

Utiliser le SDK OpenXML en VBScript

par bbil ([La page à bbil](#))

Date de publication : 4 mai 2009

Dernière mise à jour :

Avec cet article je vous propose de faire appel au **SDK OpenXML**, pour accéder, lire où modifier une valeur de cellule d'un classeur Excel .xlsx (2007). Tout en utilisant une **interface HTA** moins austère que du simple code VBScript en ligne de commande.
Vos impressions :

I - Principe.....	3
I-A - L'interface.....	3
I-B - OpenXML.....	3
II - L'application hta-openXML.....	3
II-A - Présentation.....	3
II-B - La partie interface HTA.....	3
II-C - Ouverture du classeur.....	7
II-D - Ouverture d'une feuille.....	8
II-E - Lire valeur.....	8
II-F - Ecrire valeur.....	9
II-G - Sauver feuille et Fermer classeur.....	10
III - Téléchargement.....	12
IV - Conclusion.....	12
V - Remerciements.....	12

I - Principe



I-A - L'interface

Pour donner une interface au script présenté dans cet article j'ai utilisé l'HTA, reportez vous à mon précédent article : **Donnez une interface à vos script VBS, HTA : Html Application**

I-B - OpenXML

Pour faire appel au SDK Open XML en VBScript, l'on va utiliser "Wrapper", vous le trouverez dans mon précédent article : **OpenXML en VB6 Lire et Ecrire des fichiers .xlsx** commencez donc par installer ce composant.

Procédure d'installation :

- 1 Nécessite : **NET Framework 3.5 SP1**
- 2 Outils :  **Kit de développement logiciel (SDK) Windows pour Windows Server 2008 et .NET Framework 3.5**
- 3 vb6-OpenXml :  **vb6-OpenXML.dll et install.bat**

Après l'installation du Framework .NET 3.5 et des outils en lignes de commandes du SDK, décompresser le contenu du fichier Distrib-vb6OpenXML.zip dans le répertoire "C:\Program Files\vb6-OpenXML" et exécuter install.bat

