalence
imp	implication
&	concaténation

Déclarations

```
Contrairement à beaucoup d'autres langages, le vbs n'est pas sensible à la casse.
```

- LÉGENDE

  src - source
  nm - nom
  var - variable
  expr - expression
  ins - instruction
  arg - argument
  cond - condition

- Déclaration

  option explicit - *force déclaration de toutes variables dans scr*

  dim nm(src) - *variable*

  redim [preserve] var(src)

  const nm=expr - *constante*

- Procédure - *execute instructions*

```
sub nm(arg1,arg2,...)
    [ins]
    [exit sub]
    [ins]
end sub
```

- Fonction - *retourne une valeur en sortie à nm*

```
function nm(arg1,arg2,...)
    [ins]
    [nm=expr]
    [exit function]
    [ins]
    [nm=expr]
end function
```

- Get - *comme fonction simple*

```
property get nm(arg1,arg2,...) as [type]
    ...
    [ins]
    [nm=expr]
end property
```

- Let - *comme fonction simple*

```
property let nm(arg1,arg2,...)
    ...
    [ins]
end property
```

- Set - comme fonction simple

```
property set nm(arg1,arg2,...)
    ...
    [ins]
end property
```

- Classe - *retourne initialize || terminate*

```
class ClassName
        [var]
    Private Sub Class_Initialize()
        [ins]
    End Sub
    Private Sub Class_Terminate()
        [ins]
    End Sub
    Public Function NomMethode([arg])
        [ins]
    End Function
end Class
```

---

nota : const/sub/function/property sont déclarables en [public [default] | private]

ex 1
public function nm(args) - pour toute la src
ex 2
sub nm(args)
private const=expr - *n'est déclarée et n'est valable que dans la procédure*
end sub

Instructions de contrôle

- TEST

```
If cond Then [ins] Else [ins]
```

```
If cond1 Then
 [ins]
Elseif cond2 Then
 [ins]
Else
 [ins]
Endif
```

```
Select Case expression
 Case expr1
  [ins]
 Case expr2
  [ins]
 Case Else
  [ins]
End Select
```

- BOUCLE

```
For cpt = début To fin [Step pas]
 [ins]
 [Exit For]
 [ins]
Next [cpt]
```

```
For Each élément In groupe
 [ins]
 [Exit For]
 [ins]
Next
```

```
Do [{While | Until} condition]
 [ins]
 [Exit Do]
 [ins]
Loop
```

```
Do
 [ins]
 [Exit Do]
 [ins]
Loop [{While | Until} condition]
```

```
While condition
 [ins]
Wend
```

- APPELS

```
With obj
 [ins]
End Width
```

```
Call nom [args]
```

- ERREURS

```
On Error Resume Next
On Error Goto 0
```

- REMARQUES

```
Rem
```

- AFFECTATION DE REFERENCES

```
Set objvar=[obj expr | New classname | Nothing ]
```

- ASSOCIATION DE REFERENCE

```
Set object.eventname=GetRef(procname)
```

Fonctions

conversion et mise en forme

- CBool(expr) - *conversion en variant booleen*
- CByte(expr) - *conversion en variant byte*
- CCur(expr) - *conversion en variant currency*
- CDate(expr) - *conversion en variant date*

- CDbl(expr) - *conversion en variant double*
- CInt(expr) - *conversion en variant integer*
- CLng(expr) - *conversion en variant long*
- CSng(expr) - *conversion en variant single*
- CStr(expr) - *conversion en variant string*
- FormatCurrency(expr,[NumDigitsAfterDecimal],[IncludeLeadingDigit],[UseParensForNegativeNumber], [GroupDigits]) - *conversion formatée en variant currency*
- FormatDateTime(date,[NamedFormat]) - *conversion formatée en variant date*
- FormatNumber(expr,[NumDigitsAfterDecimal],[IncludeLeadingDigit],[UseParensForNegativeNumber], [GroupDigits]) - *conversion formatée en variant nombre*
- FormatPercent(expr,[NumDigitsAfterDecimal],[IncludeLeadingDigit],[UseParensForNegativeNumber], [GroupDigits]) - *conversion formatée en variant percent*
- Hex(nbr) - *conversion en variant hexadecimal*
- IsArray(expr) - *renvoit booleen si tableau*
- IsDate(expr) - *renvoit booleen si date*
- IsEmpty(expr) - *renvoit booleen si vide*
- IsNull(expr) - *renvoit booleen si null*
- IsNumeric(expr) - *renvoit booleen si nbr*
- IsObject(identifier) - *renvoit booleen si obj automation*
- Oct(nbr) - *conversion en string octal*

mathématiques

- Abs(nbr) - *renvoie valeur absolue*
- Atn(nbr) - *renvoie arc tangent*
- Cos(nbr) - *renvoie cosinus*
- Exp(nbr) - *renvoie la puissance*
- Int(nbr) - *renvoie partie entière*
- Fix(nbr) - *renvoie partie entière*
- Log(nbr) - *renvoie log népérien*
- Sqr(nbr) - *renvoie racine carrée*
- Tan(nbr) - *renvoie tangente*
- Sgn(nbr) - *renvoie signe*
- Randomize(nbr) - *init nbr aleatoire !!les nombres ne sont pas aleatoire en vbs!!*
- Rnd(nbr) - *renvoie nbr aleatoire*
- Round(expr,[numdecimalplaces]) - *renvoie l'arrondi*
- Sin(nbr) - *renvoie sinus*

de traitement de chaînes

- Asc(string) - *renvoie code ansi du 1er char*
- Chr(charcode) - *renvoie le char du code ansi*
- inStr(start,string,string2,[compare]) - *renvoie 1ère position d'un string dans un autre*
- InStrRev(delimiter) - *renvoie position d'un string dans un autre en partant de fin*
- Join(string) - *renvoie string joins partant d'un tableau*
- LCase(string) - *conversion en minuscule*
- Left(string,length) - *renvoie sous string partant de gauche*
- Len(string) - *renvoie nbr de char dans string*
- LTrim(string) - *renvoie string sans espace à gauche*
- Mid(string,start,[length]) - *renvoie sous string partant de milieu*

- Replace(expr,find,replacement,[start],[count],[compare]) - *remplace string dans string*
- Right(string,length) - *renvoie sous string partant de droite*
- RTrim(string) - *renvoie string sans espace à droite*
- StrComp(string1,string2,[compare]) - *renvoie valeur de comparaison de string*
- String([nbr],[char]) - *renvoie string de char repete*
- StrReverse(string) - *renvoie string inverse*
- Space(nbr) - *renvoie string de space*
- Trim(string) - *renvoie string sans espace ni gauche ni droite*
- UCase(string) - *renvoie string en majuscule*

de date et d'heure

- Date() - *renvoie date sys*
- DateAdd(interval,nbr,Date) - *renvoie date + interval*. Ex : `DateAdd("d", -1, now)` affiche la date de la veille.
- DateDiff(interval,date1,date2,[firstdayofweek],[firstdayofyear]) - *renvoie interval*
- DatePart(interval,Date,[firstdayofweek],[firstdayofyear]) - *renvoie partie spécifiée*
- DateSerial(Year,Month,Date) - *renvoie variant de type date*
- DateValue(Date) - *renvoie variant de type date*
- Day(Date) - *renvoie jour du mois*
- Hour(Time) - *renvoie heure*
- Minute(Time) - *renvoie minute'*
- Month(Date) - *renvoie mois en chiffre*
- MonthName(Month,[abbreviate]) - *renvoie mois en lettre*
- Now() - *renvoie date et heure sys*
- Second(Time) - *renvoie seconde*
- Time() - *renvoie heure sys*
- Timer() - *renvoie nbre de sec depuis 00:00*
- TimeSerial(Hour,Minute,Second) - *renvoit variant de type date*
- TimeValue(Time) - *renvoit variant de type date*
- Weekday(Date,[fistdayofweek]) - *renvoie jour semaine en chiffre*
- WeekdayName(Weekday,abbreviate,firstdayofweek) - *renvoie jour semaine en lettre*
- Year(Date) - *renvoie annee de date*

liées aux tableaux

- Array(arg) - *renvoie variant contenant tableau*
- Erase tableau - *libère tableau*
- Filter(InputStrings,Value,[Include],[compare]) - *renvoie tableau commençant par 0 contenant sous ensemble de string*
- LBound(arrayname,[dimension]) - *renvoie plus petit indice tableau*
- Split(expr,[delimiter],[count],[compare]) - *renvoie tableau commençant par 0 comprenant # de string*
- UBound(arrayname,[dimension]) - *renvoie plus grand indice tableau*