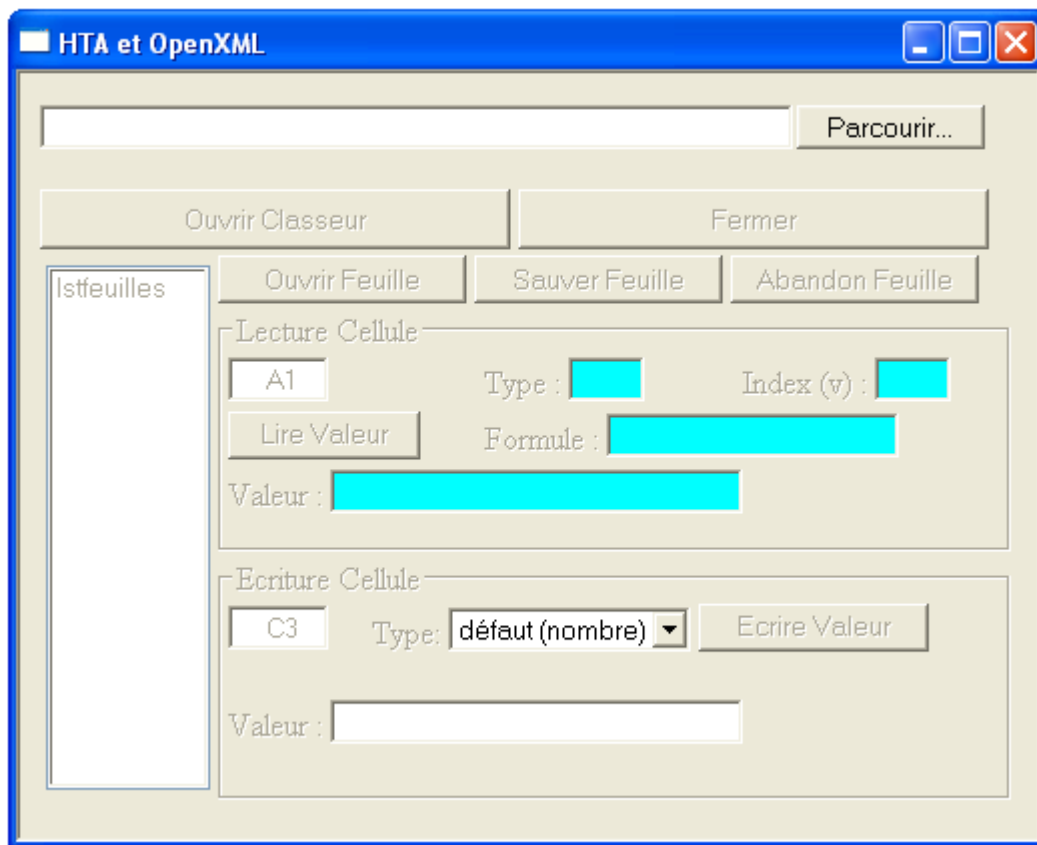
II - L'application hta-openXML

II-A - Présentation

Cette application est une adaptation VBScript de l'application exemple de **l'article VB6**. Je ne vous citerai donc que les différences par rapport à cette application.

II-B - La partie interface HTA

Celles ci est organisée à la manière d'une application style "boîte de dialogue" :



Le code HTA permettant d'obtenir ce rendu :

```
<html>
<head>
<title>HTA et OpenXML</title>
<HTA:APPLICATION ID ="HTA-OpenXML"
  APPLICATIONNAME="HTA-OpenXML"
  SINGLEINSTANCE="yes"
  VERSION ="0.1"
  SCROLL="no"
  >
<STYLE type="text/css">
  <!--
body { font-family: serif }

.label_affiche {
background-color : cyan
}
-->
</STYLE>

</head>
<script type="text/vbscript" src="inc/mdlOpenXML.inc">
</script>
<script language="VBScript">
(...)
</script>
<BODY bgcolor="buttonface" >
<INPUT TYPE="file" NAME="txtFic" SIZE="59" style="width:472px" onChange="choixfichier" ><BR/><BR/>
<input type="button" style="height:30;width:235px" value="Ouvrir Classeur"
name="cdOuvre" title="Ouvrir classeur sélectionné">
<input type="button" style="height:30;width:235px" value="Fermer"
name="cdClose" title="Fermeture du classeur" ><BR/>
<table border=0>
```

```

<tr height="10">
  <td rowspan="3">
    <SELECT NAME="lstFeuilles" SIZE=16 >
      <OPTION VALUE="lstfeuilles">lstfeuilles
    </SELECT>
  </td>
  <td width=400 >
    <input type="button" style="height:24;width:124px" value="Ouvrir Feuille"
      name="cdOuvrirFeuille" title="Ouvrir la feuille sélectionné">
    <input type="button" style="height:24;width:124px" value="Sauver Feuille"
      name="cdSauverFeuille" title="Sauver modifications feuille">
    <input type="button" style="height:24;width:124px" value="Abandon Feuille"
      name="cdAbandonFeuille" title="Abandon des modifications de la feuille">
  </td>
</tr>
<tr>
  <td HEIGHT=100>
    <fieldset style="height:120" id="FrmLectCell">
      <legend>Lecture Cellule</legend>
      <table width="100%">
        <tr>
          <td width="33%">
            <INPUT TYPE="text" NAME="txtCellule" SIZE="4"
              MAXLENGTH="30" VALUE="A1">
          </td>
          <td width="33%" border="1">
            Type : <INPUT TYPE="Text" readonly="readonly" name="lblType"
              class="label_affiche" SIZE="2">
          </td>
          <td >
            <label id="lblIndComment" >Index (v) : </label>
            <INPUT TYPE="Text" readonly="readonly" name="lblIndex"
              class="label_affiche" SIZE="2">
          </td>
        </tr>
        <tr>
          <td >
            <input type="button" value="Lire Valeur" name="cdLireValeur" >
          </td>
          <td colspan="2">
            Formule : <INPUT TYPE="Text" readonly="readonly"
              name="lblFormule" class="label_affiche" SIZE="20"></label>
          </td>
        </tr>
        <tr>
          <td colspan="3">
            Valeur : <INPUT TYPE="text" readonly="readonly" class="label_affiche"
              NAME="lblVal" SIZE="30" MAXLENGTH="30" >
          </td>
        </tr>
      </table>
    </fieldset>
  </td>
</tr>
<tr>
  <td HEIGHT=100>
    <fieldset style="height:120" id="FrmEcCell">
      <legend>Ecriture Cellule</legend>
      <table>
        <tr>
          <td width="20%">
            <INPUT TYPE="text" NAME="txtCellEcr" SIZE="4" MAXLENGTH="4" VALUE="C3">
          </td>
          <td> Type:
            <SELECT NAME="cbType" >
              <OPTION VALUE="0" SELECTED>défaut (nombre)
              <OPTION VALUE="1">Chaîne
              <OPTION VALUE="2">Booléen
            </SELECT>
          <!--</td>
        </tr>
      </table>
    </fieldset>
  </td>
</tr>