Autres fonctions

- CreateObject(class) - *renvoie ref à objet automation*
- Eval(string) - *renvoie resultat*
- Execute ins - *execute ins*
- ExecuteGlobal ins - *execute ins globale*

- GetLocale() - *renvoie valeur ID parametre regionnaux sys*
- GetObject([pathname],[class]) - *renvoie ref à objet automation de fichier*
- GetRef(string) - *renvoie ref à procedure*
- InputBox(prompt,[title],[default],[xpos],[ypos],[helpfile],[context]) - *prompt*
- LoadPicture([stringexpr) - *renvoie img*
- MsgBox(prompt,[buttons],[title],[helpfile],[context]) - *msgbox*
- RGB(red,green,blue) - *renvoie int rgb*
- ScriptEngine() - *renvoie string de script utilisé*
- ScriptEngineBuildVersion() - *renvoie N° version*
- ScriptEngineMajorVersion() - *renvoie N° version principale*
- ScriptEngineMinorVersion() - *renvoie N° version secondaire*
- SetLocate(Icid) - *def param regionnaux globaux*
- TypeName(varname) - *renvoit string d'info sur variant*
- VarType(varname) - *renvoit valeur sous type de var*

Pour déclarer ses propres fonctions, voir le chapitre Application.

Objets

- Err - *infos relatives aux erreurs d'exec*
  - Proprietes

    Description
    Helpcontext
    Helpfile
    Number
    Source

  - Methodes

    Clear()
    Raise(nbr,src,desc,hlpf,hlpcontext)

- RegExp - *gestion des regular expr*
  - Proprietes

    Global
    IgnoreCase
    Pattern

  - Methodes

    Execute(string)
    Replace(string1,string2)
    Test(string)

- Match - *accès aux proprietes correspondant de regular expr*
  - Proprietes

    FirstIndex
    Length
    Value

- Matches - *collection de match*
  - Proprietes

    Count
    Item(key)

- SubMatches - *collection de sous-match*
  - Proprietes

    Count
    Item(key)

Constantes

CONSTANTE	VALEUR	DESCRIPTION
Chaînes de caractères		
vbCr	chr(13)	Retour chariot
vbCrlf	chr(13)+chr(10)	Retour chariot et saut de ligne
vbFormFeed	chr(12)	Saut de page
vblf	chr(10)	Saut de ligne
vbNewline	chr(13)+chr(10)	Nouvelle ligne
vbNullChar	chr(0)	0
vbNullString	chaîne ayant val. 0	chaine nulle
vbTab	chr(9)	tab horizontale
vbVerticalTab	chr(11)	tab verticale
Couleur		
vbBlack	&h000000	Noir
vbRed	&h0000FF	Rouge
vbGreen	&h00FF00	Vert
vbYellow	&h00FFFF	Jaune
vbBlue	&hFF0000	Bleu
vbMagenta	&hFF00FF	Magenta
vbCyan	&hFFFF00	Cyan
vbWhite	&hFFFFFF	Blanc
Comparaison		
vbBinaryCompare	0	Comparaison binaire
vbTextCompare	1	Comparaison texte
Date et heure		
vbSunday	1	Dimanche
vbMonday	2	Lundi
vbTuesday	3	Mardi
vbWednesday	4	Mercredi
vbThursday	5	Jeudi
vbFriday	6	Vendredi
vbSaturday	7	Samedi
vbUseSystem	0	Format de date des param regionaux
vbUseSystemDayOfWeek	0	Jour semaine des param regionaux
vbFirstJan1	1	Utilise la semaine du 01 Janvier
vbFisrtFourDays	2	Première semaine avec 4 jours dans nouvelle annee
vbFirstFullWeek	3	Utilise la premiere semaine complete de l'annee
Format de date		