```

```
<tr>
  <td >-->
    <input type="button" value="Ecrire Valeur" name="cdEcrVal" >
  </td>
</tr>
<tr>
  <td colspan="2"><BR/>
  Valeur : <INPUT TYPE="text" NAME="txtVal" SIZE="30" MAXLENGTH="30" >
</td>
<tr>
</table>
</fieldset>
</td>
</tr>
</table>
</BODY>
</html>
```

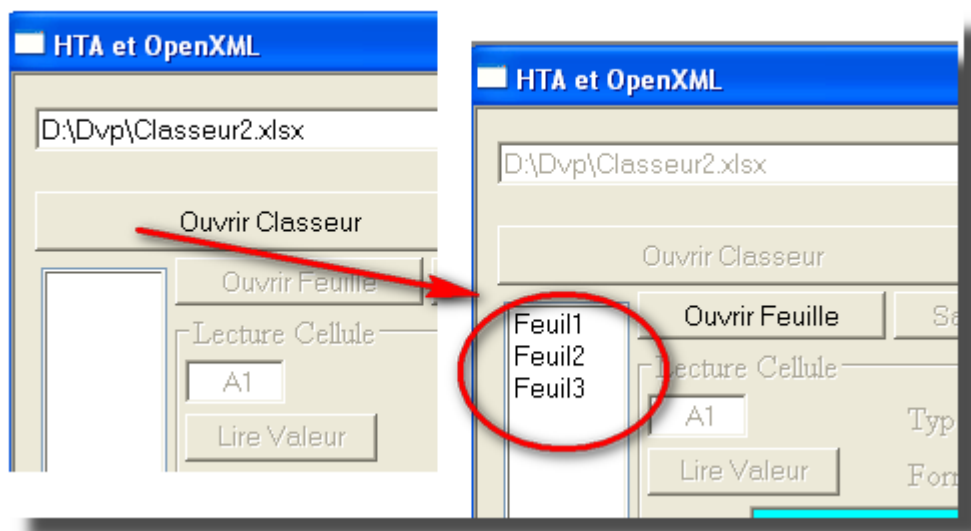
II-C - Ouverture du classeur

Même fonction qu'en VB6, aux déclarations de variables près, la seule différence importante et due à la gestion de la liste de sélection, différente des listbox en VB6.

```

'
' Ouverture classeur
'
Function OuvreClasseur(stNom, Lst ) 'As Long
On Error Resume Next
Set wk=CreateObject("vb6_OpenXml.oxWorkBook")
if Err.Number <> 0 Then
    OuvreClasseur = Err.Number
exit function
End if
On Error Goto 0
Dim xmlFeuille 'As IXMLDOMElement
Dim lRet 'As Long
Lst.Length = 0 ' vide Liste ...
bSSModif = False
lRet = wk.Open(stNom)
If lRet = 0 Then 'Ouverture classeur
Set wkXml = CreateObject("MSXML2.DOMDocument")
If wkXml.loadXML(wk.wkXml) Then 'Lecture WorkBook.xml
Set xmlListeFeuilles = wkXml.selectSingleNode("/workbook/sheets").childNodes
For Each xmlFeuille In xmlListeFeuilles
Set oOption = Document.createElement("OPTION")
oOption.Text = xmlFeuille.getAttribute("name")
oOption.Value = xmlFeuille.getAttribute("sheetId")
Lst.Add (oOption)
Next
Set SharedStringXml = CreateObject("MSXML2.DOMDocument")
SharedStringXml.loadXML wk.SharedStringXml
Else
    lRet = -1
    wk.Close
End If
End If
OuvreClasseur = lRet
End Function
    
```

J'ai laissé les déclarations de type de variables en commentaires, ce qui permet de connaître le type de donnée attendu.



Cette fonction, outre mettre à jour la liste de choix IstFeuilles avec le nom des feuilles composant le classeur, permet de renseigner deux variables de type "DOM document xml"