vbGeneralDate	0	Date et heure
vbLongDate	1	Date complète
vbShortDate	2	Date abrégée
vbLongTime	3	Heure complète
vbShortTime	4	Heure abrégée
MsgBox		
vbOkOnly	0	ok
vbOkCancel	1	ok/cancel
vbAbortRetryIgnore	2	abort/retry/ignore
vbYesNoCancel	3	yes/no/cancel
vbYesNo	4	yes/no
vbRetryCancel	5	retry/cancel
vbCritical	16	message critique
vbQuestion	32	demande avertissement
vbExclamation	48	exclamation
vbInformation	64	information
vbDefaultButton1	0	bouton1
vbDefaultButton2	256	bouton2
vbDefaultButton3	542	bouton3
vbDefaultButton4	768	bouton4
vbApplicationModal	0	boite modale pour l'app
vbSystemModal	4096	boite modale pour le sys
VarType		
vbEmpty	0	non init (defaut)
vbNull	1	pas de data
vbInteger	2	sous-type integer
vbLong	3	sous-type long
vbSingle	4	sous-type single
vbDouble	5	sous-type double
vbCurrency	6	sous-type currency
vbDate	7	sous-type date
vbString	8	sous-type string
vbObject	9	objet
vbError	10	sous-type error
vbBoolean	11	sous-type booléen
vbVariant	12	variant pour tableau variant
vbDataObject	13	objet d’accès aux données

vbDecimal	14	sous-type decimal
vbByte	17	sous-type byte
vbArray	8192	tableau
Autres constantes		
vbObjectError	-2147221504	num d'erreur doivent être supérieur
vbUseDefault	-2	utiliser la valeur par défaut du paramètre
vbTrue	-1	true
vbFalse	0	false

Codes d'erreur

Il y a deux types d'erreurs :

- Syntaxe
- Exécution

Exécution

NUMERO	DESCRIPTION
5	appel de procédure incorrect
6	depassement de capacite
7	memoire insuffisante
9	indice hors plage
10	tableau fixe ou verrouille
11	division par zero
13	type incompatible
14	espace de chaine insuffisant
17	operation impossible
28	espace pile insuffisant
35	sub non definie
48	erreur chargement de dll
51	erreur interne
91	var d'objet non definie
92	boucle non init
94	utilisation incorrecte de null
424	objet requis
429	activex ne peut pas creer l'objet
430	classe ne gere pas automation
432	fichier ou classe introuvable
438	methode ou propriete non gere

445	objet ne gere pas action
447	objet ne gere pas les param
448	arg introuvable
449	arg obligatoire
450	nbre arg ou propriete incorrecte
451	objet n'est pas une collection
458	type automation non gere
462	machine distante indisponnible
481	image incorrecte
500	variable indefinie
502	objet non securise pour script
503	objet non securise pour init
504	objet non securise pour creation
505	reference incorrecte ou non qualifiee
506	classe non definie
507	exception
5008	affectation illegale
5017	Erreur syntaxique dans ER
5018	quantifiant innatendu
5019	']' attendu dans ER
5020	')' attendu dans ER
5021	jeu de char incorrect

Syntaxe

NUMERO	DESCRIPTION
1052	trop de méthodes et propriétés
1044	parentheses interdites
1053	classe sans arg
1058	'default' que dans get
1057	'default' + 'public'
1005	'(' attendu
1006	')' attendu
1011	'=' attendu
1021	'case' attendu
1047	'class' attendu
1025	fin ins attendu
1014	'end' attendu
1023	expr attendu
1015	'function' attendu
1010	identificateur attendu
1012	'lf' attendu