- **wkXml** : Le document xml principal du classeur (contient entre autre la liste des feuilles de celui-ci)
- **SharedStringXml** : Le document xml reprenant la liste des chaînes partagées.

II-D - Ouverture d'une feuille

La fonction LireFeuille, prends en paramètre le numéro de la feuille dans le classeur, et renseigne une variable globale de type "Dom Document XML" : shXML.

```
' Ouverture d'une feuille en lecture
'
Function LireFeuille(iNum ) 'As Boolean
    Dim oNode 'As IXMLDOMElement
    Set oNode = xmlListeFeuilles(iNum)
    Set shXml = CreateObject("MSXML2.DOMDocument") 'New DOMDocument30
    shModif = False
    shRelID = oNode.getAttribute("r:id")
    LireFeuille = shXml.loadXML(wk.Sh_Open(shRelID)) 'Lecture XML
    Set oNode = Nothing
End Function
```

L'ouverture de la feuille ouvre l'accès à l'appel des fonctions "Lire valeur" où "Ecrire valeur"

II-E - Lire valeur

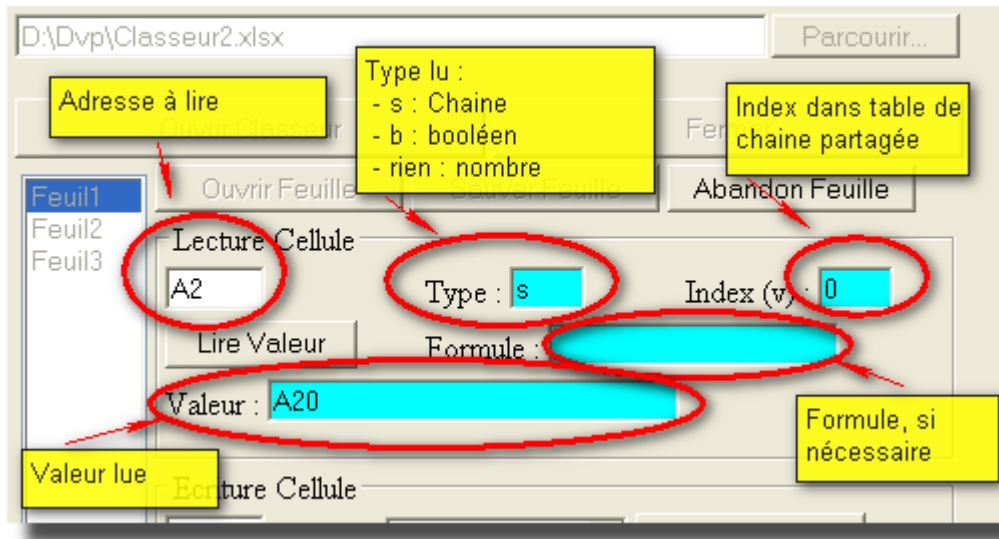
La lecture de la valeur d'une cellule est effectuée grâce à une requête XPath, sur l'XML de la feuille :

```
'
' Lecture d'une cellule
'
Function LectCell(stAdd, mCell_v, mCell_t, mCell_f) 'As Boolean
    Dim oCell 'As MSXML2.IXMLDOMElement
    Set oCell = ExisteCell(stAdd)

    If Not oCell Is Nothing Then
        On Error Resume Next
        mCell_v = oCell.selectSingleNode("v").Text 'Val
        mCell_t = oCell.getAttribute("t") 'Type
        mCell_f = oCell.selectSingleNode("f").Text 'Formule
        LectCell = True
    End If
End Function

'
' Test Existence Cellule, la renvoi si existe
'
Function ExisteCell(stAdd) 'As IXMLDOMElement
    On Error Resume Next ' Renvoi Nothing si Inexistant.
    Set ExisteCell = shXml.selectSingleNode("/worksheet/sheetData/row/c[@r='" & stAdd & "']")
End Function
```