1046	'ln' attendu
1026	constante (int) attendu
1049	let, get, set attendu
1045	constante (string) attendu
1019	'loop' attendu
1020	'next' attendu
1050	'property' attendu
1022	'select' attendu
1024	ins attendu
1016	'sub' attendu
1017	'then' attendu
1013	'to' attendu
1018	'wend' attendu
1027	'while' 'until' attendu
1028	'while' 'until' ou fin ins attendu
1029	'with' attendu
1030	identificateur trop long
1014	char incorrect
1039	'exit' incorrect
1040	variable de ctrl de 'for' incorrecte
1013	nombre incorrect
1037	utilisation incorrecte de 'me'
1038	'loop' sans 'do'
1048	doit être défini dans classe
1042	doit premiere ins de ligne
1041	nom redéfini
1051	nombre arg inegal
1001	mémoire insuffisante
1054	arg manquant dans let ou set
1002	erreur syntaxe
1055	next attendu
1015	constante de chaine non terminée

Application

Pour commencer, VBS comme tous les langages d'édition nécessite un éditeur de texte. Il doit être enregistré avec l'extension .vbs pour être reconnu par MIME et interprété par wscript.exe ou cscript.exe qui se trouvent dans le répertoire %system% de votre OS Windows.

1. Pour écrire un .vbs, il faut au minimum un éditeur de texte comme notepad.exe et sauver le fichier en NomProgramme.vbs en mettant Type à "Tous les fichiers".
2. Pour avoir une description complète des fonctions et de leurs paramètres, consulter le livre VBScript

et ses liens externes.

Qu'est ce qu'un programme

Un programme est une suite d'instructions qui comporte des **traitements** sur des **données**.

- Légende des abréviations utilisées ici :

cond	condition
arg	argument
ins	instruction
var	variable
cmt	commentaire
expr	expression
var	variable
res	resultat
str	string
>	Sortie
<	Entree

Paradigme de programmation

Un programme fonctionne en terme d'entrée de données et de sortie de données. Les fonctions reçoivent des données (entrée de données) et les renvoient (sortie de données). Ces entrées et sorties (E/S) permettent à un programme d'être rédigé comme une pile d'instructions ou pipeline traitant/renvoyant des données. Pour concevoir un programme, il faut d'abord vous demander ce que vous voulez faire, c'est à dire le résultat à produire.

Les résultats peuvent être transmis à d'autres (sous)programmes.

- **Qu'est-ce qu'une donnée ?**

Une donnée est une valeur.

- **Qu'est-ce qu'une expression ?**

Une expression est un TERME comprenant des opérations sur données et qui produisent un résultat.

- **Qu'est-ce qu'un résultat ?**

Un résultat est une nouvelle donnée.

- **Qu'est-ce qu'une instruction ?**

Une instruction est une pile d'opérations sur données.

- **Qu'est-ce qu'une variable ?**

Une variable est une zone temporaire recevant une donnée. Elle doit commencer par une lettre. Une var CONSTANTE est une cellule de donnée permanente pour toute la durée du programme.

- **Qu'est-ce qu'une condition ?**

Une condition est une expression à TESTER.

- **Qu'est-ce qu'un "type" ?**

Un type est une façon d'interpréter une donnée. Il y a les nombres, les caractères, les chaînes de caractère, les tableaux...

Notions élémentaires

- Faire un commentaire

Un commentaire est une ligne du programme qui commence par ' et qui indique au programme qu'il doit la sauter car ne faisant pas partie des [ins].

```
'ceci est un commentaire ...
```

- Afficher un string

```
msgbox "bonjour tout le monde"

'>Bonjour tout le monde
```

- Initialiser une variable <

Il n'est pas encore utile de définir le type de variable, par défaut entre "" c'est du [str] sinon c'est du [nbr].

```
msg="salut à tous"
msgbox msg

'>salut à tous
```

- Introduire une donnée au clavier > sortie

```
msg=inputbox(msg)
msgbox msg

'>la valeur que vous avez tapée
```

La var msg reçoit la saisie clavier et la sort.

- Faire un programme qui calcule toutes les opérations sur 2 [nbr] <

```
'opération sur 2 <
nbr1=inputbox(msg,"introduisez le nbr1")
nbr2=inputbox(msg,"introduisez le nbr2")
MsgBox "addition:"&nbr1+nbr2&VbCrLf&"multiplication:"&nbr1*nbr2&VbCrLf _
&"division:"&nbr1/nbr2&VbCrLf&"soustraction:"&nbr1-nbr2,vbokonly,"OPERATIONS"

'>addition:res
```

```
'>multiplication:res
'>division:res
'>soustraction:res
```

! Dans ce programme, msgbox retourne un string. Ici cette chaîne est constituée par [str]& [expr]&VbCrLf&... VbCrLf passe à une nouvelle ligne et fait un retour chariot. _ permet de couper une [expr] en deux lignes.

- Faire une boucle qui calcule

Une boucle doit TOUJOURS avoir une condition d'arrêt sans quoi elle est infinie.

```
'BOUCLE qui calcule et >
nbr1=1
nbr2=1
For cpt = 1 To 15 Step 1
 nbr1=nbr1+1
 nbr2=nbr1 mod cpt
 somme=somme&(nbr2/nbr1)&vbCrlf
 MsgBox somme
Next
```

! les nbr sont initialisés avant de rentrer dans la boucle qui compte de 1 à 15 en rajoutant 1 à chaque tour de boucle.

- Même boucle avec [cond]

```
'BOUCLE qui calcule et >
cpt=0 '=== toujours initialiser une variable de compteur à zéro
Do Until cpt=11
 nbr1=nbr1+1
 nbr2=nbr1+1
 somme=somme&(nbr2/nbr1)&vbCrlf
 MsgBox somme
 cpt=cpt+1
Loop
```

! Ici il a fallu incrémenter de 1 cpt dans cette boucle qui continue jusqu'à ce que cpt=15. Par défaut une var=0=null.

- Test booléen (#res={vrai,faux})

```
msg=InputBox (msg,"Tapez 1")
If msg=1 Then MsgBox "vrai",,"vous avez tapé 1 ?" Else MsgBox "faux",,"vous avez tapé 1 ?
```

Si vous tapez 1 dans le prompt c'est >vrai. Pour avoir plusieurs instructions il faut adopter le format suivant :

```
msg=InputBox (msg,"Tapez 1")
If msg=1 Then
 res="vrai"
Else
 res="faux"
End if
MsgBox Resultat,,"vous avez tapé 1 ?"
```

! Ici [res] est la var pour la transmettre > msgbox il ne faut pas la mettre entre "" comme pour les str>[var].

- Test non-booléen (#[res]={[res1],[res2],...})

```
msg=InputBox (msg,"Tapez 1 ou 2 ou 3")
Select Case msg
 Case 1
  MsgBox "un"
 Case 2
  MsgBox "deux"
 Case 3
  MsgBox "trois"
 Case Else
  MsgBox "autre"
End Select
```

! Après le test de msg, il y a plusieurs cas possibles de [res]. Selon que on a entré 1 ou autre chose on est pipé -> vers le Case correspondant au [res] de la [var] msg.

Type de variable

- VAR

Une var sous vbs est en mode auto, ainsi lorsque l'on affecte une valeur ou [expr] à [], le [str] est typé en fonction de son contenu ce qui n'est pas toujours vrai en visual basic.

```
'déclaration
Dim var1,var2,var3
var1=10.3
var2="il etait une fois"
var3=var1&var2
```

! On ne peut effectuer d'opération sur var1 et var2 en var3 car elle ne sont pas de même type. Seul la concaténation fonctionne ici.

- CONST

Une constante est une valeur qui ne change pas.

```
Const1=15
Const2=30
Const3=Const1+Const2
```

- ARRAY

Un tableau est une var contenant n cellule []. On peut y stocker des [str]. Un array se déclare comme une var et reçoit des valeurs. Chaque cellule a un indice, ainsi la cellule n°n a l'indice n.

```
dim MonTableau()
'on definit le nombre de [] de MonTableau
Redim MonTableau(10)
For cpt=0 to 9
 MonTableau(cpt)="Cellule_"&cpt
Next
```

```
MsgBox MonTableau(1)

'>Cellule_1
```

! La première cellule a l'indice 0. MonTableau(1) ou la deuxième cellule de MonTableau aura donc pour [str]="Cellule_1".