La différence par rapport à la version VB6, est dû au fait que VBScript ne permet pas l'utilisation de type personnalisé, le paramètre mCell de type tyCell est remplacé par 3 variables.



Après saisie de l'adresse concernée une action sur le bouton "Lire Valeur" permet l'extraction du contenu de la cellule.

II-F - Ecrire valeur

L'écriture d'une valeur est effectuée grâce à la fonction écrire cellule :

```

'
' Ecrire la cellule
'
Function EcrireCellule(stAdd, MaCell_t, MaCell_v, MaCell_f)
    Dim oCell 'As MSXML2.IXMLDOMElement 'Cellule Cherchée
    Dim oVal 'As MSXML2.IXMLDOMNode
    Dim oSuiv 'As MSXML2.IXMLDOMElement 'Element suivant
    Dim oRow 'As MSXML2.IXMLDOMElement
    Dim MonAdd_iR 'As tyADD
    Dim MonAdd_iC
    Dim AddSuiv_iR' As tyADD
    Dim AddSuiv_iC
    Dim bSuivant 'As Boolean ' A 1 Celule suivante existe
    If shXml Is Nothing Then
        MsgBox "Ouvrir une feuille", vbCritical
        Exit Function
    End If
    CalcAdd stAdd, MonAdd_iR, MonAdd_iC 'Calcul ligne/Colonne.
    Set oCell = ExisteCell(stAdd)
    If Not (oCell Is Nothing) Then
        oCell.removeAttribute "t"
        EffaceChilds oCell
    Else
        Set oCell = shXml.createElement(1, "c", worksheetSchema)
        Set oRow = shXml.selectSingleNode("/worksheet/sheetData/row[@r='" & MonAdd_iR & "']")
        If Not (oRow Is Nothing) Then 'Cas ou ligne existe
            'L'ordre des cellules doit-être respecté, on cherche donc cellule suivante pour se placer
            'avant
            For Each oSuiv In oRow.childNodes
                CalcAdd oSuiv.getAttribute("r"), AddSuiv_iR, AddSuiv_iC
                If AddSuiv_iC > MonAdd_iC Then
                    bSuivant = True
                    Exit For
                End If
            Next
            If bSuivant Then
                oRow.insertBefore oCell, oSuiv
            End If
        End If
    End If
End Function