Fonctions et procédures

A partir d'ici, pour une meilleure écriture et lisibilité des programmes, il est conseillé de séparer les parties déclaration et instruction :

[var]
[ins]

Les procédures et les fonctions constituent des sous programmes permettant de définir une pile d'[ins] qui ont pour point d'entrée le nom. Pour les exécuter il suffit d'appeler la méthode Call Nom([arg]) ou simplement Nom([]) si elle ne comporte pas d'[arg].Les [arg]/datas sont à passer en paramètre selon leur définition.

Les fonctions et procédures sont publiques (Public) par défaut, c'est-à-dire disponible dans l'ensemble d'une application VBScript. Public Default est utilisé pour indiquer que la fonction ou la procédure est utilisée par défaut dans une classe. Private signifie que la fonction ou la procédure ne sera valable que dans l'espace du nom global qui l'encapsule, par exemple une classe ou le script en cours.

Pour déclarer leur nom, attention de ne pas prendre un nom déjà attribué. Voir librairie VBScript.

Procédures

Sub permet de déclarer nom [args] et [ins] formant la procédure. Son but est d'exécuter des [ins].

[Public [Default] | Private] Sub Nom_Procédure [(arg1,arg2,...)]
    [ins]
End Sub

- Sans arguments

```
'Afficher l'heure système
Sub heure()
 MsgBox Now()
End Sub
heure()
```

On déclare ici la procédure heure() - sans [arg] - qui > now() - procédure affichant heure sys. On appelle ensuite l'exécution de cette procédure.

- Avec arguments

```
'calc + et >
Sub adi(a,b)
 Dim somme
 somme=a+b
```

```
 MsgBox somme
End Sub
Call adi(15,20)

'>35
```

Vous remarquez ici que Dim déclare [somme] en tête du sous-programme adi qui est appelé avec deux args/paramètres définis par adi(a,b).

Fonctions

Function permet de déclarer nom [args] et [ins] formant la fonction. Son but est de retourner une valeur. Sinon, autant utiliser une procédure.

> [Public [Default] | Private] Function Nom_Fonction [(arg1,arg2,...)]
> [ins]
> Nom_fonction=[expr]
> End Function

La fonction à Nom_Fonction comme point d'entrée () et retourne Nom_Fonction=[expr] en sortie(). Vous comprenez maintenant ? Un programme c'est un pipeline, les [res] ne font que passer d'un bout à l'autre jusqu'au terminus, l'écran ou un fichier ou autre :)

! Il est impossible de créer Function à l'intérieur de Sub et réciproquement.

- Sans arg

```
Function Addition1()
 a=10
 b=15
 Addition1=a+b
End Function
MsgBox Addition1()
```

- Avec arg

```
Function Addition(val1, val2)
  Addition = val1 + val2
End Function
MsgBox Addition(10,15)

'>25
```

- Autre exemple

```
'calc (a-b)/c >
Function calc(a,b,c)
 calc=(a-b)/Round(c)
End Function
MsgBox calc(15,10,2.5)&vbcrlf&Abs(calc(10,20,30))
```

! Dans calc() Round retourne directement l'arrondi. A la ligne 5, le pipe s'effectue de la plus petite parenthèse à l'intérieur, ici calc(10,20,30) pour retourner sa valeur a Abs() qui retourne sa valeur en même

temps que calc(15,10,2.5) > [str] > MsgBox qui affiche [str]. C'est simple comme 'Hello World'.