```

```

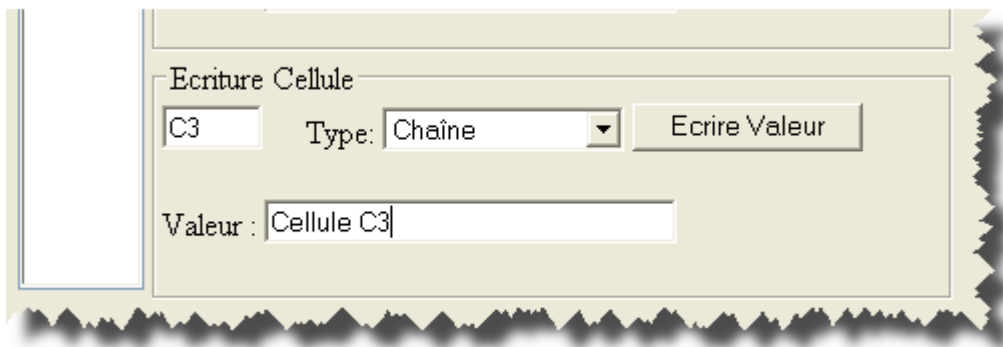
Else
    oRow.appendChild oCell
End If
Else ' Cas o ligne n'existe pas
Set oRow = shXml.createElement(1, "row", worksheetSchema)
oRow.setAttribute "r", MonAdd_iR
bSuivant = False
For Each oSuiv In shXml.selectNodes("/worksheet/sheetData/row")
    If CLng(oSuiv.getAttribute("r")) > MonAdd_iR Then
        bSuivant = True
        Exit For
    End If
Next
If bSuivant Then
    shXml.selectSingleNode("/worksheet/sheetData").insertBefore oRow, oSuiv
Else
    shXml.selectSingleNode("/worksheet/sheetData").appendChild oRow
End If
oRow.appendChild oCell
End If '-- Fin test existence oRoww..
End If

'-- Mise à jour des donnée cellule
oCell.setAttribute "r", stAdd
If MaCell_t <> "" Then oCell.setAttribute "t", MaCell_t
Set oVal = shXml.createElement(1, "v", worksheetSchema)
oVal.Text = MaCell_v
oCell.appendChild oVal
EcritCellule = True
shModif = True

End Function

```

Cette écriture est effectuée en respectant l'ordre des données dans la feuille.

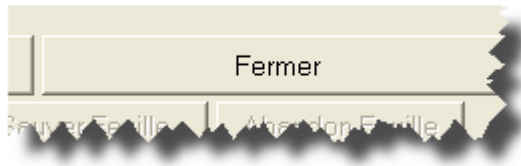


II-G - Sauver feuille et Fermer classeur

Une fois les modifications effectués, il convient de refermer le package grâce au boutons, sauver feuilles



et fermer classeur :



qui par l'appel des procédures FermerFeuille et FermerClasseur ce chargent d'enregistrer les changements dans la feuille et le classeur.

```

'
' Fermer feuille
'
Sub FermerFeuille( bSauve )

If bSauve And shModif Then
    If wk.sh_close(shXml.xml, shRelID) And bSSModif Then
        'Sauve chaines partagées.
        wk.SharedStringXml = SharedStringXml.xml
        bSSModif = False
    End If
End If
Set shXml = Nothing
shModif = False
End Sub
'
' Ferme le classeur
'
Sub FermerClasseur()
Set shXml = Nothing
If Not wk Is Nothing Then
    wk.Close
Set wk = Nothing
End If
Set SharedStringXml = Nothing
End Sub

```

III - Téléchargement

L'application HTA-OpenXML est composée de 2 fichiers :

- HTA-OpenXML.hta : le fichier principal de l'application, il contient l'interface et les procédures événementielles
- mdlOpenXML.inc : contient les différentes procédures et fonctions de l'application.

Le lien entre ces deux fichiers est effectué grâce à l'instruction :

```
<script type="text/vbscript" src="inc/mdlOpenXML.inc"></script>
```

au début du fichier HTA-OpenXML.hta.



L'application HTA-OpenXML

IV - Conclusion

Pour finir je vous propose d'utiliser la discussion :

Pour exprimer vos retours sur cette application.

Et répondre à cette question : verriez-vous l'évolution de ce wrapper VB6-openXML

- Plus de .Net en traitant la partie XML en .Net et simplifiant ainsi le code VBScript (où VB6).
- Moins de .Net en trouvant une autre librairie pour l'extraction des XML et ainsi devenir indépendant du Framework .Net
- Autre.

V - Remerciements

Tous mes remerciements à **ram-0000** pour sa relecture.