Classe d'objets

En POO, une classe encapsule des données et [ins] définissant des propriétés [var] et méthodes [ins]. Les classes permettent de construire des objets types à réutiliser. L'instanciation consiste à créer un nouvel objet de classe. Un objet est donc une instance de classe.

```
Class NomDeClasse
    [var]
    Private Sub Class_Initialize()
        [ins]
    End Sub
    Private Sub Class_Terminate()
        [ins]
    End Sub
    Public Function NomMethode([arg])
        [ins]
    End Function
End Class
```

- Application

```
'------------------------------------
'- ouvre une classe et crée tableau -
'- place des articles dedans -
'- affiche l'article par numéro -
'------------------------------------
On Error Resume Next
  'déclaration de la classe
  class strArticle
    Private index, article()

    'Initialise la classe
    Private Sub Class_Initialize()
      'définit 50 champs ds tableau
      redim article(50)
      index=0
    End Sub

    'Termine la classe
    Private Sub Class_Terminate()
      erase article
    End Sub

    'public car devant etre accessible hors classe
    public default Sub ajout(Val)
      article(index)=Val
      index=index+1
      if index>50 then
        MsgBox "plus de place dans tableau"
      end if
    end sub

    'public car devant etre accessible hors classe
    public Function Affiche(idx)
      MsgBox article(idx)
```

```
    end function
  end class

'initialisation d'une nouvelle classe
Set cat= new strArticle
'place les article dans le tableau
For cpt = 1 to 50
 cat.ajout("ARTICLE_"&cpt)
Next

msg=0
While (IsNumeric(msg))
 msg=InputBox(msg,"Choisissez l'article à sortir / [!0-9] pour sortir")
 cat.Affiche(msg-1)
Wend

Set cat=nothing
```

! On initialise et on termine une classe pour libérer la ressource apres usage. On declare plusieurs fonctions qui sont les methodes à appeler. Les Init et Termin sont en private car utile que dans la classe, en revanche toute autre function ou sub doit être public pour l'E/S hors classe.

Pour utiliser une classe :

- on initialise d'abord l'objet obj= new NomDeClass;
- on appele la methode dans l'objet obj.NomDeFonction([arg]);
- comme dans fonction ordinaire [arg] [var] [expr] < en parametre ;
- Voir cet article (http://jerome.developpez.com/tutos/classesvbscript/) [archive]

E/S fichier

Le programme suivant crée un fichier, en supprimer un autre, puis copie le premier[1],[2] :

```
dim filesys
Set filesys = CreateObject("Scripting.FileSystemObject")

filesys.CreateTextFile "c:\NouveauFichier.txt", True
If filesys.FileExists("c:\AncienFichier.txt") Then
   filesys.DeleteFile "c:\AncienFichier.txt"
   Response.Write("Le fichier a été supprimé.")
End If
filesys.CopyFile "c:\NouveauFichier.txt", "c:\NouvelAncienFichier.txt"  ' Ecrase si exist
```

VBS peut aussi faire plus en utilisant WSH (Windows Script Host)[3].

Pour lire et écrire dans un fichier, utiliser la méthode *OpenTextFile()* du *FileSystemObject*[4].

Bases de données

Pour encapsuler du SQL dans du VBS pour Microsoft SQL Server, créer une connexion ADOdb[5] :

```
Set cnx = CreateObject("ADODB.Connection")
```

```
cnx.ConnectionString = "Provider=SQLOLEDB.1; Server=NomDuServeur; Database=NomDeLaBase; U
cnx.Open
Requete = "select Nom, Prenom from Utilisateur"
set recordset = cnx.execute(Requete)
recordset.MoveFirst
msgbox recordset.fields("Prenom") & " " recordset.fields("Nom")
cnx.Close
```

Théoriquement plusieurs SGBD peuvent être manipulés en changeant le paramètre *Provider* (ou driver[6]).

Références

1. http://www.devguru.com/technologies/vbscript/quickref/filesystemobject_deletefile.html
2. http://msdn.microsoft.com/en-us/library/e1wf9e7w%28v=vs.85%29.aspx
3. http://www.computing.net/answers/programming/vbs-set-subdirectories-files-too/9928.html
4. http://msdn.microsoft.com/en-us/library/t5399c99%28v=vs.84%29.aspx
5. http://stackoverflow.com/questions/11143764/vbscript-and-sql-server-2008-questions
6. http://drq.developpez.com/vb/tutoriels/ADO/Chapitre2/

- http://www.activexperts.com/activmonitor/windowsmanagement/adminscripts/filesfolders/folders/